WHITEPAPER

Content Migration Approaches for Documentum

M. Scott Roth
Content Management Solutions Architect for Government Solutions

September 20, 2008

Table of Contents

Introduction 3
Preparation 4
Scenario 1 5
Scenario 2 7
Scenario 3 10
Scenario 4 13
Summary 19
References and Resources 21
About Flatirons Solutions 23
About the Author 23

Introduction

Recently, there has been a lot of discussion about content migration[1]; in the blogosphere, on the message boards, and in my company. Some of it may be fueled by the push to upgrade everyone to D6/6.5. Or, perhaps, everyone is virtualizing their infrastructure and needs to move Docbases and servers to VMware. Whatever the reason, everyone seems to be talking about it, including me.

After doing a quick survey of the projects we are pursuing at Flatirons Solutions, I came to the realization that a large portion of them involve content migration to some degree. The diversity of the projects and the degree to which content migration was required started me thinking about how each project should approach their migration needs. Some projects required the simplest of approaches while other required "a big hammer." The "big hammer" approach would, of course, address the needs of the simple migration, but at too great a cost. The converse was not true: the simple approach would not meet the needs of the complex migrations. In an effort to best address the needs of each project (i.e., cost, effort, time); I drew upon personal experiences and those of colleagues to put together this brief overview of common approaches for migrating content from one Documentum repository to another.
There are many reasons you might want to migrate content from one Documentum repository to another[2]. Here are the four most common scenarios I discovered in my survey of projects at Flatirons Solutions:

1. You need to migrate the Content Server to a more modern and powerful hardware platform. To do so, you need to clone the Documentum repository from the old platform to the new platform.
2. You need to migrate the Content Server to a different operating system and hardware platform. However, you are keeping the database vendor the same.
3. You need to migrate the Content Server to a new enterprise architecture. This requires changing the Content Server's hardware platform, the operating system and database vendors.
4. You need to migrate the Content Server to a new hardware and database platform. In the process, you need to implement a new doctype model and patch up the metadata.

Table 1 summarizes the primary challenges presented by each of these scenarios.

[1] Content migration is the act of moving content objects (content files and metadata) from one Documentum repository to another, while transforming or retaining the objects' source metadata. This activity is often called Extract, Translate and Load (ETL) in the database world.
[2] There are plenty of reasons for wanting to migrate content from other content management systems into Documentum also, but those scenarios are not the subject of this paper.

Table 1 - Summary of Migration Scenarios

Scenario	Change Hardware Platform	Change Operating System	Change Database Vendor	Change Object Model
Scenario 1	●			
Scenario 2	●	●		
Scenario 3	●	●	●	
Scenario 4	●	●	●	●

There are certainly many other reasons and scenarios for migrating content. As illustrated by these scenarios—and perhaps by the ones missing from this list—content migrations offer many challenges, and no two migrations are ever the same. Therefore, a Documentum solution provider needs to be knowledgeable of the many migration options available to him.

This paper offers an overview of migration scenarios paired with approaches, tools and best practices for each. It was not designed to provide a comprehensive approach for all migration scenarios. Rather, its intent is to provide an overview of approaches for how you might meet the challenges presented by each migration scenario. In the following pages, each of the scenarios outlined above will be expanded upon to discuss inherent challenges presented by each migration, and the best approach for meeting them. Therefore, you will find the level of detail for each approach varies among scenarios as the needs and challenges of each scenario vary. The end of this paper presents a lengthy list of articles, books, blogs, and tools to consult for further information.

Preparation

Before beginning any migration tasks, there are several preparatory things you should do:

- *Run housekeeping jobs* – Run Documentum's housekeeping jobs. Details regarding these jobs can be found in the *Seven Jobs Every Documentum Developer Should Know and Run* article referenced at the end of this paper. Resolve any issues identified by the Consistency Checker. There is no sense in migrating dirty data, unnecessary data or coping with referential integrity issues during the migration.
- *Fill out migration worksheet* – Fill in all the data on the migration worksheet (see reference at end of paper). This worksheet helps you plan your migration, ensures you have documented all the necessary account names and passwords, and estimate needed disk space. It is a little dated, but still relevant and helpful.
- *Verify connectivity and disk space* – Verify that the connectivity among the servers and other devices involved in the migration is adequate (i.e., bandwidth is sufficient to move large amounts of data) and that enough

free disk space is available for the temporary storage needed during the migration.

- *Verify downtime* – Verify that your planned downtime for the migration is adequate and well advertised. You may want to do some testing by copying large files across the network to gauge transfer rates. You should also plan your migration during a time when it will impact your customer the least. This usually means at night or over the weekend, but make contingencies plans for if it takes longer and overlaps expected operational hours.
- *Practice* – Practice the migration in a lab or VMware environment before attempting the real thing. Practice often identifies unknown and unexpected situations or problems. You know what they say about the best laid plans of mice and men[3].

After completing these preparatory activities, it is time to move on to addressing your specific migration scenario.

Scenario 1

The first scenario is very simple: you have the Content Server, the database server and the content files all on a homogeneous hardware platform. You need to move the Content Server to a more modern, powerful and robust platform. There is no need to change operating systems, hardware architecture, or Content Server versions. All you really want to do is clone the existing Docbase to the new hardware platform quickly and easily.

Challenge

The challenge in this scenario is how to make this migration quick, easy and complete. You also want to do it with minimal monetary investment in software tools.

Approach

Everyone's instinctive first approach to this scenario will be to use Documentum's Dump and Load operations. Dump and Load have been around since the beginning, but have received some criticism for not being user friendly or reliable. Dump and Load serialize and de-serialize object content and metadata as specified by a set of attributes on each Dump and Load object created in the Docbase. For simple migrations this is an acceptable process. However, Dump and Load have not kept up with the latest releases of Content Server and the operations can miss several very important object types in the Docbase, for example: workflow instances, doctypes without any instances, DocApps, registered tables, aspects and objects under records retention. Be aware of Dump and Load's short comings before attempting a migration using them. See the *Content Server Administrator's Guide* for more details.

[3] "Go often askew, and leave us nothing but grief and pain for promised joy!", *To A Mouse*, Robert Burns, 1785.

Dump

The act of creating and saving a Dump object (dm_dump_record) in the repository initiates the Dump process. The following steps outline the process for creating and saving a Dump object using an API script.

- Perform the preparatory steps outlined in the *Preparation* section above to make sure the Docbase is in prime condition for migration.
- Create a Documentum API script similar to the one following. This script has been adapted from the script described in the *Content Server Administrator's Guide* and will Dump the entire Docbase. There are many options available for the Dump operation, consult the guide for details.

```
create,c,dm_dump_record
# include the full path
set,c,l,file_name
c:\temp\dumpfile.out
# dump entire Docbase
set,c,l,dump_operation
full_docbase_dump
append,c,l,dump_parameter
# include content
set,c,l,include_content
T
# compress content
append,c,l,dump_parameter
compress_content=T
save,c,l
# check for dump errors
getmessage,c
```

- Run this script using the IAPI32 command line utility.
- It is also important to note that Dump will not dump content in the /System or /Temp cabinets, the repository owner user object or the docu group.

Load

Like the Dump operation, creating and saving a Load object (dm_load_record) initiates the Load operation.

- Run the Preload utility. This utility will examine the dump file and the target repository and identify objects that must be created in the target Docbase before loading the dump file.
- Run the DQL script created by the Preload utility to create the missing objects.
- To start the Load operation, create a Documentum API script similar to the one following. This script has been adapted from the script described in *Content Server Administrator's Guide*.

```
create,c,dm_load_record
# include the full path
set,c,l,file_name
c:\temp\dumpfile.out
```

```
save,c,l
# check for load errors
getmessage,c
```

- Run this script using the IAPI32 command line utility.
- Remember, Load will not load content for `/System` or `/Temp` cabinets, the repository owner user object or the docu group. See the *Content Server Administrator's Guide* for more details.
- It is also important to note that the target Docbase cannot have the same Docbase ID or Docbase name as the source Docbase.
- Finally, run the Consistency Checker and State of the Docbase jobs (see the *Seven Jobs Every Documentum Developer Should Know and Run* article referenced at the end of this paper for details). Resolve any issues identified by the Consistency Checker. Compare the State of the Docbase report generated during your preparatory steps with the one generated in the new environment. This comparison should be a good indicator of the thoroughness of your migration.

The bottom line with this approach is that it is simple and utilizes built-in capabilities and tools. However, there are some potentially serious drawbacks to using the Dump and Load approach. Make sure you understand the limitations and practice your migration.

Alternate Approach

Alternatively, if the migration described by this scenario is really just a virtualization of your existing Content Server, then a simpler approach is to use VMware's Converter utility. The Converter utility will create an exact copy of a physical system in a VMware image. See the VMware Converter reference at the end of the paper.

Tools

The software tools used in this scenario are:

- `%DM_HOME%/bin/IAPI32#`
- `%DM_HOME%/bin/PreLoad#`
- VMware Converter (see reference at end of this paper).

Scenario 2

In this scenario, you need to migrate the Content Server to a more modern, powerful and robust hardware platform and change the operating system. The database will remain Oracle. On the surface, changing the OS might seem like a big deal, but it really isn't. A change in the Content Server OS would be important if our migration tools were OS-specific, but they aren't. Therefore, the OS changing is a bit of a red herring in this scenario.

Ignoring the OS factor then, this scenario reads much like Scenario 1. Therefore, the approach suggested in Scenario 1 is certainly valid. However, for interest and variety, assume the Docbase is too complicated to be migrated using Dump and Load.

Challenge

The challenge in this scenario is to create an exact copy of a Docbase on an entirely different hardware and operating system platform. The tools and approach must, therefore, be cross-platform.

Approach

The approach described here works under the covers of the Content Server at the database and content store levels. Essentially we are going to clone the database and slip it in under the new installation of the Content Server on the target platform. We will then move the content stores to the new platform thus creating an exact copy of the repository on the new hardware and OS platform. For the sake of our scenario, assume the source Docbase is on Windows and the new Docbase will run on Solaris.

The details of this approach are described in the Documentum presentation, *Upgrading to Documentum Server 5.1*. Though some of the details of that presentation are dated, the overall approach is still valid. A reference to this presentation can be found at the end of the paper.

Database Cloning

Begin by creating an exact copy of the Content Server's database (i.e., a clone).

- Perform the preparatory steps outlined in the *Preparation* section above to make sure the Docbase is in prime condition for migration.
- Install the exact same version of the database and the Content Server on the target environment. If this migration requires an upgrade also, make sure you perform the upgrade such that you always migrate content to the exact same version of the Content Server. This means you perform the upgrade before or after migrating the content.
- When installing the Content Server in the target environment, use the same installation owner as the source Content Server and create a Docbase with the exact same name, ID, and owner as the source Docbase.
- Shutdown the target Docbase.
- Logon to the target database and drop all of the tables and views in the Docbase schema. You can drop the existing database tables using the `dm_DeleteTableSpace.sql` script in the `%DOCUMENTUM%/dba/config/<repository_name>` directory.

- Shutdown the source Docbase and export the database tables and views using the Oracle exp command line utility.[4] Consult the Oracle documentation for syntax and options for the exp and imp commands.
- Import the database tables and views into the target database using the Oracle imp command.

Copy Content

Copy all of the content from the source system to the target system. This can be accomplished in a number of ways: Microsoft Explorer, Samba, XCopy, FTP, ZIP, TAR, cp, etc. There are two key things to remember with this copy:

- Make sure you copy *all* of the content storage areas and retain the file ownership and permissions;
- The path on the target system must be *exactly* the same as the path on the source system.

If either of these two properties change, additional OS and database changes must be made. See the *Upgrading to Documentum Server 5.1* and/or *The Content Server Installation Guide* document for details.

Reconfigure Target Server

- Verify that the contents of the server.ini and the dfc.properties files on the target system match those on the source system.
- Start the target Docbase.
- Update the hostname on the target system using DQL:
  - update dm_mount_point_s set host_name = '<*target server name*>';
  - update dm_server_config_s set r_host_name = '<*target server name*>';
- Update the job targets on the target system using DQL:
  - update dm_job_s set target_server = '<*target server name*>' where target_server = '<*source server name*>';
- Invalidate the views on the target system so they will be rebuilt, using DQL:
  - update dm_type_s set views_valid = 0;
- Finally, run the Consistency Checker and State of the Docbase jobs (see the *Seven Jobs Every Documentum Developer Should Know and Run* article referenced at the end of this paper for details). Resolve any issues identified by the Consistency Checker. Compare the State of the Docbase report generated during your preparatory steps with the one generated in the new environment. This comparison should be a good indicator of the thoroughness of your migration.

This approach involves a little database and file system trickery to achieve its goal. Other than sufficient attention to detail and enough storage space to

[4] If your migration is from Windows/SQL Server to Windows/SQL Server (as opposed to the Windows/Oracle – Unix/Oracle in this example), you can use the SQL Server Copy Database Wizard, or create a backup of the database to achieve the same purpose.

accommodate the temporary storage of the export tables, this approach is straightforward. The fact that the OS changes from the source to the target Docbase is completely nullified by moving the database tables via vendor-provided tools, and copying the file stores.

Alternate Approach

As mentioned earlier, Dump and Load may be a valid alternate approach for this scenario depending upon the complexity of the Docbase. In addition, any of the other approaches mentioned in this paper would be equally valid as well.

Tools

The tools needed to implement this approach are all native to their respective database vendors:

- `%ORACLE_HOME%/bin/exp`
- `%ORACLE_HOME%/bin/imp`
- `TAR`
- `FTP`

Scenario 3

In the third scenario you are changing three major variables in the repository configuration: the hardware platform, the operating system for the Content Server, and the database vendor. This scenario is the most common and can prove to be very challenging because it has so many variables.

The approach to this scenario must neutralize as many of these variables as possible. In the previous scenario, we overcame some of these challenges by using database vendor tools, but in this scenario we don't have that luxury because the database differs between the source and target environments. We have already established that Dump and Load is not the best approach for complex Docbases, so that option is eliminated also. It should also be apparent that we can't use the Oracle `exp/imp` tools to migrate data to/from a Microsoft SQL Server database. The best way I know to neutralize these challenges is to work directly through the Content Server API. The Content Server then shields you from the details of implementing on different hardware, OSes and databases.

Challenge

The challenge in this scenario really lies with changing the database vendor although changing the hardware platform and OS add drama to the scenario. Unlike the previous scenarios where you could export the database and re-import it in the new environment, you cannot easily clone a database among different vendors.

Approach

Unlike the Docbase cloning approach that treated the repository metadata and content separately, this approach addresses them both concurrently. The recommended approach in this situation is to use the Documentum Application Builder (DAB) to build data DocApps to transport the repository content across the hardware/OS/database transom to the target environment. The data DocApps bundle object metadata and content into environment-neutral archives that can be easily installed in the target environment using the Documentum Application Installer (DAI).

Create Data DocApps

Begin by creating an initial DocApp that contains all of the repository's customizations (e.g., doctypes, workflows, etc.). Then create DocApps that contain the actual content. The steps for this approach are outlined below.

- Perform the preparatory steps outlined in the *Preparation* section above to make sure the Docbase is in prime condition for migration.
- Analyze the content. Determine how best to carve up the content in the Docbase so that each resulting DocApp is between 5 – 8 GB in size. Experience shows that this size DocApp is the maximum capacity for DAI. Don't be fooled by the apparent simplicity of this analysis; this analysis is key to making this approach work.
- Create an initial DocApp that contains all of the repository's custom doctypes, permission sets, workflow templates, lifecycles, methods, etc., but no content.
    - **NOTE**: users, groups and ACLs cannot be included in a DocApp. You will need to recreate these objects manually (or with a script) on your target Docbase.
- If you have never created a DocApp that contains objects already resident in a Docbase, you may want to consult the DAB documentation. Briefly, here are the steps:
    - Create a new DocApp
    - Choose **Insert->Objects From Docbase->Object Type** from the menu. Select object type from the Docbase.
    - Repeat for all custom object types in the Docbase.
    - Once everything has been added to your initial DocApp, choose **DocApp->Set Installation Options** and ensure that all of the objects are set to **Overwrite in the Target Docbase**.
    - Choose **DocApp->Checkin** and save the DocApp in the Docbase.
    - Finally, choose **DocApp->Create DocApp Archive** from the menu to save the DocApp to the local file system.
- After creating your initial DocApp, begin creating data DocApps in the same manner. Insert cabinets, folders and objects that contain content in accordance to the content analysis you performed. Don't forget to include doctypes with the data DocApps, otherwise DAB will not include the content.

- Checkin the DocApps, set the installation options to include content, and export the DocApps by choosing **DocApp->Create DocApp Archive**.
- Repeat until you have created DocApps for all the content in the Docbase.
- It is likely that DAB will fail with an **Out of Memory Error** when you try to create large data DocApp archives. You will need to adjust (maximize) the amount of memory allocated to the Java heap, and the size of your DocApp to find the right balance. In order to give the maximum heap space to the JVM, run DAB on a workstation with 4GB of RAM. For details on adjusting the JVM heap, see the references at the end of this paper.

Install Data DocApps

- Create users, groups and ACLs in the target Docbase either manually or using a script.
- Start the Documentum Application Installer (DAI) and login as the Docbase owner.
- Using the DAI screen controls, open the initial DocApp and install it. Watch the log messages to ensure the installation was successful.
- After the initial DocApp is loaded, systematically install all of the data DocApps.
- If you are installing from a different workstation than the one on which you created the DocApps, you will need to maximize the amount of memory allocated to the Java heap on this machine too. For details on adjusting the JVM heap, see the references at the end of this paper.[5]
- An odd side effect of using data DocApps, is that all of the content will be installed in the /System/Applications/<DocApp Name> folder. You will need to move it to its proper cabinet when the installation is completed.
- Finally, run the Consistency Checker and State of the Docbase jobs (see the *Seven Jobs Every Documentum Developer Should Know and Run* article referenced at the end of this paper for details). Resolve any issues identified by the Consistency Checker. Compare the State of the Docbase report generated during your preparatory steps with the one generated in the new environment. This comparison should be a good indicator of the thoroughness of your migration. The good news is, if you missed something, simply go back and create another DocApp to capture the missing content.

Using data DocApps to migrate content is a great approach, which neutralizes all of the environment variables (platform, OS and database) by operating completely within the Documentum framework and API. The drawbacks are that you can't selectively migrate data based upon doctype, only based upon its location in the repository (e.g., cabinets or folders), and the content ends up in the DocApp's application folder. Another drawback is the inherent instability of DAB and DAI and finding the threshold at which the JVM runs out of memory.

[5] You can also run DAI from the command line and control the Java heap with –Xms and –Xmx arguments. The command looks something like this: `java -Xms256mb -Xmx256mb com.documentum.ApplicationInstall. DfAppInstaller -d <docbase> -n <user> -p <password> -a <docapp> -l <log file> -f <properties file>.`

Alternate Approaches

The only viable alternative for this scenario is a custom tool or one of the third part tools described in the next scenario.

Tools

- Documentum Application Builder
- Documentum Application Installer
- Webtop (to move content after installation)

Scenario 4

The final scenario to consider is one that contains a little bit of everything. We must deal with the "three variables" discussed in Scenario 3, but in addition, the content of the source Docbase must be split into two different doctypes based upon a business rule, the object name will become a combination of other metadata values, and the title must be trimmed.

Specifically, the original Docbase contains standard operating procedures (SOP) for a small manufacturing company. Over the course of time, these SOPs have taken on two distinctly different applications: one for the manufacturing of widgets, the other for running the business. Unfortunately, the current object model does not distinguish between the two, so users have resorted to giving the SOPs creative titles to distinguish them. SOPs related to the manufacturing of widgets have a title that starts with "M-", while SOPs related to the business have a title that start with "B-". During the migration, you must distinguish between the manufacturing SOPs and the business SOPs and create `man_sop` objects or `bus_sop` objects, respectively. You must also store them in separate folders in the SOP cabinet.

As for the metadata, the following metadata mapping rules need to be implemented during the migration:

- The `object_name`, which could contain anything in the source repository, must be the concatenation of "M-" or "B-" (for manufacturing or business depending upon which SOP type it is), the SOP number and the title of the SOP. For example: "M-0001-Extrusion Molds" or "B-021-Electronic Funds Transfers".
- The `title` should have the leading "M-" or "B-" removed if it has one.

Challenge

The challenges in this scenario are to identify, segregate and move content based upon specific business rules, and apply metadata conversion and mapping rules as the content is migrated from one repository to another. The hardware platform, OS and databases are also different between the source and target environments.

Approach

The complexity of this migration demands a custom or 3rd party tool. Although custom written migration tools are always an option (and fun to write!), their cost and general applicability from one migration to another is usually limiting, and therefore will not be considered here. Instead, this approach will focus on the use of 3rd party tools. There are a number of 3rd party tools available to accomplish this kind of migration; Table 2 contains a list of several.

Table 2 – 3rd Party Migration Tools

Tool	Vendor	URL
OpenMigrate	TSG	www.tsgrp.com
DocLoader/Solo	McLaren Software	www.mclarensoftware.com
DIXI[6]	Blue Fish Group	www.bluefishgroup.com
Buldoser	Crown Partners	www.crownpartners.com
DocMigrator/ SysMigrator	Generis	www.generiscorp.us
migration-center	fme AG	www.migration-center.de
Q-Transfer	Impact Systems	www.impactinfosys.com
TadsBits	Theodore Watson	www.tadsbits.com

For this scenario I chose to use OpenMigrate, an open source migration tool created and maintained by the Technology Services Group (TSG). Why? Because it is an open source/free resource that is available to everyone. Most of the other tools listed in Table 2 are cost prohibitive for a small migration of this sort.

OpenMigrate (OM) is a Java application framework built using the Spring Framework and Hibernate. OM contains a migration engine that handles the logistics of moving generic migration nodes from a source to a target. The migration nodes are defined using an interface that abstracts the specific implementation from the engine. The source and target are abstracted via adaptors that plug into the framework. Out-of-the-box, OM contains source and target adaptors for Documentum and the file system.

OM relies heavily on a lot of XML configuration files that are injected into the framework at various times and override each other at various levels. This has many advantages and disadvantages: it makes the tool very flexible and allows non-programmers to configure complex migrations by simply writing or tweaking XML files. The disadvantage is that these XML files are not for the faint of heart. A thorough understanding of the OM architecture and its features is a must for writing usable configuration files. I found the documentation for OM and the config files a little sparse. After obtaining the source code from TSG, setting up an Eclipse project and running the sample projects from the IDE, the

[6] Shortly after completing the research for this article, Blue Fish decommissioned DIXI and replaced it with their new content migration product, Migration Work Bench.

configuration was a little clearer, but still not for the faint of heart. If you choose to use OM, I suggest contacting TSG for help in getting things setup and your project started.

There are three primary XML configuration files that drive the migration process for this scenario. They are:

- `app-ctx.xml`—this is the primary configuration file. In it you specify the migration source adaptor, target adaptor and the process pipeline. OpenMigrate comes with an application context file for Documentum-to-Documentum migrations. It shouldn't require any changes for our migration.
- `config.xml`—this file contains additional configuration information and is generally only necessary for the Spring Framework and the Jakarta Commons configuration. In our migration, it simply points to the configuration details file.
- `config-details.xml`—this file contains target and source-specific mapping information such as doctypes, attribute names, attribute values, etc.

There is also a properties file:

- `ctx-placeholders.properties` – this file holds configuration information for the queue populator (a DQL query used to select qualified objects to migrate) and repository access (i.e., login information). The properties defined here are injected into the `app-ctx.xml` file when needed.

config-details.xml

The configuration details file is where all the interesting things in this migration happen. The first two sections simply indicate which doctypes you will deal with in this migration. From the source repository, you will process all sop_docs, and in the target repository man_sops and bus_sops.

```
<config>
      <source-dctm-main>
            <objects>
                  <object>
                        <description>Settings for sop</description>
                        <criteria>
                              <attribute name="om_object_type"
                                    value="sop_doc" type=
                                    "java.lang.String"/>
                        </criteria>
                        <renditions formats=".*"/>
                        <versions values=".*"/>
                  </object>
            </objects>
      </source-dctm-main>

      <target-dctm-main>
```

```
        <objects>
            <object>
                <description>Setting for man and bus sop
                </description>
                <criteria>
                    <attribute name="om_object_type"
                    value="man_sop | bus_sop"
                    type="java.lang.String"/>
                </criteria>
            </object>
        </objects>
    </target-dctm-main>
```

Next, we map the business SOPs to the bus_sop type and the manufacturing SOPs to the man_sop type. To accomplish this, we use a simple regular expression in the `attribute.value` criteria. If the SOP's title begins with *B-*, it is assumed to be a business SOP. If the title begins with *M-* it is assumed to be a manufacturing SOP.

```
    <mappings>
        <type-mapping-bus>
            <name>bus sop type mapping</name>
            <description>mapping for bus sop type</description>
            <mapping-criteria>
                <attribute name="om_object_type" value="sop_doc"
                    type="java.lang.String"/>
                <attribute name="title" value="^B-.*"
                    type="java.lang.String"/>
            </mapping-criteria>
            <attribute-mappings>
                <attribute-mapping attr="om_object_type"
                    type="java.lang.String">
                    <value val="bus_sop"/>
                </attribute-mapping>
            </attribute-mappings>
        </type-mapping-bus>

        <type-mapping-man>
            <name>man sop type mapping</name>
            <description>mapping for man sop type</description>
            <mapping-criteria>
                <attribute name="om_object_type" value="sop_doc"
                    type="java.lang.String"/>
                <attribute name="title" value="^M-.*"
                    type="java.lang.String"/>
            </mapping-criteria>
            <attribute-mappings>
                <attribute-mapping attr="om_object_type"
                    type="java.lang.String">
                    <value val="man_sop"/>
                </attribute-mapping>
            </attribute-mappings>
        </type-mapping-bus>
```

The next section of the `config-details.xml` file sets up the rules to make sure the manufacturing and business SOPs are stored in the appropriate folders when they are migrated.

```
<folder-mapping-man>
      <name>man sop folder mapping</name>
      <description>mapping for man sop folders</description>
      <mapping-criteria>
            <attribute name="om_object_type" value="man_sop"
                  type="java.lang.String"/>
      </mapping-criteria>
      <attribute-mappings>
            <attribute-mapping attr="om_folder"
                  type="java.lang.String">
                  <value val="/SOPs/Manufacturing"/>
            </attribute-mapping>
      </attribute-mappings>
</folder-mapping-man>

<folder-mapping-bus>
      <name>bus sop folder mapping</name>
      <description>mapping for bus sop folders</description>
      <mapping-criteria>
            <attribute name="om_object_type" value="bus_sop"
                  type="java.lang.String"/>
      </mapping-criteria>
      <attribute-mappings>
            <attribute-mapping attr="om_folder"
                  type="java.lang.String">
                  <value val="/SOPs/Business"/>
            </attribute-mapping>
      </attribute-mappings>
</folder-mapping-bus>
```

This final section sets up the mapping rules for the `title` and `object_name` attributes. The first section removes the *M-* and the *B-* from the beginning of the `title`, if they exist, using the built-in *`ReplaceChars()`* function. The final two sections construct the object's name by concatenating the SOP's number and title with the *M-* or *B-* indicator.

```
<default-mapping>
      <name>default sop mapping</name>
      <description>General Mapping for sops</description>
      <mapping-criteria>
            <attribute name="om_object_type"
                  value="man_sop | bus_sop"
                  type="java.lang.String"/>
      </mapping-criteria>
      <attribute-mappings>
            <attribute-mapping attr="title"
                  type="java.lang.String">
                  <value val=
                  "${XReplaceChars~title~/M-|B-///"/>
      </attribute-mapping>
```

```
                  <mapping-criteria>
                        <attribute name="om_object_type"
                              value="man_sop"
                              type="java.lang.String"/>
                  </mapping-criteria>
                  <attribute-mappings>
                        <attribute-mapping attr="object_name"
                              type="java.lang.String">
                              <value val="M-${sop_number}-${title}"/>
                  </attribute-mapping>

                  <mapping-criteria>
                        <attribute name="om_object_type"
                              value="bus_sop"
                              type="java.lang.String"/>
                  </mapping-criteria>
                  <attribute-mappings>
                        <attribute-mapping attr="object_name"
                              type="java.lang.String">
                              <value val="B-${sop_number}-${title}"/>
                  </attribute-mapping>
            </default-mapping>
      </mappings>
</config>
```

ctx-placeholders.properties

Make sure the source and target repositories are defined and login credentials provided. Also, set up the query that will pump data into the queue. For this scenario, the property looks like this:

```
#Queue Populator Query
queuepop.query=SELECT * FROM sop_doc
```

Migration

- Perform the preparatory steps outlined in the *Preparation* section above to make sure the Docbase is in prime condition for migration.
- Run OpenMigrate using the configuration files described above. I found it easiest to run OpenMigrate using the source code in Eclipse. However, it can also be run using the provided batch file:

  ```
  >OpenMigrateCL.bat -config app-ctx.xml
  ```

- Run the Consistency Checker and State of the Docbase jobs (see the *Seven Jobs Every Documentum Developer Should Know and Run* article referenced at the end of this paper for details). Compare the State of the Docbase report generated during your preparatory steps with the one generated in the new environment. This comparison should be a good indicator of the thoroughness of your migration. Of course, in this scenario the before and after Docbases are radically different, so you may want to employ some other quality control methodology.

OpenMigrate is a fantastic open source ETL tool for Documentum. The price is obviously right, and with a little effort and trial and error, the configuration isn't too daunting. It is also possible to write Java plugins for the framework if you need additional functionality that cannot be achieved with the configuration files and built-in transformers. The only drawback, again, is the lack of thorough documentation and tutorials, though the sample projects are very helpful.

Tools

- OpenMigrate

Summary

Many of the approaches presented here will work for many other scenarios in addition to the one for which it was presented. I associated each approach with the scenario where it made the most sense and provided the most benefit. Table 3 contains a summary of each approach's applicability to each of the scenarios described in this paper. The 3rd party tools that are truly generic ETL tools are applicable to all of the scenarios. The question that must be answered then is: Is the tool worth the expense or can an alternate approach work just as well? If an alternate approach is suitable, then I hope I have provided you with enough information to get you headed in the right direction.

Table 3 - Summary of Approaches vs. Scenarios

Approach	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Dump & Load	●#	#	#	#
VMware Converter	●#	#	#	#
Database Cloning	●#	●#	#	#
Data DocApps	●#	●#	●#	#
3rd Party Tools	●#	●#	●#	●#

As I stated earlier, no two migrations are ever the same, they all have unique requirements and nuances, and this paper has not tried to address them all. For example, some additional challenges you might face in your migration are:

- How do you migrate content (and metadata) from another content management system to Documentum?
- How do you bulk importing content from a file system to Documentum?
- What happens if you encounter an unexpected problem half way through the migration? How do you roll it back?
- How do you best deal with content that accumulates in the source system during the migration?

The answer to most of these questions is to use a 3rd party tool designed for versatility with the necessary safeguards to roll back mistakes. As I was finishing this paper, I learned about a new approach and tool for content migration from the Blue Fish Group called Agile Migration and the Migration Workbench, respectively. This approach and new tool address these problems (and many others). My experience with the tool has been very positive.

In closing, I hope this paper has given you some direction, some insight and some ideas, so as you face a content migration challenge—and you will—you feel well versed in the approaches and tools available to you.

Please drop me a note and let me know what you think about this paper and the ideas it contains: scott.roth@flatironssolutions.com.

Thanks to Todd Pierzina at TSG for all of his help with OpenMigrate, and to all the Flatirons Solutions engineers who knowingly and unknowingly contributed to this paper!

References and Resources

The techniques discussed in this paper were developed and tested using Documentum 6.0.

Here are some links and references to resources cited, mentioned or otherwise consulted in this paper.

Blogs

- Documentum for Dummies: http://d4d2.com/?cat=55
- Ask Johnny!: http://johnnygee.wordpress.com/?s=migration
- Word of Pie: http://wordofpie.wordpress.com/?s=migration

Articles

- *Seven Jobs Every Documentum Developer Should Know and Use*, http://developer.emc.com/developer/edn_redirect_secure.htm?redirectURL=http://developer.emc.com/developer/Articles/SevenDCTMJobs.pdf
- *Content Migration: Seven Steps to Success*, http://www.vamosa.com/content_migration_seven_steps_to_success.pdf
- *Technical Challenges Faced During Content Migrations*, http://www.bluefishgroup.com/library/2007/technical-challenges-faced-during-content-migrations/
- *Common Content Migration Scenarios*, http://www.bluefishgroup.com/library/2007/common-migration-scenarios/
- *Agile Migrations – A Revolutionary Approach to Content Migrations,* http://www.bluefishgroup.com/content-migrations/agile-migrations.php

Websites

- JVM heap: http://javahowto.blogspot.com/2006/06/6-common-errors-in-setting-java-heap.html
- Control Panel: http://forums.java.net/jive/thread.jspa?threadID=1292
- Oracle FAQ: http://www.orafaq.com/wiki/Import_Export_FAQ

EMC Developer Site / PowerLink

- Developer Site Migration section: http://developer.emc.com/developer/migration.htm
- *Upgrading to Content Server 5.1*: http://developer.emc.com/developer/downloads/MigrationUpgradingToServer5.pdf
- Migration Worksheet: http://developer.emc.com/developer/downloads/TemplateSectionSamples.zip
- Documentum Migration/Upgrade Forum: https://forums.emc.com/nsepn/webapps/xpsgggfs31465mnldcpq14543562/forums/forum.jspa?forumID=4

- https://forums.emc.com/nsepn/webapps/xpsgggfs31465mnldcpq14543562/forums/thread.jspa?tstart=0&threadID=56686

Documentum Documentation

- *Web Content Management Best Practices: WCM Custom DocApp Migration*, v2.0, 9/7/2004. (Contact EMC Consulting for a copy of this paper).
- *Documentum Content Server Administration Guide*, v6, 2007.
- *Documentum Content Server Installation Guide*, v6, 2007.
- *Documentum Application Builder User Guide*, v5.3, March 2005.

Books

- *A Beginner's Guide to Developing Documentum Desktop Applications*, M. Scott Roth, 2005, ISBN: 0-595-33968-9

Tools

- OpenMigrate: http://www.tsgrp.com/Open_Source/OpenMigrate/open-migrate.jsp
- OpenMigrate demo video: http://tsgrp.com/multimedia/MillenniaGroup_TSG/MillenniaGroup_TSG.html
- TadsBits: http://www.tadsbits.com
- VMware Converter: http://www.vmware.com/products/converter/
- DocLoader: http://www.mclarensoftware.com/ms/products/Docloader_Promo.asp
- DIXI: http://www.bluefishgroup.com/content-migrations/dixi.php
- Buldozer: http://software.emc.com/microsites/application_portfolio/collateral/data_sheets/Crown_ds_BULDOSER.pdf
- DocMigrator/SysMigrator: http://generiscorp.us/migration_tools
- migration-center: http://fme.de/Product.178.0.html?&L=1
- Q-Transfer: http://www.impactinfosys.com/downloads/Q-Transfer%20data%20sheet.pdf
- TadsBits: http://www.tadsbits.com/

About Flatirons Solutions

Flatirons Solutions Corp., an Inc. 500 company, provides consulting, systems integration, and systems & software engineering services to Fortune 500 companies and government agencies. A leading content management solutions provider specializing in XML-based publishing and digital asset management, Flatirons has provided enterprise-wide solutions in industries such as high technology, aerospace, transportation, publishing, manufacturing, financial services, insurance, media and entertainment, retail, and healthcare. Flatirons Solutions also actively participates in both DITA and DocBook XML technical committees. Established in 2001, Flatirons Solutions is a privately-held company headquartered in Boulder, Colorado, with offices in Washington D.C. and Ft Worth, TX. For more information visit Flatirons Solutions on the web at http://www.FlatironsSolutions.com.

About the Author

M. Scott Roth is the content management solutions architect for Government Solutions at Flatirons Solutions Corp. He is also the author of the book, *A Beginners Guide to Developing Documentum Desktop Applications*, and the open source Documentum command line client, DOCS.

<SDG><

4747 Table Mesa Drive, Suite 200
Boulder, Colorado 80305
(303) 544-0514
info@FlatironsSolutions.com

www.FlatironsSolutions.com