DEMA USER GUIDE

Making Migration Easier and Faster

EMA TEAM

[1.6.0]

Table of Contents

INTRODUCTION

Definition	DEMA (Documentum Enterprise Migration Appliance) is a suite of tools that enable a consultant to move data, content, even whole repositories from point to point in the enterprise or to the cloud. • Check the training recordings - https://inside.emc.com/groups/ecd-architects/blog/2016/03/17/updated-ema-training-recordings-available?sr=stream • Check the training simulations present at "EMA1.6.0\Training" folder

Components	**EMA-Cloner**	Used when migrating an entire repository and there is a database change from Oracle to SQL Server.	
	EMA-Migrate	Used when you have a typical ETL requirement and you are migrating parts of a repository and not the entire repository.	
	EMA-API	Java API's to help you extract data from a third party source like a CSV, YAML or database where Documentum is not the source system to build custom adaptors.	
	EMA-Tools	Morph	Used to do mass object type changes within a repository. Object ids remain the same, inflight workflow retain their state and audit trail stays intact.
		Replatform	Used to update hostname entries in configurations settings stored inside the repository. Can also be used to modify configurations when moving from Unix to Windows and vice versa.
		File Copier	Used for content copy when the content has not been already copied before Ingestion is run. Picks up the FileList generated by the Ingestor and runs a multi-threaded copying process.
		Transform	Used when the data is not to be migrated as is and needs to be modified as per business requirements.
		Link Count Update	Tool to update the link count of a folder. • Required when there is a transformation being done which requires moving documents to different folders. • Not required when documents are not moved to new folders as part of transformation or if D2-Core job will be used to apply auto-linking rules.
		Folder Structure Generator	Used to generate a folder structure based on a simple text file containing a list of folders. A sample file "folderListSample.txt" is provided in the samples directory.
		Compare	Used to compare an object pre/post-transformation. New properties, deleted properties, and modified properties will be displayed in the output with before/after values.
		Data Verifier	Used to test the compatibility of a MongoDB database with the target DB schema into which it is intended to be ingested. This is a quicker and more efficient approach than running dry-runs until all INSERTs pass.
		Default File Creator	Used to generate default files (required during Ingestion) for the types specified.
		Type Extractor	Used to extract the types present in the source system. It generates a dql file which can be run in the target system to

		create the corresponding types.
	Audit Trail Extractor	Used to extract the audit trail from the system.
	Group User Extractor	Used to extract the users and groups in a particular system.
	Encrypt Utils	Used to encrypt passwords and can be used in all the EMA components wherever passwords are being used.
	Content Migrator	Helps in moving data from Centera to Isilon
	Reference Update	Updates references to relations etc. if everything is not migrated together
	Connectivity Checker	Checks connectivity and authentication for source/target databases, MongoDB, Content Server & File shares.

Planning your Migration

EMA-Migrate Considerations

Batch Segmentation	There is no technical limit to the size of a batch migrated with EMA, except for storage considerations. However, there are a couple of additional practical considerations: - Long running processes, if they fail, require a lot of time and effort to re-run. Typically we have set the size of a batch to around 1-2 million documents in most engagements. - For very large dataset, we do not want to have 100s of batches to execute. If we have 100 million documents to migrate, using 1-2 million document batches is probably near or over the limit of the number of batches that we want to manage. So we might increase the size of a batch to around 5 million objects. Batch Approaches 1. Modify Date – This approach can be used when the data being partitioned does not have versions. We can split the data using r_modify_date and provide the ranges. In the case where there are versions of documents and we use the modify date criteria the partitioned data could separate documents in a version tree into separate batches, which will cause complications during ingestion. This will be detected during extraction as there is a version tree check. But if we choose to ignore the version tree check (using --ivc option) and migrate batches with split version trees, we will need to ingest all batches in “delta” mode to ensure consistency of the version trees. This will cause the overall ingestion process to take longer, so is not preferred: the preference is to perform the main ingestion using “ingest” mode, and use the “delta” mode only for the final delta of the process. 2. Object Type – This approach is used when after the data analysis you find that the data can be partitioned using object types. Different object types can be grouped together to form a batch. • Extract all the folders first in a batch and ingest them e.g. run ExtractManager: -wh "i_cabinet_id='0cXXX' and r_object_type

	IN ('dm_cabinet','dm_folder','custom_folder1','custom_folder2') Remember to dump the IDs during Ingestion using the --dump-ids <ORIG_DB_NAME> option • For all the subsequent batches provide the object types e.g. run ExtractManager: -wh "i_cabinet_id='0cXXX' and r_object_type IN ('dm_document','custom_type1','custom_type2') Do remember to use the preload db option to load the new object ids of the folders. The preload db holds the old vs. new object_id for reprocessing the same objects respectively detecting already ingested documents. 3. Chronicle ID – similarly to using object type, chronicle ID can be used, as it will ensure that version trees stay together. 4. Other attribute – If after data analysis you find another attribute (or custom attribute) is a better way to partition the data then go ahead with it.  When we talk about the number of objects in a batch, we are typically concerned with the number of sysobject objects (e.g. documents, folders) and not concerned with additional objects such as relationships, virtual document structures, ACLs etc. If there are extreme numbers of these other objects, it could affect our thinking, but typically such objects are very small and therefore quickly ingested compared to sysobjects and subtypes.
Data to Migrate	Typically, we use EMA-Migrate to move business data only. To move configuration data, use DAR files and similar tools, such as D2-Config and xCP Designer. To this end, we usually do NOT extract data from System, Temp, Templates or Resources cabinets. In some specific cases this may make sense, but please consult with the EMA team before you confirm this as part of your approach to a migration. The most common exception to this rule is for handling deleted chronicle objects. A chronicle object is the first version created in the system (typically, but not always ,version 1.0). When a user deletes a chronicle object, it is not immediately deleted from the system, as is the case with any other version in the version tree. Instead, it is marked as with the flag is_deleted = True which causes the object not to be visible in Documentum UIs. This object is also moved to the Temp cabinet. From an EMA perspective, when we define the criteria to be extracted, we might not specify a where clause that matches such deleted objects, and if they exist, we will get an error due to a failed "version check". If this happens, consider adjusting the where clause to include delete chronicle objects.
Delta Migrations	Mostly, we can expect to migrate about 20 million objects during a weekend with EMA. In some cases we may need to stretch this time and potentially exceed the time limits imposed by the customer for a “black-out” period. Then we will need to consider running a delta. Follow these steps for running Delta

Step	Action
1	Extraction - Divide the data into batches and run the extraction for each batch. The division can be based on a Date criteria / Range of object id / Something else that you believe would divide the data into multiple batches.
2	Transformation - Executed normally without any changes.
3	Ingestion - Specify delta mode (--mode DELTA), and specify the Mongo DB used for the initial migration as a preload DB source of ID mappings (--preload-dbs <ORIG_DB_NAME>). If multiple deltas are expected, plan to either: • Dump out the IDs used for each delta to a Mongo DB that is loaded each time delta runs; • Use new DBs for each delta, and add the name to the preload DBs list. Multiple deltas should not be necessary in most cases; however this may be needed where the source system is still being actively used, during a more complex transitional period.

Ensure that the data of the initial run is not deleted, as we will need those mappings (old object ID to new object ID) again.

Storage Management

Plan how the content from the migration will be accommodated in the new system. In case we plan to use the same source storage in the target system also, we are not required to migrate the content. Create the filestores with same names as in source and change the storage pointers to the same storage location.

Two main options exist when content is to be migrated:

1. "Merge Content" – merge the content into the filestore(s) existing in the target
2. "Copy Content" – create "legacy" file stores in the target, and store migrated content there.

	Merge Content	Copy Content
PROS	Simpler content management going forward	Can copy content ahead of migration and even extraction Can re-use existing storage
CONS	Have to copy content post-migration	New content for new versions created in same "legacy" stores

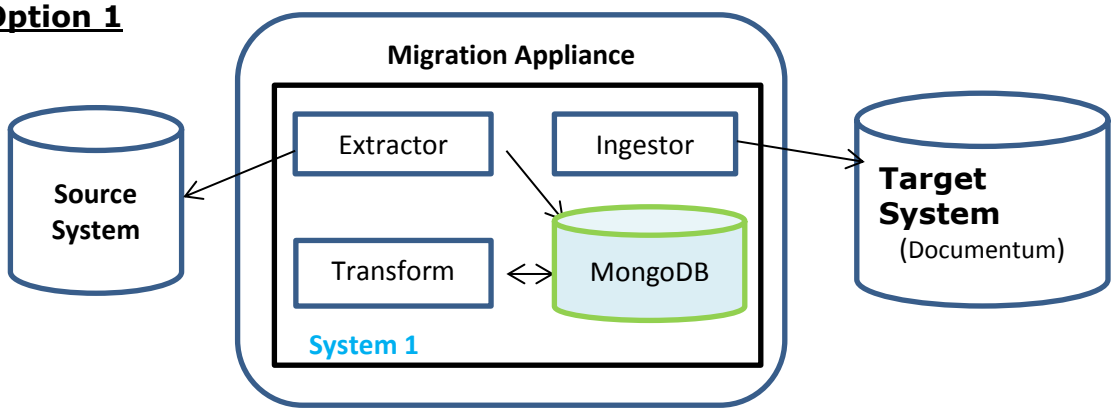
As a rule of thumb, we would typically use the "Copy" option where volume is high (say in the TBs), and the "Merge" option for smaller volumes (or where the data is coming from a 3rd party system).

We also support for extern, ca and atmos filestores added. Content files present in these filestore can now be migrated using EMA.

	Check the PPT deck (@EMA SyncP): "**KT_EMA – Migrate Content Management**" for more details.
Retention Policies	1. Retention Policies need to be exported from the Source system 2. Then Imported to the Source system 3. The System Cabinet needs to be extracted without any documents: During ingestion aspect type and the attribute needs to be provided so a proper target ID is mapped: • Get the aspect type for the particular aspect name select * from dmc_aspect_type where object_name = 'dmc_folder_markup_coordinator' Dump r_object_id get the value of (i_attr_def) • The properties file (IngerstorProperties.properties) needs to be updated with the (i_attr_def) value i.e: <dmi_0300000c800001e7> Edit/create file as: id.dmi_0300000c800001e7.repeating=markup_retainer_id • Remember to add the property file in the Ingestion cmd -Doptions.default=<file path with name>

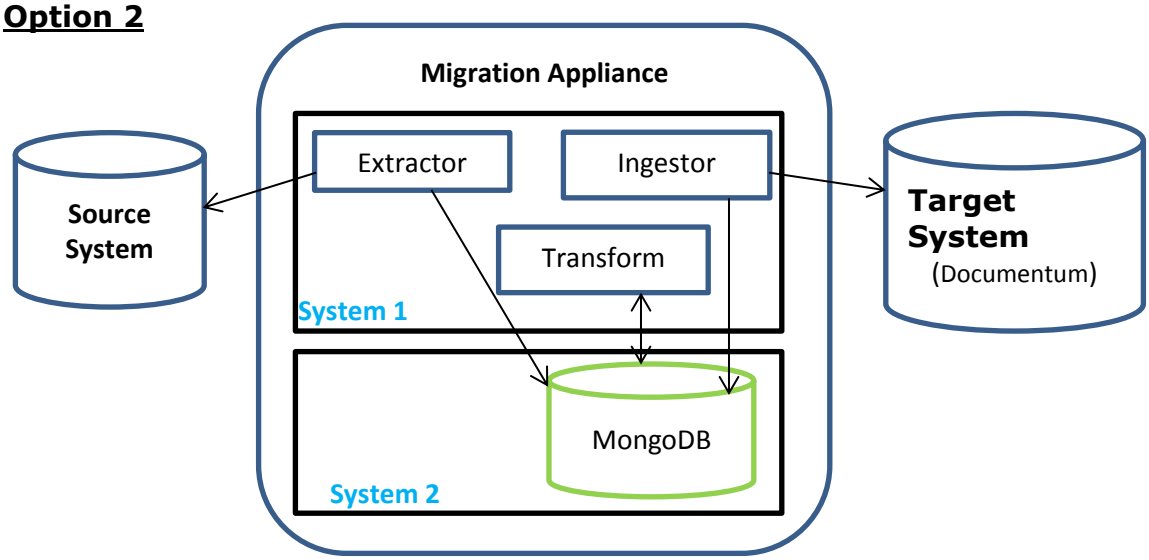
Deployment Plan out your EMA deployment depending on the amount of data you plan to migrate

Option 1



This option goes good when you have less than 1 million documents to migrate in each batch but anything more than that we would suggest going to option 2.

Option 2



When you have more than 1 million documents to migrate in each batch we would recommend going with this option where MongoDB is setup on a different system. This is done because MongoDB is memory hungry and can deprive memory to other components like Ingestor.

Migrating to a D2 Environment

If the target system to migrate is a D2 system then you can think of migrating the documents to a temporary folder and then run the OOB D2CoreJob. This would move the documents to appropriate location, apply security based on the D2 configurations.

OOB D2CoreJob can take a lot of time if the number of documents is huge. So there is an alternative standalone utility provided by Engineering which can be used for this purpose.

Contact the EMA team for more information on this.

INSTALLING EMA

Requirements

Software Requirements

- Windows Server 2012 R2 Standard 64bit server
  OR
  Windows 2008 R2 64bit server (We have implementations where consultants have used EMA in Linux environments also without any issue.)
- Java 8 SDK

Hardware Requirements

- CPU – 4
- RAM – 16 GB (If you have millions of documents being Ingested increase RAM to 32/64GB)
- Disk space – 120 GB (depends on the size of the metadata being migrated)

Other Requirements

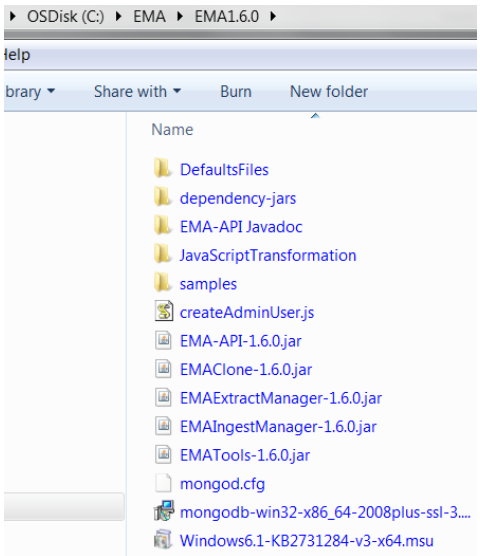
- SQL Developer / Toad / SQL Server Management studio – To check the connectivity, credentials of the source and target database as well as to do data verification.
- Robomongo (0.9.x) for MongoDB management and analysis. In case you use eclipse there is a MonjaDB plugin for eclipse that can be used.
- Database connectivity to both source and target along with the superuser credentials
- Documentum superuser authentication details for the target system (D7).
- Some familiarity with Mongo concepts and commands. Please look here.
- For additional transformations requirement not provided by EMA you might be required to write new transformations. We have transformations in both Java and JavaScript so knowledge of any of these would help.

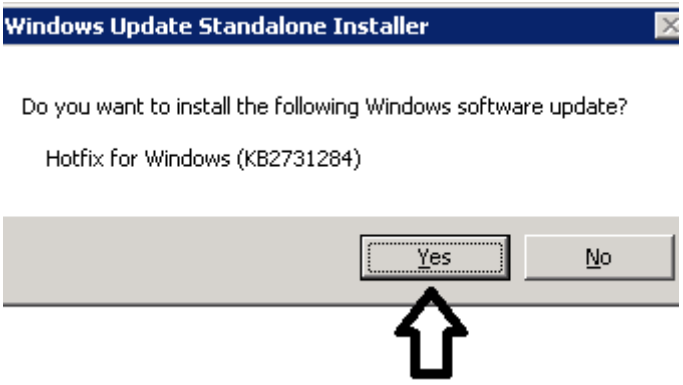
Installation Steps

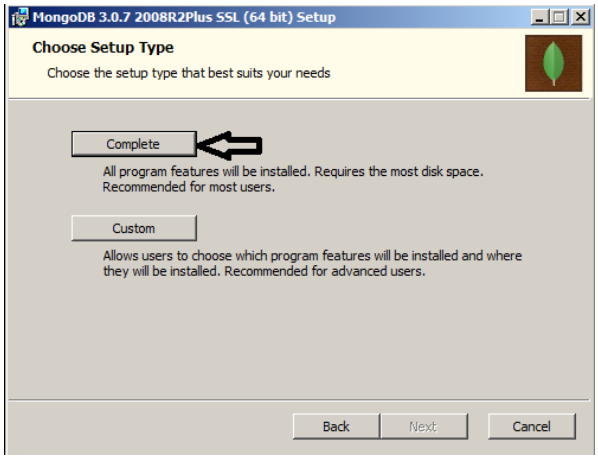
Step 1

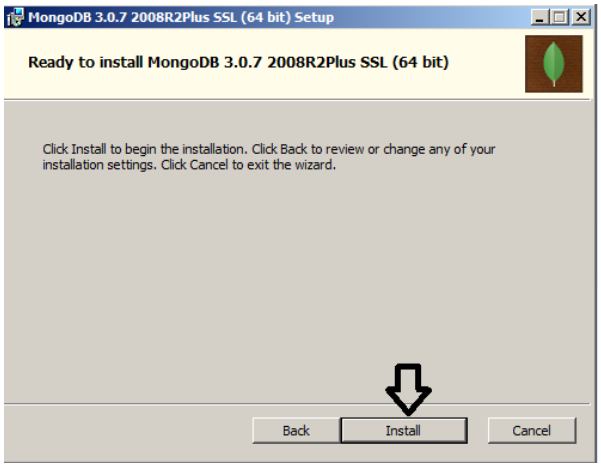
Get the latest EMA package EMA1.6.0.zip & Unzip the contents to a location e.g. C:\EMA

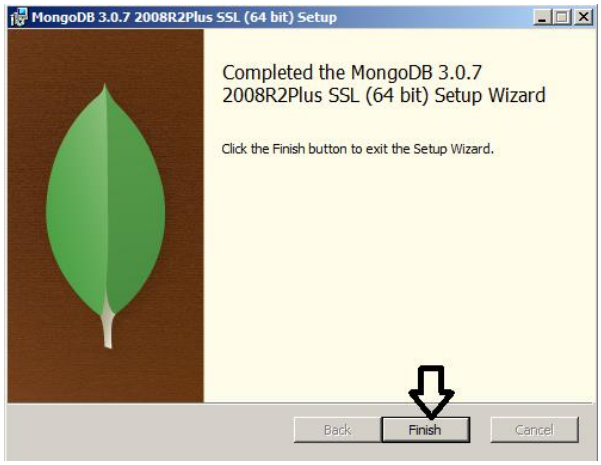
If using a different location sample scripts and files will need modifications.



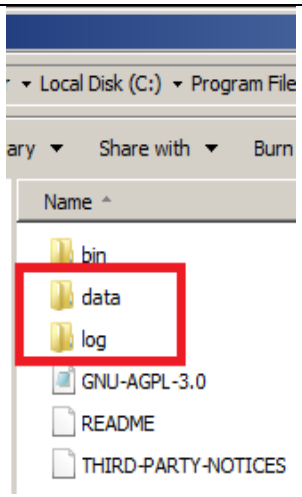
Step 2	**Install Hotfix KB2731284** (Only on Windows server 2008) – Double click on hotfix KB2731284 installer “Windows6.1-KB2731284-v3-x64.msu” file  Restart your system after the installation.
Step 3	**Install MongoDB** • If you are installing MongoDB on a separate machine copy the “**mongodb-win32-x86_64-2008plus-ssl-3.0.7-signed.msi**” file along with the “mongod.cfg” file. • Double Click on the MongoDB Installer “**mongodb-win32-x86_64-2008plus-ssl-3.0.7-signed.msi**” file •  



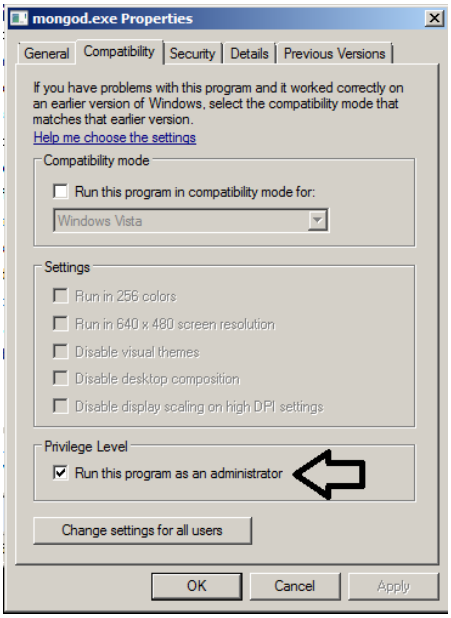




Step 4	Create folders named **log** and **data** inside "C:\Program Files\MongoDB\Server\3.0"



Step 5 **Install MongoDB as a Service**



Change the privilege of "C:\Program Files\MongoDB\Server\3.0\bin\mongod.exe" so that it can be installed as a service. For this Right click on the executable and enable the "Run this program as an administrator".

- In cmd prompt go to the Mongo installed directory "C:\Program Files\MongoDB\Server\3.0\bin" and execute the below command

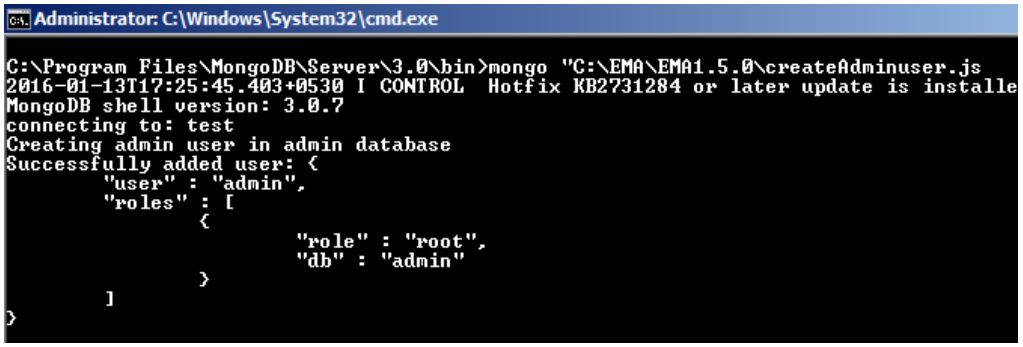
mongod.exe --auth --config C:\EMA\EMA1.6.0\mongod.cfg --install

```
C:\>cd C:\Program Files\MongoDB\Server\3.0\bin

C:\Program Files\MongoDB\Server\3.0\bin>mongod.exe --auth --config C:\EMA\EMA1.6.0\mongod.cfg --install
```

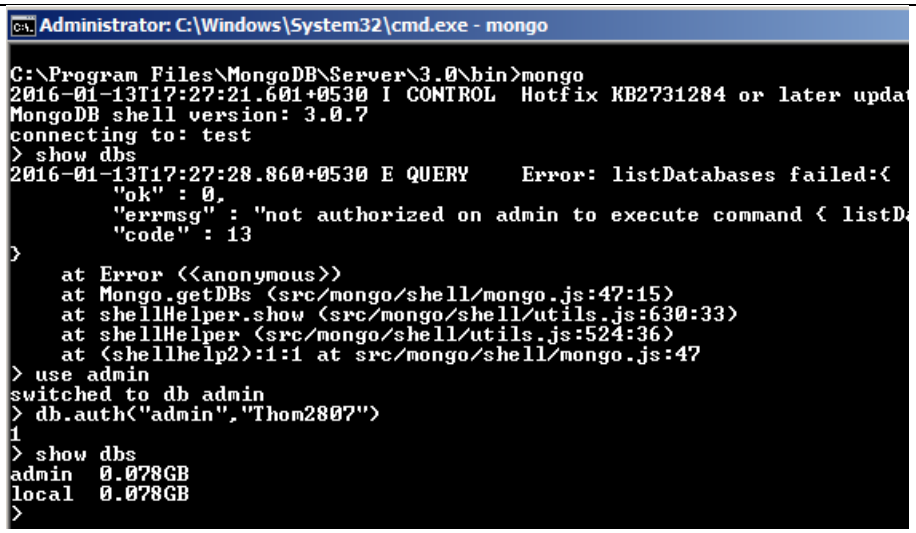
- Start the Mongo Service

Step 6	**Configure Authorization** Configure MongoDB for admin access by creating an “admin” user in the DB. You can use “**createAdminUser.js**” script for this




```
C:\Program Files\MongoDB\Server\3.0\bin>mongo
        MongoDB shell version: 3.0.7
        connecting to: test
        Welcome to the MongoDB shell. For interactive help, type "help".
        For more comprehensive documentation, see http://docs.mongodb.org/
        Questions? Try the support group
        http://groups.google.com/group/mongodb-user
> use admin
        switched to db admin
> db.createUser(
 {
  user: "admin",
  pwd: "Thom2807",
  roles:
  [
   {
    role: "root",
    db: "admin"
   }
  ]
 });
```

Step 7	Verify the Mongo Installed successfully and is running in authentication mode


Administrator: C:\Windows\System32\cmd.exe - mongo
C:\Program Files\MongoDB\Server\3.0\bin>mongo
2016-01-13T17:27:21.601+0530 I CONTROL Hotfix KB2731284 or later updat
MongoDB shell version: 3.0.7
connecting to: test
> show dbs
2016-01-13T17:27:28.860+0530 E QUERY   Error: listDatabases failed:{
        "ok" : 0,
        "errmsg" : "not authorized on admin to execute command { listDa
        "code" : 13
}
    at Error (<anonymous>)
    at Mongo.getDBs (src/mongo/shell/mongo.js:47:15)
    at shellHelper.show (src/mongo/shell/utils.js:630:33)
    at shellHelper (src/mongo/shell/utils.js:524:36)
    at (shellhelp2):1:1 at src/mongo/shell/mongo.js:47
> use admin
switched to db admin
> db.auth("admin","Thom2807")
1
> show dbs
admin 0.078GB
local 0.078GB
>

EXTRACTION

Definition

Extract data (segmented in defined batches) from the source Documentum Repository.
Sample script "*Extract.bat*" is provided in the "*samples*" directory.

Options available to define a batch:

1. **Cabinet Name**: extracts sysobjects in the cabinet and related objects*.
2. **Where Clause**: extracts sysobjects as defined by the where clause and related objects*
3. **Folder Path**: extracts sysobjects in the folder (along with sub folders) and related objects*.

*related objects – These are

1. Relation objects with the child or parent in the sysobject dataset.
2. Containment objects with the child or parent in the sysobject dataset.
3. ACL objects referenced by objects in the sysobject dataset.
4. Alias set
5. Content object
6. Assembly – Snapshots of a virtual document
7. Filestore – Filestores where the content object are present
8. Format – Format of the object
9. Policy – Lifecycle policy attached to the object.

While you technically can extract lifecycle objects (as they are sys object sub types) we do not support the ingestion of lifecycles. Configuration should be handled outside of EMA. You might see references to lifecycles, or retention policies, in an EMA extract, but they are reference objects, not the actual objects.

10. Aspects attached to the object.
11. RPS attached to the object.
12. User objects for documents with subscription.

MongoDB must be running before we can start extraction

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-sd	--source-driver	driver class	Yes	Source JDBC driver to use for connection
-sc	--source-connection	connect string	Yes	Source JDBC connection string
-su	--source-user	Username	Yes	Source JDBC username
-sp	--source-password	Password	Yes	Source JDBC password
-mh	--mongo-host	Host	Yes	Mongo DB Hostname
-mp	--mongo-port	Port	Yes	Mongo DB Port number
-md	--mongo-db	Database	Yes	Mongo DB Database name
-mu	--mongo-user	Username	Yes	Mongo DB User name
-mpa	--mongo-password	Password	Yes	Mongo DB User password
-c	--cabinet-name	Cabinet	Yes	Name of cabinet or folderpath to extract. Multiple cabinets or folderpath are seperated by pipe | operator
-wh	--where-clause	Sql clause	No	Where sql clause to extract. e.g. (i_cabinet_id = '0c0004d28000b94b' and (r_modify_date > convert(DATETIME,'2013-01-01')and r_modify_date < convert(DATETIME,'2013-10-01')))
-x	--exclude		No	Exclude objects in selected cabinets rather than include them

	-ot	--other-type	Type or where clause	No	Extract types not extracted by default. Multiple types are seperated by a pipe | operator {e.g --other-type dm_user|dm_group (r_object_id in (select r_object_id from dm_group_sp where group_name like '%docu%'))}  Only the particular type objects are extracted and not the related objects.
	-fm	--force-migrate		No	Continue extraction even in case of errors
	-ivc	--ignore-version-check		No	Skip version check (Checks if all versions of a document are present in the where clause/Cabinet specified)
	-et	--exclude-type		No	Skip extraction of specific types. Types that can be excluded are: dm_relation, dmr_content, dmr_containment, dm_assembly, dm_acl, dm_filestore, dm_format, dm_policy, rps, all Multiple types are seperated by a pipe | To exclude all default types use 'all'
	-ds	--db-schema	Schema name	No	DB schema to be added to the tables
	-h	--help		No	Show this text

Scenario	**Help**	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager* *OR* *java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager --help*
	Providing properties file for parameters  You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.	*java -Doptions.default="E:/ema/ExtractorProperties.properties" -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager* ExtractorProperties.properties `source-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver` `source-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase` `source-user=Test2` `source-password=Thom2807` `mongo-host=127.0.0.1` `mongo-port=27017` `mongo-db=ExtractorDB_10` `mongo-user=admin` `mongo-password=Thom2807` `cabinet-name=testCab`
	Extract from a cabinet (SQL Server Database)	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "testCab"*
	Extract from a cabinet (Oracle Database)	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.8.XX.XX:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "testCab"*
	Extract from a folderpath	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd*

	com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "/testCab/folder"
Where clause example  Where clause only takes SQL statement and NOT DQL.Potentially ambiguous fields should be prefixed by "s."	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -wh "(i_cabinet_id = '0c0004d28000b94b' and (r_modify_date > convert(DATETIME,'2013-01-01')and r_modify_date < convert(DATETIME,'2013-10-01')))"*
Extract from multiple cabinets	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "Resources|System|Temp|Templates|dmadmin|netvis_own"*
Exclude multiple cabinets  Interpret it as Extract Data from all cabinets not in the exclude list. Will work only with cabinets.	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "Resources|System|Temp|Templates|dmadmin|netvis_own" –x*
Extract other types not extracted by default	**Extract Users & Groups** *java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.8.58.48:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -ot "dm_user|dm_group"* **Extract Formats** *java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.8.58.48:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -ot "dm_format"*
Extract only Inline users along with the cabinet	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.8.58.48:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c " testCab " -ot "dm_user(r_object_id in (select r_object_id from dm_user_sp where user_source='inline password'))"*
Exclude types that are being extracted by default  To exclude all types extracted by default use 'all' in -et option.	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.8.58.48:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "testCab" -et "rps"*

Extract register table	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_source_docbase -su sa -sp password@123 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md regTable -ot regtable.table_name*
Extract Saved searches	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_source_docbase -su sa -sp password@123 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md savedSearch -wh "s.r_object_id IN (select r_object_id from dm_smart_list_sp where query_type='query_builder')"*
Extract Data from Oracle using db-schema	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.8.XX.XX:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "testCab" -ds "source"*
Extraction using encrypted password	*java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.extractor.ExtractManager -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.8.58.48:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa "DM_ENCR=B8yMJvZwYLKy8bEG1zZ8AQ==" -md ExtractorDB_10 -c "testCab"*
Running Extraction in single threaded mode	Modify documentum-extractor-context.xml file (present inside the jar file) and add this line
	`<bean id="taskExecutor" class="org.springframework.core.task.SyncTaskExecutor"/>` after this comment: `<!-- Documentum Extraction Job ENDS here -->` `<!--`
	This would run the Extractor in single threaded mode. Check this link for more details Customize Extractor XML file for your project

Extraction from 3rd party systems

An adaptor is required to extract data from the 3rd party systems.

If your source system is not present in the list an adaptor has to be created. Look into EMA-API for more information on it.

Combined JavaScript Approach

Process single and repeating attributes in a single pass through. This enables us to simplify processes such as moving properties from single to repeating (or vice versa), and hopefully provide migration consultants with a simpler task to create their complex transformations.

To use this approach use TransformCombo as the classname in the command line:
java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.TransformCombo -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet --script-files "C:\EMA\EMA 1.6.0\sample files\transformComboSample.js"*

Here is an example of a JavaScript transformation file using this mode:

```
var oldTypeName = "dm_document";
var newTypeName = "dm_document";

var sourceDB = { driver: "com.microsoft.sqlserver.jdbc.SQLServerDriver",
                 connection: "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase",
                 username: "sa",
                 password: "Thom2807" };

/* a DB mapper to find the name of each parent folder of the document being processed
   Note: in this case we are not using any single properties as targets, but the configuration requires one, so
         we use a "dummy" property. Object name is repeated in the SELECT columns so that one value goes to the
 dummy property, and the second one goes to parent_folder_name as a repeating property */

var parentFolderDef  = { sql: "SELECT object_name, object_name FROM dm_folder_sp WHERE r_object_id IN (SELECT
i_folder_id FROM dm_document_rp WHERE r_object_id = ?)",
                         paramlist: "r_object_id",
                        targetlist: "dummy",
                        repeatinglist: "parent_folder_name",
                        defaultlist: "NULL",
                        conn: sourceDB };

var findParentFolders;

function init() {
        findParentFolders = initExtendedDBLookupWithCacheMapper(parentFolderDef);
}

function transformCombo() {
        /* output message to the log using logger reference */
        logger.info("Processing object: " + getString("r_object_id"));

        /* set the title of the document to indicate transformation by EMA */
        setString("title", "Transformed by EMA");

        /* remove existing author entries and add the following three names */
        truncate("authors");
        appendString("authors", "Ueli Wehrli");
        appendString("authors", "Chris Dyde");
        appendString("authors", "Ingo Zitzmann");

        /* add the following text in addition to the existing keywords */
        appendString("keywords", "Added by EMA-Transform");

        /* move object_name property to a (new) repeating attribute */
        appendString("all_object_names", getString("object_name"));

        /* create a new property containing comma separated list of version labels */
```

```
        var all_labels = "";
        var labelList = getValues("r_version_label"); /* returns ArrayList<String> to labelList */

        for (var i = 0; i < labelList.size(); i++) {
                if (i != 0) {
                        all_labels += ", ";
                }
                all_labels += labelList.get(i);
        }
        setString("all_labels", all_labels);

        /* find parent folder names */
        mapCombo(findParentFolders);
}
```

Main Differences for the JavaScript coder:

1. New name for the function to transform the data: transformCombo. Has no parameter indicating single/repeating, as they are processed at the same time.
2. setString etc. used for single properties exactly as they were before.
3. Functions (see transformLibrary.js) support append/truncate/remove functions on repeating attributes.
4. DB Mappers work as before, with the same parameters, but use mapCombo(<mapper variable>) to execute the mapping function.

There are two additional advantages to using combo mode:

1. It will perform quicker than the old approach if multiple single/repeating transformations are needed and/or if repeating fields will be manipulated (e.g. changing/removing version labels, moving property values into a repeating property table), because the initial manipulation is in memory, and we write the final values with a simple append (rather than an update) into mongo.
2. There is no need to take a "backup" of the repeating collection prior to transformation, which can be a time-saver for larger migrations.

Java Approach

While JavaScript provides a much easier environment for the developer to work in for more complex transformations, it does have limitations in terms of: debugging, leveraging third party components. To end this, we have added the option to use Java code itself to create transformations for EMA.

When the developer is using Java code, they can use the power of the IDE they use to debug issues, and have full access to all third party libraries available in Java.

To use Java specify the transformation classname in the command line:

java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.TransformCombo -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet* ***--java-transform com.emc.monorail.tools.transform.javasample.SampleJavaTypeTransform -ct --java-param "dm_document,dm_document"***

The parameters are passed into the initialization method of the class, and can provide information to allow the code to be more generic (instead of creating a Java class for each and every From-To combination). The transform class must extend the class com.emc.monorail.tools.transform.JavaTypeTransform and implement (override) the following methods:

```
public String getOldTypeName();
// returns the old type name(s) to be transformed. Use ",” to separate multiple type
// names
```

```
public String getNewTypeName();
// returns the new type name(s) to be transformed into. Use “,” to separate multiple type
// names

public boolean applyTypeTransform(DB db, DBObject dbo, boolean bRepeating) throws
TransformException;
// apply transformation to the current row in the non-Combo mode of transformation

public boolean applyComboTransform(DB db, DBObject dbo, RepeatingAttributeSet rptg,
boolean bRepeating) throws TransformException;
// apply transformation to the current object in the Combo mode of transformation

public boolean hasNameChange();
// return true if the type is changing, otherwise false

public boolean hasMultipleTargets();
// return true if there are multiple values for new type name

public void init(Mongo m, String mongoDB, String javaParam);
// called to initialize the transformation class. If no additional initialization is
// required, the base class provides this method

public void deInit();
// called to release resources, output summary information at the end of transformation
```

An example of a Java transform is shown here (sample project “TransformJavaSample” provided in samples). The transformation actions are the same as those shown in the JavaScript example, above:

```
package com.emc.monorail.tools.transform.javasample;

import java.util.ArrayList;

import org.apache.log4j.Logger;

import com.emc.monorail.tools.transform.ConditionalMultiDBLookupWithCacheMapper;
import com.emc.monorail.tools.transform.JavaTypeTransform;
import com.emc.monorail.tools.transform.TransformException;
import com.emc.monorail.tools.transform.ValueMapperException;
import com.mongodb.DB;
import com.mongodb.DBObject;
import com.mongodb.Mongo;

public class SampleJavaTypeTransform extends JavaTypeTransform {
        Logger logger = Logger.getLogger(this.getClass());
        String oldType = "undefined";
        String newType = "undefined";
        ConditionalMultiDBLookupWithCacheMapper findParentFolders;

        @Override
        public String getOldTypeName() {
                return oldType;
        }

        @Override
        public String getNewTypeName() {
                return newType;
        }

        @Override
        public boolean applyTypeTransform(DB db, DBObject dbo, boolean bRepeating) throws TransformException {
                throw new TransformException("Only combo transform mode is supported by " +
this.getClass().getName());
```

```
	}

	@Override
	public boolean applyComboTransform() {
		boolean retcode = true;

		/* output message to the log */
		logger.info("Processing object: " + getString("r_object_id"));

		/* set the title of the document to indicate transformation by EMA */
		setString("title", "Transformed by EMA");

		/* remove existing author entries and add the following three names */
		truncate("authors");
		appendString("authors", "Ueli Wehrli");
		appendString("authors", "Chris Dyde");
		appendString("authors", "Ingo Zitzmann");

		/* add the following text in addition to the existing keywords */
		appendString("keywords", "Added by EMA-Transform");

		/* move object_name property to a (new) repeating attribute */
		appendString("all_object_names", getString("object_name"));

		/* create a new property containing comma separated list of version labels */
		StringBuffer all_labels = new StringBuffer();
		ArrayList<String> labelList = getValues("r_version_label"); /* returns ArrayList<String> to
labelList */

		for (int i = 0; i < labelList.size(); i++) {
			if (i != 0) {
				all_labels.append(", ");
			}
			all_labels.append(labelList.get(i));
		}
		setString("all_labels", all_labels.toString());

		/* find parent folder names */
		try {
			mapCombo(findParentFolders);
		} catch (ValueMapperException vme) {
			logger.error("Error mapping DB values", vme);
			retcode = false;
		}
		return retcode;
	}

	@Override
	public boolean hasNameChange() {
		return !oldType.equals(newType); // for this sample, object types are not modified
	}

	@Override
	public boolean hasMultipleTargets() {
		return false;
	}

	@Override
	public void deInit() {
		logger.info("All done!");
	}

	@Override
	public void init(Mongo m, String mongoDB, String javaParam) {
		String[] types = javaParam.split(",");
		oldType = types[0];
		newType = types[1];
```

```
                // initialize the DB mapper
                findParentFolders = new ConditionalMultiDBLookupWithCacheMapper();

                try {
                        findParentFolders.init("com.microsoft.sqlserver.jdbc.SQLServerDriver",
                                               "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test20_docbase",
                                               "sa",
                                               "Thom2807",
                                               oldType,
                                               "SELECT object_name, object_name FROM dm_folder_sp WHERE
r_object_id IN (SELECT i_folder_id FROM dm_document_rp WHERE r_object_id = ?)",
                                               "r_object_id",
                                               "dummy",
                                               "parent_folder_names",
                                               "NULL");
                }
                catch (ValueMapperException vme) {
                        logger.error("Error initializing DB mapper", vme);
                }
        }
}
```

Typically, the Java transform will support only one mode of operation, whichever is most suited to the complexity of the use case. As we would expect a complex use case to decide on using a Java transform, we would recommend the use of the TransformCombo model in most cases.

Mapper objects can be used as in JavaScript, but the developer may find it easier to create their own code to process database lookups etc. and has total flexibility of java code to use whatever is at their disposal.

The class JavaTransform should send output to Log4j to provide a central location for log info that can be fine tuned using log4j.properties. The code will need to instantiate it's own Logger object during class instantiation or in the init() method.

Properties File Approach (Deprecated)

This section defines the transform properties files approach that can be used for transformation. A further section describes the creation of a custom transform plug-in with a worked example. Sample scripts "Transform-properties(Deprecated).bat"

Terminology	Definition
Legacy type	the name of the existing type, assumed to be subject to a change as part of the transformation
Existing type	a type that exists in the source or target or both
Day-forward type	a type that exists in the target only, assumed to be the new type for one of more legacy types

Parameters	Short Option	Long Option	Argument	Mandatory	Description
	-mh	--mongo-host	hostname	Yes	Mongo DB Hostname
	-mp	--mongo-port	port number	Yes	Mongo DB Port number
	-md	--mongo-db	db name	Yes	Mongo DB Database name
	-mu	--mongo-username	username	Yes	Mongo DB Username
	-mpw	--mongo-password	password	Yes	Mongo DB Password
	-bs	--batch-size	batch size	No	Size of batch to write into Mongo (if absent, individual

					rows are written)
	-tf	--transform-files	file1, file2..	No	List of properties file defining property transformations
	-sf	--script-files	file1, file2..	No	List of javascript file defining property transformations
	-sl	--script-libraries	lib1, lib2…	No	List of javascript library
	-tc	--thread-count	count	No	Number of threads to use for transform
	-ct	--clear-target		No	Clear collections with previous transformation run data
	-h	--help		No	Show this text

Scenario	**Help**	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.Transform*
	Providing properties file for parameters  Options can be added to a properties file instead of providing them on the command line. In case an option is provided both in properties file as well as in command line, the command line option value will override the value provided in the properties file.	*java -Doptions.default="E:/ema/TransformProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.Transform* TransformProperties.properties `mongo-host=127.0.0.1` `mongo-port=27017` `mongo-db=ExtractorDB_10` `mongo-user=admin` `mongo-password=Thom2807` `transform-files=C:\\EMA1.6.0\\samplefiles\\transform.properties`
	Transform sample	*java -cp "E:\ema\EMATools-1.6.0.jar;E:\ema\EMA-API-1.6.0.jar;E:\ema\EMAIngestManager-1.6.0.jar; ;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.Transform -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet --transform-files "C:\EMA\EMA 1.6.0\sample files\transformSample.properties"*

Java Properties File Approach

General

A sample properties file "*transformJavaSample.properties*" is provided in the "*samples*" directory. The following properties are mandatory for all transform properties files:

```
Newname=<new object type name>
Oldname=<old object type name>
Propcount=<number of properties that have treansforms to be applied>
Mode=<single, repeating or both>
```

1) Newname and oldname are both required fields, even if there is no change to the object type required in the transformation
2) Newname and oldname must be the Documentum object type names as defined in the type_name field of the dm_type object
3) "mode" is used when the property in question is new and does not exist in the source object model, and therefore does not exist in the mongo collection. "single" or "both" are applied to single property rows, "repeating" or "both" are applied to repeating property rows

One file is needed for each and every oldname+newname combination. For example, if we are mapping several legacy types to a common day-forward type, then we will need a transform properties file for each of these transitions, and the EMA-Transform tool will receive a list of transformation properties files to be applied to the mongo data.

In the reverse example, where a single legacy type will map to more than one day-forward type, properties must exist in the legacy object that will allow us to create an SQL query identifying which set of documents transition to which day-forward type.

Transforming to Multiple Object Types

The transform tool supports transformation from a single input type to one of many output types. For example, if we have a source document type of "my_source_doc" and three potential target document types of "my_target_doc_1", "my_target_doc_2" and "my_target_doc_3" we can support that. This gives us the flexibility to extract a document type together in a single batch, rather than processing each target type as its own batch. If using this option, the transformation script must set the r_object_type property when processing the single rows. If no new value is set for r_object_type by the transformation script, the object type of the document will not be changed.

In order to configure this option, the following steps are required:

Use the class TransformEx in your command line.
Define old type and define new type as a list:

```
oldname=my_source_doc
newname=my_target_doc_1,my_target_doc_2,my_target_doc_3
```

Transforming to Multiple Object Types on Condition

In the single property processing, you will need to update the r_object_type within your transformation. Here is an example using a combination of IfThenElseMappers to set the new type based on a property value:

```
prop.X.oldname=r_object_type
prop.X.newname=r_object_type
prop.X.mode=single
prop.X.mapperclass=com.emc.monorail.tools.transform.IfThenEls
eMapper
prop.X.testpropname=future_object_type
prop.X.testpropvalue=1
prop.X.true.value=my_target_doc_1
prop.X.false.mapperclass=
com.emc.monorail.tools.transform.IfThenElseMapper
prop.X.false.testpropname=future_object_type
prop.X.false.testpropvalue=2
prop.X.false.true.value=my_target_doc_2
prop.X.false.false.value=my_target_doc_3
```

Any kind of mapper can be used to set the value of r_object_type, for example a DBLookup, or of course a custom mapper for the customer situation.

Individual Property Transforms

Each property change is defined in a property transform section in the transform properties file. The parameters of this section will vary depending on the transform to be applied, but there are some common properties (see example below). The settings for each property mapping are prefixed with an index (0-based), for example:

```
prop.X.oldname=<old property name>
prop.X.newname=<new property name>
prop.X.mapperclass=<class of mapper to use, see available list>
```

Where "X" is replaced by the running index, starting from 0, of the property in question. For the special case that we are simply changing the property name, this will suffice. If a property is not changing, no entry at all is required in the transform properties file.

Transforms provided in EMA JAR file

The following transform plug-ins are provided in the EMA JAR file and can be used for both Transform and Morph scenarios.

Mapper	Description	Configuration
AdjustVersionLabelMapper	**com.emc.monorail.tools.transform.AdjustVersionLabelMapper** Used mostly for DCM conversion projects, and intended to remove symbolic version labels used by DCM that are no longer used in the D2 application that replaces DCM. To remove a specific value from the version labels, use type=value, with the value in remove. To remove the value of a specific attribute, use type=attribute, and specify the attribute name in remove. Specify the outgoing property name in the target field.	prop.X.oldname=<use a dummy value here> prop.X.newname=<use a dummy value here> prop.X.mode=SINGLE prop.X.mapperclass=com.emc.monorail.tools.transform.AdjustVersionLabelMapper prop.X.type=attribute OR value prop.X.source=<repeating attribute to read from> prop.X.target=<attribute to write to, can be the same as source> prop.X.remove=<value to remove, or name of attribute who's value should be removed>
	Example: Remove the symbolic version label stored in a_status from the list of values of r_version_label:	prop.X.oldname=dummy prop.X.newname=dummy prop.X.mode=**single** prop.X.mapperclass=com.emc.monorail.tools.transform.AdjustVersionLabelMapper prop.X.type=**attribute** prop.X.source=r_version_label prop.X.target=r_version_label prop.X.remove=a_status
Cabinet RenameFolderPathMapper	**com.emc.monorail.tools.transform.CabinetRenameFolderMapper** In consolidation projects, it may be necessary to change the cabinet structure of a repository. For example, in an architecture where each line of business has a folder "PROJECTS", and the consolidated structure is "PROJECT – LOBX", we need to modify not just the object name of the cabinet, but also the corresponding r_folder_path entries in the folders and cabinets associated. This mapper takes care of this change. Oldname/newname	prop.X.oldname=r_folder_path prop.X.newname=r_folder_path prop.X.mapperclass=com.emc.monorail.tools.transform.CabinetRenameFolderPathMapper prop.X.cabinet.oldvalue=PROJECTS prop.X.cabinet.newvalue=Marketing Projects  Combo patterns use ${<field name>} format to substitute in property variables (including the property itself, if needed).

	in this case should ALWAYS be r_folder_path. The object_name of the cabinet itself is modified by a ValueListMapper.	
	Example:	prop.X.oldname=r_folder_path prop.X.newname=r_folder_path prop.X.mapperclass=com.emc.monorail.tools.transform.CabinetRenameFolderPathMapper prop.X.cabinet.oldvalue=PROJECTS prop.X.cabinet.newvalue=Marketing Projects
ComboMapper	**com.emc.monorail.tools.transform.ComboMapper** This mapper allows us to build a property value based on constant elements and variable elements drawn from the same database row. This can be used to simulate D2 auto-naming configurations.	prop.X.oldname=<property source not used but needed!> prop.X.newname=<property to write the value into> prop.X.mapperclass=com.emc.monorail.tools.transform.ComboMapper prop.X.cabinet.pattern=<combo pattern>
	Example:	prop.X.oldname=object_name prop.X.newname=object_name prop.X.mapperclass=com.emc.monorail.tools.transform.ComboMapper prop.X.cabinet.pattern=DOC-${document_id} ${document_type}

ConditionalMultiDBLookupWithCacheMapper	**com.emc.monorail.tools.transform.ConditionalMultiDBLookupWithCacheMapper** Used to chain DB lookups together – if the first produces no results, the second query is executed and so on. Also includes functionality to write multiple returned rows into repeating attributes of an object (see below).	prop.X.oldname=<use a dummy value here> prop.X.newname=<use a dummy value here> prop.X.mode=single prop.X.mapperclass=com.emc.monorail.tools.transform.ConditionalMultiDBLookupWithCacheMapper prop.X.connectionInfo=<see DBLookupMapper> prop.X.statementcount=<number of statements in the mapper> prop.X.statement.Y.sql=<statement to execute> prop.X.statement.Y.paramlist=<list of parameters for the statement> prop.X.targetlist=<list of single value result columns> prop.X.repeatinglist=<list of repeating value result columns> prop.X.defaultlist=<default value list, used for single value columns only>
	Example: For a DCM/D2LSQM conversion, try to find the matching product code and generic name from the registration forms for a given legacy name. If the first query fails, try a different source field.	prop.X.oldname=dummy prop.X.newname=dummy prop.X.mode=single prop.X.mapperclass=com.emc.monorail.tools.transform.ConditionalMultiDBLookupWithCacheMapper prop.X.connectionInfo=<see DBLookupMapper> prop.X.statementcount=2 prop.X.statement.0.sql=SELECT s.product_code, r.product_generic_name FROM cd_product_info_sp s, cd_product_info_rp r WHERE s.r_object_id = r.r_object_id AND s.product_name = ? prop.X.statement.0.paramlist=our_product_name prop.X.statement.1.sql=SELECT s.product_code, r.product_generic_name FROM cd_product_info_sp s, cd_product_info_rp r WHERE s.r_object_id = r.r_object_id AND s.product_code = ? prop.X.statement.1.paramlist=our_product_code prop.X.targetlist=d2_product_code prop.X.repeatinglist=d2_generic_name prop.X.defaultlist=NOTFOUND!!
ConstantValueMapper	**com.emc.monorail.tools.transform.ConstantValueMapper** Used to overwrite/insert values with a constant value. To insert default values into unoccupied properties, we can use defaults files during ingestion. But if the property already has a value, the default is ignored. Therefore, we have implemented a mapper to overwrite the existing field with a new, constant value.	prop.X.oldname=<property source not used but needed!> prop.X.newname=<property to write the value into> prop.X.mapperclass=com.emc.monorail.tools.transform.ConstantValueMapper prop.X.constantvalue=<value to write into the property>

	Example:	prop.X.oldname=subject prop.X.newname=subject prop.X.mapperclass=com.emc.monorail.tools.transform.ConstantValueMapper prop.X.constantvalue=Migrated by EMA!
CopyProperty ListMapper	**com.emc.monorail.tools.transform. CopyListPropertyMapper** A common requirement is to copy certain fields into "legacy" property fields in the day-forward object type. For modification dates, this is not strictly necessary, but for object ID it can have some utility for running deltas with Trinity Bridge or other tools, or for third-party integrations. Copying single fields can be done by simply renaming the field (as old values are not removed) but if you have multiple properties to copy, we would recommend using this transform. In this case, oldname/newname are effectively dummy and will be unaffected by the transform.	prop.X.oldname=<property source not used but needed!> prop.X.newname=<same as oldname> prop.X.mapperclass=com.emc.monorail.tools.transform.CopyPropertyListMapper prop.X.propcount=<number of properties to copy> prop.X.name.Y.source=<source of property value> prop.X.name.Y.target=<target of property value>  Y starts with 0.
	Example:	prop.X.oldname=object_name prop.X.newname=object_name prop.X.mapperclass=com.emc.monorail.tools.transform.CopyPropertyListMapper prop.X.propcount=3 prop.X.name.0.source=r_object_id prop.X.name.0.target=legacy_r_object_id prop.X.name.1.source=r_modified prop.X.name.1.target=legacy_modifier prop.X.name.2.source=r_modify_date prop.X.name.2.target=legacy_r_modify_date
Date Adjustment Mapper	**com.emc.monorail.tools.transform. DateAdjustmentMapper** Used when extracting data from older Oracle/Documentum versions, because up until version 5.3 (++TODO - check this version #++) date/time values were stored in local server timezone, not UTC. Now all dates are stored in UTC form. To compensate for the difference between local/server time and UTC, we need to add/subtract a number of hours to ALL date VALUES found in the source collection, date being defined as matching the EMA date format: yyyy/MM/dd HH24:MI:SS. **Note:** not supported for JavaScript transformations.	prop.X.oldname=<property source not used but needed!> prop.X.newname=<same as oldname> prop.X.mapperclass=com.emc.monorail.tools.transform.DateAdjustmentMapper prop.X.dateOffset=<number of units to add, use negative number to subtract> prop.X.dateNull=<date to sub in if a current value is invalid. OPTIONAL, if omitted, "null" is subbed in>

	Example:	`#note: applies to both single and repeating!` `prop.X.oldname=r_object_id` `prop.X.newname=r_object_id` `prop.X.mapperclass=com.emc.monorail.tools.transform.DateAdjustmentMapper` `# local time is Central US Time` `prop.X.dateOffset=6` `# default suitable for ingestion into SQL Server` `prop.X.dateNull=1753/01/01 00:00:00`

DateReformatMapper	**com.emc.monorail.tools.transform.DateReformatMapper** Used to reformat a date value from source to target, would typically be used if dates were stored in non-Date types in the source, or if the source was a third party system, in order to get the date value into the EMA format. Alternatively, can be used to create a string value from a date value. **Note:** not supported for JavaScript transformations.	`prop.X.oldname=<field to reformat>` `prop.X.newname=<field to write out to>` `prop.X.mapperclass=com.emc.monorail.tools.transform.DateReformatMapper` `prop.X.inputformat=<format of data going in>` `prop.X.outputformat=<format of data going out>`
	Example:	`prop.X.oldname=contract_date_string` `prop.X.newname=contract_date_timestamp` `prop.X.mapperclass=com.emc.monorail.tools.transform.DateReformatMapper` `prop.X.inputformat=MM/dd/yyyy` `prop.X.outputformat=yyyy/MM/dd HH24:MM:SS`
DBLookup Mapper	**com.emc.monorail.tools.transform.DBLookupMapper** This mapper is used to look up data in a database, using a parameterized query, plugging in property values to the SELECT statement for each row processed. The SELECT statement is issued once per row in the source collection – for a dataset << number of rows to process, we recommend using the caching variant	`prop.X.oldname=cutomer_code` `prop.X.newname=customer_code` `prop.X.mapperclass=com.emc.monorail.tools.transform.DBLookupMapper` `prop.X.connectInfo=sourceRepo` `prop.X.sql=SELECT city_gov, county_gov, state_gov, federal_gov FROM customer_db WHERE customer_code=?` `prop.X.resultmapmode=concat` `prop.X.resultmap.0.value.default=NOT FOUND!!` `prop.X.resultmap.0.valuecount=1` `prop.X.resultmap.0.value.0.oldvalue=YNNN` `prop.X.resultmap.0.value.0.newvalue=CITY`

	 The first row returned by the JDBC connection is used to provide the value.	prop.X.oldname=<parameter field> prop.X.newname=<field to write out to> prop.X.mapperclass=com.emc.monorail.tools.transform.DBLookupMapper prop.X.connectionInfo=<name of connection info block to use> prop.X.sql=<SELECT statement. Use "?"s as placeholders for parameters> prop.X.paramlist=<list of parameter fields. If absent, single parameter defined in "oldname" is passed to the SELECT> prop.X.defaultValue=<default value if no match is found in the SELECT> # connection info is defined separately, so it can be referenced # by multiple mappings db.<connection info name>.driver=<JDBC driver class> db.<connection info name>.connection=<JDBC connection string> db.<connection info name>.username=<JDBC username> db.<connection info name>.password=<JDBC password>
	Example:	prop.X.oldname=customer_name prop.X.newname=customer_name prop.X.mapperclass=com.emc.monorail.tools.transform.DBLookupMapper prop.X.connectionInfo=sourceRepo prop.X.sql=SELECT customer_name FROM customer_db WHERE customer_code = ? AND business_unit = ? prop.X.paramlist=cust_code,cust_bu prop.X.defaultValue=NOT FOUND!! db.sourceRepo.driver=oracle.jdbc.OracleDriver db.sourceRepo.connection=jdbc:oracle:thin:@localhost:1521:ORCL db.sourceRepo.username=dbowner db.sourceRepo.password=top_secret_password
EditRepeatingAttribute Mapper	**com.emc.monorail.tools.transform.EditRepeatingAttributeMapper** Used mostly for DCM conversion projects, or for FirstDocs etc. in order to remove symbolic version labels that are no longer in use in the new applications. It can also be used to move the CURRENT label to the document carrying the Most-Recent symbolic label.	prop.X.oldname=<use a dummy value> prop.X.newname=<use a dummy value> prop.X.mode=single prop.X.mapperclass=com.emc.monorail.tools.transform.EditRepeatingAttributeMapper prop.X.source=<source attribute> prop.X.target=<target attribute> prop.X.removelist=<list of values to remove, separated by ","> prop.X.maplist=<list of values to map to new values, separated by ","> prop.X.mapvaluelist=<list of new values, separated by ","> prop.X.sethasfolder=<update the i_has_folder property, if the list we write out contains the "CURRENT" value>

	Example: DCM Example, remove common DCM labels, move CURRENT to the document with Most-Recent label.	prop.X.oldname=dummy prop.X.newname=dummy prop.X.mode=single prop.X.mapperclass=com.emc.monorail.tools.transform.EditRepeatingAttributeMapper prop.X.source=r_version_label prop.X.target=r_version_label prop.X.removelist=CURRENT,Approved,Draft,Effective,Obsolete,Superseded prop.X.maplist=Most-Recent prop.X.mapvaluelist=CURRENT prop.X.sethasfolder=true
IfThenElse Mapper	**com.emc.monorail.tools.transform.IfThenElseMapper** Used to create conditional logic in the linear structure of the properties file. If the logic is more complex, a custom mapper class is probably more appropriate!	prop.X.oldname=<not used but must be defined> prop.X.newname=<attribute set by the mapper> prop.X.mode=<single/repeating/both> prop.X.mapperclass=com.emc.monorail.tools.transform.IfThenElseMapper prop.X.testpropname=<attribute to test> prop.X.testpropvalue=<value to test against> prop.X.true.value=<value to assign if the prop check is true> prop.X.true.mapperclass=<if value is not set, use this mapper class> prop.X.true.YYYY=<define true mapper parameters with this prefix> prop.X.false.value=<OPTIONAL: value to assign if prop check is false> prop.X.false.mapperclass=<if value is not set, use this mapper class> prop.X.false.ZZZZ=<define false mapper parameters with this prefix>

	Example: Check the value of a property, if it is not set to NOTFOUND!! Use a DBLookup  To Compare values of two attributes use: prop.X.true.value=$new_prop_name	prop.X.oldname=customer_name prop.X.newname=customer_name prop.X.mode=single prop.X.mapperclass=com.emc.monorail.tools.transform.IfThenElseMapper prop.X.testpropname=customer_code prop.X.testpropvalue=NOTFOUND!! prop.X.true.value=NOCUSTCODE!! prop.X.false.mapperclass=com.emc.monorail.tools.transform.DBLookupMapper prop.X.false.connectionInfo=sourceRepo prop.X.false.sql=SELECT customer_name FROM customer_db WHERE customer_code = ? prop.X.false.paramlist=customer_code prop.X.false.defaultValue=NOTFOUND!!
MergeFromDBMapper	**com.emc.monorail.tools.transform.MergeFromDBMapper** Used to merge the results of a query into an existing repeating attribute. This can be used multiple times to merge properties together and is used mostly to map data from fields that are being dropped into a common field in order not to lose the information contained. Note: although this field ultimately affects Repeating properties, it should be configured to run in the single properties collection so that it only executes once.  for connection info details and example, see DBLookupMapper	prop.X.oldname=<not used but must be present!> prop.X.newname=<same as oldname> prop.X.mapperclass=com.emc.monorail.tools.transform.MergeFromDBMapper prop.X.connectionInfo=<name of connection info block to use> prop.X.sql=<SELECT statement. Use "?"s as placeholders for parameters> prop.X.paramlist=<list of parameter fields. If absent, single parameter defined in "oldname" is passed to the SELECT> prop.X.defaultValue=<default value if no match is found in the SELECT> prop.X.target=<target field for the value>
	Example: Appends values of a custom_keyword field to the values already present for the existing keywords field.	prop.X.oldname=<not used but must be present!> prop.X.newname=<same as oldname> prop.X.mapperclass=com.emc.monorail.tools.transform.MergeFromDBMapper prop.X.connectionInfo=sourceRepo prop.X.sql=SELECT custom_keyword FROM my_type_rp WHERE r_object_id = ? prop.X.paramlist=r_object_id prop.X.target=keywords
MoveTo Cabinet Mapper	**com.emc.monorail.tools.transform.MoveToCabinetMapper** Move the sysobject to the specified cabinet. Often used in conjunction with MoveToFolderMapper. Note: this applies to a constant cabinet name, for variable cabinet name use some form of DB lookup. Additionally, the connection info should point to the target repository, as we will be looking for the target cabinet ID.	prop.X.oldname=i_cabinet_id prop.X.newname=i_cabinet_id prop.X.mapperclass=com.emc.monorail.tools.transform.MoveToCabinetMapper prop.X.connectionInfo=<see DBLookupMapper> prop.X.cabinetname=<name of cabinet to move objects to>

	Example: Move all documents to a “Migration” cabinet during migration	prop.X.oldname=i_cabinet_id prop.X.newname=i_cabinet_id prop.X.mapperclass=com.emc.monorail.tools.transform.MoveToCabinetMapper prop.X.connectionInfo=<see DBLookupMapper> prop.X.cabinetname=Migration
MoveToFolderMapper	**com.emc.monorail.tools.transform.MoveToFolderMapper** Move the sysobject to the specified folder. Often used in conjunction with MoveToCabinetMapper. Note: this applies to a constant folder name, for variable folder name use some form of DB lookup. Additionally, the connection info should point to the target repository, as we will be looking for the target folder ID. The value of i_folder_id[0] is set to the folder ID that we find in the target repository.	prop.X.oldname=i_folder_id prop.X.newname=i_folder_id prop.X.mapperclass=com.emc.monorail.tools.transform.MoveToFolderMapper prop.X.connectionInfo=<see DBLookupMapper> prop.X.foldername=<name of folder to move objects to>
	Example: Move all documents to a folder called “/Migration/Documents” during migration	prop.X.oldname=i_folder_id prop.X.newname=i_folder_id prop.X.mapperclass=com.emc.monorail.tools.transform.MoveToFolderMapper prop.X.connectionInfo=<see DBLookupMapper> prop.X.foldername=/Migration/Documents
MultiDBLookupMapper	**com.emc.monorail.tools.transform.MultiDBLookupMapper** Similar to DBLookupMapper, but allowing the same mapping to populate multiple fields. for connection info details and example, see DBLookupMapper.	prop.X.oldname=<parameter field> prop.X.newname=<field to write out to> prop.X.mapperclass=com.emc.monorail.tools.transform.MultiDBLookupMapper prop.X.connectionInfo=<name of connection info block to use> prop.X.sql=<SELECT statement. Use “?”s as placeholders for parameters> prop.X.paramlist=<list of parameter fields. If absent, single parameter defined in “oldname” is passed to the SELECT> prop.X.targetlist=<comma separated list of properties to populate, order is as in the SELECT statement> prop.X.defaultValuelist=<comma separated list of default values> # optional result mapping: prop.X.resultmapmode=<OPTIONAL: if = concat, the columns of the SELECT result are concatenated and used as a map to find a value from the list below> prop.X.resultmap.0.valuecount=<number of entries in the map> prop.X.resultmap.0.value.Y.oldvalue=<result of SQL statement> prop.X.resultmap.0.value.Y.newvalue=<value to insert into the field>

	Example: Simple look up of multiple fields.	prop.X.oldname=customer_code prop.X.newname=customer_code prop.X.mapperclass=com.emc.monorail.tools.transform.MultiDBLookupMapper prop.X.connectionInfo=sourceRepo prop.X.sql=SELECT customer_name, customer_address1, customer_city, customer_state FROM customer_db WHERE customer_code = ? prop.X.defaultValue=CUST_NAME_NOT_FOUND,CUST_ADDR1_NOT_FOUND,CUST_CITY_NOT_FOUND,CUST_STATE_NOT_FOUND
	Example 2: The DB table provides four properties that are either true (Y) or false (N). These properties are: city_gov, county_gov, state_gov, federal_gov, whereby the values indicate the level of government, and only one of the four can be true. We are mapping to a new property, gov_level, which will have a text representation of this value.	prop.X.oldname=customer_code prop.X.newname=customer_code prop.X.mapperclass=com.emc.monorail.tools.transform.MultiDBLookupMapper prop.X.connectionInfo=sourceRepo prop.X.sql=SELECT city_gov, county_gov, state_gov, federal_gov FROM customer_db WHERE customer_code = ? Prop.X.resultmapmode=concat Prop.X.resultmap.0.value.default=NOT FOUND!! Prop.X.resultmap.0.valuecount=4 Prop.X.resultmap.0.value.0.oldvalue=YNNN Prop.X.resultmap.0.value.0.newvalue=CITY Prop.X.resultmap.0.value.1.oldvalue=NYNN Prop.X.resultmap.0.value.1.newvalue=COUNTY Prop.X.resultmap.0.value.2.oldvalue=NNYN Prop.X.resultmap.0.value.2.newvalue=STATE Prop.X.resultmap.0.value.3.oldvalue=NNNY Prop.X.resultmap.0.value.3.newvalue=FEDERAL
MultiDBLookupWithCacheMapper	**com.emc.monorail.tools.transform.MultiDBLookupWithCacheMapper** This mapper executes the SELECT statement and populates value(s) exactly as MultiDBLookupMapper, but it also populates a cache with values keyed by the input parameters. When subsequent calls are made with the same parameters, the cached value is returned instead of executing another query. This is very suitable where large numbers of rows need to be processed, but the combinations of input parameters are small (and therefore we hit the cache more often than not).	 For connection info details and example, see DBLookupMapper, for other configuration, see MultiDBLookupMapper.
NullValueMapper	**com.emc.monorail.tools.transform.NullValueMapper** Sets the property to a java null value. Used to remove repeating properties, or to allow the property value of a single property to be overwritten by a default value during ingestion.	
RegExMapper	**com.emc.monorail.tools.transform.RegExMapper** Used to apply a regular expression to the value of a property. The regular expression consists of two parts: a pattern to match, and a pattern	prop.X.oldname=<source property name> prop.X.newname=<target property name> prop.X.mapperclass=com.emc.monorail.tools.transform.RegExMapper prop.X.expCount=<number of expressions to apply>

	with which to replace the text. There are a lot of resources on the web on regular expressions, for example: http://docs.oracle.com/javase/tutorial/essential/regex/index.html.	`prop.X.exp.Y.match=<Yth matching pattern>` `prop.X.exp.Y.replace=<Yth replacement pattern>`
	Example: Remove leading and trailing space in folder names from the r_folder_path property  Regular expression format is very sensitive to escape characters. The example above performs three regular expressions: 0: match and number of trailing spaces on the property value, replacing with a zero length string – essentially removes all trailing spaces 1: match any number of spaces before a path separator ("/"), replacing simply with the separator 2: match any number of spaces after a path separator ("/"), replacing simply with the separator	`prop.X.oldname=r_folder_path` `prop.X.newname=r_folder_path` `prop.X.mapperclass=com.emc.monorail.tools.transform.RegExMapper` `prop.3.expCount=3` `prop.3.exp.0.match=\\ +$` `prop.3.exp.0.replace=` `prop.3.exp.1.match=\\ +/` `prop.3.exp.1.replace=/` `prop.3.exp.2.match=/\\ +` `prop.3.exp.2.replace=/`
Remove CustomACL Mapper	**com.emc.monorail.tools.transform.RemoveCustomACLMapper** This mapper checks for documents that have custom ACLs, and replaces the ACL of such documents with the ACL of their parent folder. This was done to reduce the number of custom ACLs in a repository.  For connection info details and example, see DBLookupMapper.	`prop.X.oldname=acl_name` `prop.X.newname=acl_name` `prop.X.mapperclass=com.emc.monorail.tools.transform.RemoveCustomACLMapper` `prop.X.defaultACL=<default ACL to apply if not found in the repo>` `prop.X.connectionInfo=<OPTIONAL: if absent, lookup in Mongo>`
	Example:	`prop.X.oldname=acl_name` `prop.X.newname=acl_name` `prop.X.mapperclass=com.emc.monorail.tools.transform.RemoveCustomACLMapper` `prop.X.defaultACL=Default App ACL` `prop.X.connectionInfo=sourceRepo`
RepeatingTo SingleMapper	**com.emc.monorail.tools.transform.RepeatingToSingleMapper** EMA stores single and repeating properties in separate collections, in much the same way that single and repeating properties are stored in separate tables in the database. In order to move from one to the other, a mapper is required. This mapper moves repeating properties to the single collection.	`prop.X.oldname=<source repeating property>` `prop.X.newname=<same as oldname>` `prop.X.mapperclass=com.emc.monorail.tools.transform.RepeatingToSingleMapper` `prop.X.transmode=<SIMPLE or MERGE. SIMPLE maps the value at index 0 from the repeating property into the single property. MERGE concatenates all values into a single value, appending the separator defined below>` `prop.X.target=<name of single property to receive the value>` `prop.X.separator=<separator string if using MERGE mode>`

	Example: Place all keyword values into a single property, all_keywords, separated by a comma.	`prop.X.oldname=keywords` `prop.X.newname=keywords` `prop.X.mapperclass=com.emc.monorail.tools.transform.RepeatingToSingleMapper` `prop.X.transmode=MERGE` `prop.X.target=all_keywords` `# need to escape the space or the Properties class will trim it out!` `prop.X.separator=,\`
SingleTo Repeating Mapper	**com.emc.monorail.tools.transform. SingleToRepeatingMapper** As for RepeatingToSingleMapper, this mapper effectively moves a property from being a single property to being a repeating property.	`prop.X.oldname=<source single property>` `prop.X.newname=<same as oldname>` `prop.X.mapperclass=com.emc.monorail.tools.transform.SingleToRepeatingMapper` `prop.X.transmode=<SIMPLE or SPLIT. SIMPLE maps the value of the single property to the value of the repeating property at index 0. SPLIT splits the value based on a defined separator and writes all values into the repeating property at index 0..N>` `prop.X.target=<name of single property to receive the value>` `prop.X.separator=<separator string if using SPLIT mode>`
	Example: Split the value of all_keywords by a comma and write back into keywords.  In this case, the repeating values overwrite existing values (if present).	`prop.X.oldname=all_keywords` `prop.X.newname=all_keywords` `prop.X.mapperclass=com.emc.monorail.tools.transform.RepeatingToSingleMapper` `prop.X.transmode=SPLIT` `prop.X.target=keywords` `# trailing space of separate needs to be escaped` `prop.X.separator=,\`
SplitTo Multiple Property Mapper	**com.emc.monorail.tools.transform.SplitToMultiplePropertyMapper** Splits the string value into multiple properties. In this case, all properties belong to the same collection (single or repeating). It is unlikely that this mapper will be used much as-is but may form the basis of a custom mapper	`prop.X.oldname=<source property>` `prop.X.newname=<same as oldname>` `prop.X.mapperclass=com.emc.monorail.tools.transform.SplitToMultiplePropertyMapper` `prop.X.separator=<separator to split the value>` `prop.X.proplist=<comma separated list of properties to receive the data>`
	Example: Split the value of title, put the part before the ":" into the object_name, and the part after the ":" into the subject property.	`prop.X.oldname=title` `prop.X.newname=title` `prop.X.mapperclass=com.emc.monorail.tools.transform.SplitToMultiplePropertyMapper` `prop.X.separator=:` `prop.X.proplist=object_name,subject`
SubString Mapper	**com.emc.monorail.tools.transform.SubString Mapper** Perform a basic string manipulation on the incoming value  For substring, start and end use the same 0 start indexing as the String.substring() java function.	`prop.X.oldname=<source of value>` `prop.X.newname=<destination of value>` `prop.X.mapperclass=com.emc.monorail.tools.transform.SubStringMapper` `prop.X.target=<OPTIONAL name of property to write value to, if source should be preserved>` `prop.X.function=<right, left or substring>` `# for "right":` `prop.X.rightchars=<number of characters to take from the end of string>` `# for "left":`

		prop.X.leftchars=<number of characters to take from the beginning of the string # for "substring": prop.X.start=<start position> prop.X.end=<end position>
	Example: Take the last four digits of an SSN into a new property called last_4_ssn.	prop.X.oldname=ssn prop.X.newname=ssn prop.X.mapperclass=com.emc.monorail.tools.transform.SubStringMapper prop.X.function=right prop.X.rightchars=4 prop.X.target=last_4_ssn
TrimMapper	**com.emc.monorail.tools.transform.TrimMapper** This mapper simply applies the String.trim() java function to the value.	prop.X.oldname=<source of value> prop.X.newname=<destination of value> prop.X.mapperclass=com.emc.monorail.tools.transform.TrimMapper
	Example: Trim the object name of folders/cabinets to remove spaces, that are illegal in newer Documentuim versions, but were possible in earlier versions.	prop.X.oldname=object_name prop.X.newname=object_name prop.X.mapperclass=com.emc.monorail.tools.transform.TrimMapper
ValueList Mapper	**com.emc.monorail.tools.transform.ValueListMapper** This mapper is used to replace a value if it matches one of a fixed list of values. This can be used for multiple different purposes, two examples are shown below. Configuration Properties:	prop.X.oldname=<source of value> prop.X.newname=<destination of value> prop.X.mapperclass=com.emc.monorail.tools.transform.TrimMapper prop.X.valuecount=<number of values in the list> prop.X.value.Y.oldvalue=<old value> prop.X.value.Y.newvalue=<new value>
	Example 1: PROJECTS and TEMPLATES cabinets are being merged from repositories of different departments. We need to modify the object_name of the cabinet if it matches PROJECTS or TEMPLATES	prop.X.oldname=object_name prop.X.newname=object_name prop.X.mapperclass=com.emc.monorail.tools.transform.ValueListMapper prop.X.valuecount=2 prop.X.value.0.oldvalue=PROJECTS prop.X.value.0.newvalue=Marketing Projects prop.X.value.1.oldvalue=TEMPLATES prop.X.value.1.newvalue=Marketing Templates
	Example 2: Replace the two-letter state abbreviation with the full name	prop.X.oldname=us_state prop.X.newname=us_state prop.X.mapperclass=com.emc.monorail.tools.transform.ValueListMapper prop.X.valuecount=50 prop.X.value.0.oldvalue=AK prop.X.value.0.newvalue=Alaska prop.X.value.1.oldvalue=AL prop.X.value.1.newvalue=Alabama prop.X.value.2.oldvalue=AR prop.X.value.2.newvalue=Arkansas

Creating a Custom Transform

Definition

Essentially, all that is needed to create a custom transform is a java class that extends the AbstractPropertyValueMapper class. The abstract class is part of the core EMA Transform JAR file. This class itself will need to be in the classpath when the transform engine is running.

Example 1:
Create a mapper that reformats a numerical value into a string, depending on a configurable format.
We will use java.text.DecimalFormat as the basis for this change, and read configuration from the transform properties file.

```
prop.X.oldname=<source property>
prop.X.newname=<target property>
prop.X.mapperclass=com.emc.examples.NumberFormatMapper
prop.X.format=<format to apply, see java.text.DecimalFormat for details>
prop.X.target=<OPTIONAL: property to write out to, if not newname>
```

Creating a Custom Transform

Step 1	Create the class. If using Eclipse, and entering the superclass in the "New Class" dialog, the required methods will be created automatically. There are two methods: init and mapValue.
Step 2	Complete the init() method. This method reads the transform properties file to get the mapper specific configuration. In our case, we need to read format and target. Target is optional, but if format is not present, we should raise an exception
Step 3	Complete the mapValue() method. Convert the input text into a Double, then use the formatter to put it in the requested format as a String. If target is present, write the value out into this property, otherwise return the formatted value from mapValue().
	Use log4j to log information within the methods, but remember that the framework will log all mapped values already so only log additional information if appropriate.

Code

```
package com.emc.examples;
import java.util.Properties;
import com.mongodb.DB;
import com.mongodb.DBObject;

import java.text.DecimalFormat;
import java.text.NumberFormat;
import com.emc.monorail.tools.transform.AbstractPropertyValueMapper;
import com.emc.monorail.tools.transform.ValueMapperException;

import org.apache.log4j.Logger;

public class NumberFormatMapper extends AbstractPropertyValueMapper {
	NumberFormat nf;
	String target = null;
	Logger lgr = Logger.getLogger(this.getClass());

	@Override
	public void init(Properties props, String propertyPrefix)
			throws ValueMapperException {
		// read the configuration file
		lgr.info("Initializing NumberFormatMapper...");
		String format = props.getProperty(propertyPrefix + ".format");
		if (format == null) {
			// if format is absent, throw an exception
			lgr.error("Invalid transform configuration: " + propertyPrefix + ".format is missing");
			throw new ValueMapperException("Invalid transform configuration: " +
			propertyPrefix + ".format is missing");
		}
		// create formatter object that we will reuse for each row
		nf = new DecimalFormat(format);
		target = props.getProperty(propertyPrefix + ".target");
	}

	@Override
	public String mapValue(DB db, DBObject dbo, String propName)
			throws ValueMapperException {
		// get the current value
		String currValue = (String)dbo.get(propName);

		// catch any errors
		try {
			// convert value to a double
			Double currDbl = Double.parseDouble(currValue);
			// format into a string
			String formattedValue = nf.format(currDbl);
			// either return this value, or write it into the separate property if present
			if (target != null) {
				dbo.put(target, formattedValue);
			} else {
				currValue = formattedValue;
			}
		}
		catch (Exception exc) {
			lgr.error("Error formating property " + propName + " for object: " +
			(String)dbo.get("r_object_id"));
			throw new ValueMapperException("Error formating property " + propName + " for
			object: " + (String)dbo.get("r_object_id"));
		}
		return currValue;
	}
}
```

JavaScript Approach (Deprecated)

A sample JavaScript file "*transformJSSample.js*" is provided in the "*samples*" directory.
In the more complex transformation use cases (e.g. DCM to D2LS Suite) the properties file approach can become very time consuming to create, and very difficult to maintain as changes are made to the transformation design. In order to address this, in March 2014 we introduced the option to create transformations using the Java 7 embedded JavaScript engine.
Essentially, the approach is the same as for the properties file: each row of the SP collection is read and processed, and each row of the RP collection is read and processed. The original collections remain intact as "ORIG" collections, and a single type going in can be split into multiple types during transformation. A shared library provides the building block functions that will be used by most transformations, and the transformation JavaScript for a given type can make calls to additional libraries, enabling us to share common code (e.g. applying transformation to properties defined at the super-type level).

Script Structure

We provide a script that includes many building blocks that will be used by most transformations (transfromLibrary.js), the developer creates a script that performs transformations for that specific transformation (see below), and can also create library script(s) to implement common segments that can be included in multiple transformations (e.g. transformations on a super type that need to be applied for all subtypes).

Anatomy of a Transformation Script

A transformation script needs to follow the following structure:

```
oldTypeName = "<pre-transformation type name>";
newTypeName = "<post-transformation type name>";

function init() {
        /* place lookup initializers, any other common variables here */
        /* call shared library initializers here, if required */
}

function transform(repeating) {
        if (repeating == "true") {
                /* perform RP transformations */
        } else {
                /* perform SP transformations */
        }
}

/* optional: */
function deInit() {
        /* perform any deinitialization, here or in the shared libraries */
}
```

Building Blocks

The following functions are part of the transfromLibrary.js core library:

```
/* used internally - returns the value of oldTypeName defined in the specific transform script */

function getOldTypeName()

/* used internally - retuens the value of newTypeName defined in the specific transform script */
function getNewTypeName()

/* set the value of a property */
function set(varName, value)

/* get the value of a property (as a generic object - see below) */
function get(varName)

/* copy the value of a property to another property */
function copy(varName, newVarName)
```

```
/* check if the property is present (or = java NULL) */
function isnull(varname)

/* checks if the property is present */
function containsfield(varname)

/* initialize a DB lookup mapper for the specified configuration
   the object returned must be global for use later in the "transform" routine */
function initDBLookupMapper(config)

/* initialize a DB lookup mapper (with cache) for the specified configuration
   the object returned must be global for use later in the "transform" routine */
function initDBLookupWithCacheMapper(config)

/* initialize a Conditional DB lookup mapper (with cache) for the specified configuration
   the object returned must be global for use later in the "transform" routine */
function initExtendedDBLookupWithCacheMapper(config)

/* initialize a Merge from DB mapper for the specified configuration
   the object returned must be global for use later in the "transform" routine */
function initMergeFromDBMapper(config)

/* initialize a DB lookup mapper with full parameters for the specified configuration
   the object returned must be global for use later in the "transform" routine */
function initMergeFromDBMapperEx(config)

/* checks if the value of the given property matches a specific value */
function equals(varname, value)

/* initialize an Edit Repeating Attribute mapper for the specified parameters (see properties file examples for
details).Tthe object returned must be global for use later in the "transform" routine */
function initEditRepeatingAttributeMapper(sourceAttribute,targetAttribute, removeList, mapList, mapValueList,
sethasfolder)

/* used internally - developers can use the singleToRepeating and repeatingToSingle functions to cause
transformations, they do not need to
function initSingleToRepeatingMapper(mode, sourcelist, targetlist, sep)

/* called by the main java code during initialization. For internal use only
   includes the call to the specific transform's "init()" function */
function initStandardObjects()

/* move a property from single to repeating. Call during processing of single properties */
function singleToRepeating(source, target)

/* move a property from single to repeating with more options (see properties file description
   for details) */
function singleToRepeatingEx(mode, source, target, sep)

/* move a property from repeating to single. Call during processing of repeating properties */
function repeatingToSingle(source, target

/* move a property from repeating to single with more options (see properties file description
   for details) */
function repeatingToSingleEx(mode, source, target, sep, length)

/* create a relation between parent and child objects. Call additional methods of
   com.emc.monorail.api.Relation to set additional properties, and remember to call write() to
   save changes to Mongo */
function createRelation(relationName, parentId, childId, permanentLink)

/* initialize an Adjust Version Label mapper for the specified parameters (see properties file
   examples for details). The object returned must be global for use later in the
   "transform" routine */
function initAdjustVersionLabelMapper(mode, srcAttr, trgAttr, remove)
```

```
/* create a relation between parent and child objects. Call additional methods of
   com.emc.monorail.api.Document to set additional properties, and remember to call write() to
   save changes to Mongo */
function createDocument(objName)
```

"logger" object in JavaScript gives you access to log4j from your JS code:
logger.info("This is awesome");

Reusing DB lookups and Custom Mappers

We have incorporated the ability to use some of our existing mappers also from within JavaScript code. The following mappers have been adapted to date, if you require another mapper to be accessible, for example a custom mapper developed earlier for the properties approach, then please contact the EMA development team to discuss. In some cases, actions previously performed by mappers are easily done within JavaScript (e.g. combining field values, parsing strings) but for others it may be more appropriate to adapt the code for use from within JavaScript.

Accessing the EMA API from a Javascript Transformation

This is a capability unique to the JavaScript model and it enables the transformation to “burst” or “normalize” an object model, or create relations or new folder structure during transformation. This is an extremely powerful capability particularly useful in xCP transformations.
To access the EMAFactory instance, use the JavaScript variable “apiFactory” directly, or use the core library functions provided in transformLibrary.js, e.g. to create a relationship to another object based on a property containing an “ID”:

```
function transform(repeating) {
        If (repeating == "false") {
                var relatedObjId = get("related_object_id");
                If (relatedObjId != "0000000000000000") {
                        var newRelation = createRelation("related_document", get("r_object_id"), relatedObjId,
true);
                        newRelation.write();
                }
        }
}
```

Creating new document objects works in much the same way. In this example, we create a new document object containing comment information, and create a relation to link to the current document:

```
function transform(repeating) {
        if (repeating == "true") {
                var commentText = get("comment_text");
                if (commentText != null) {
                        var newDoc = createDocument(commentText);
                        newDoc.setObjectType("comment_doc");
                        newDoc.setModifyDate(get("comment_date");
                        newDoc.setCreationDate(get("comment_date"));
                        newDoc.setOwnerName(get("comment_user"));
                        newDoc.write();

                        var newRelation = createRelation("related_document", get("r_object_id"),
newDoc.getObjectId(), true);
                        newRelation.write();
                }
        }
}
```

FolderCache objects can also be used (remember to use the deInit() function to write out all objects stored in the folder cache!). In this example, we use the “product_name” property to build out the folder structure:

```
var fc;

function init() {
        fc = apiFactory.newFolderCache();
}

function transform(repeating) {
        if (repeating == "false") {
                var productName = getString("product_name");
                var folderName = "/Products/" + productName;
                var folder = fc.getCreateFolder(folderName);
                set("temp_folder_id", folder.getObjectId());
                singleToRepating("temp_folder_id", "i_folder_id");
        }
}

function deInit() {
        fc.writeAll();
}
```

Note: This approach will also work where the folder structure, or part of the folder structure, already exists in the target.

INGESTION

Definition

Ingests data to the target Documentum repository. Reads data from the staging area (MongoDB)

Sample ingest script "*Ingest.bat*" along with sample storage map "*changeFilestore.storagemap.properties*" is provided in the "samples" directory

Do take your database backup before starting Ingestion so in case you encounter any issue during Ingestion you can quickly restore back to the correct state.

Ingestion Considerations

Storage mapping

Plan your approach to storage management ahead of time! Review details in: Storage Management

Format of the Entries in the Storage Map

```
#Required entry in the storage map, one per source filestore:
source.<source filestore name>=<target filestore>
#Optional entries, used to provide full path in the file copy list
source_prefix.<source filestore name>=<path to start of source filestore>
dest_prefix.<source filestore name>=<path to start of target filestore>
```

dest_prefix uses the source filestore name as a suffix to determine the destination path, not the target filestore name. Paths should be provided up to the "80" directory that starts the filestore structure. Use forward slashes for compatibility with Java even on windows platforms.

Source system is Documentum

Is your content already copied into a new filestore?

Yes, then we need to map the source filestore to the new filestore in the target in the storagemap.properties file (using *--storage-map*)

```
#StorageMap.properties
source.filestore_01=legacy_01
source.filestore_02=legacy_02
```

If No, are you planning to copy the content to an existing filestore or a new filestore

If you are copying the content to an existing filestore (which already has content files) we have to merge the content (using *--merge-content*) so that the data tickets don't collide. EMA takes care of updating the last data_ticket sequence number so nothing has to be done manually.

Here is how the storage map properties file should have the entries

```
#StorageMap.properties
source.filestore_01=filestore_01
source_prefix.filestore_01=E:/Documentum/data/Test2/
content_storage_01/000004d2/
dest_prefix.filestore_01=E:/Documentum/data/Test15/
content_storage_01/0000264f/

source.filestore_02=filestore_01
source_prefix.filestore_02=E:/Documentum/data/Test2/
content_storage_02/000004d2/
dest_prefix.filestore_02=E:/Documentum/data/Test15/
content_storage_01/0000264f/
```

	Source system is a 3rd party system "default" is used by the API as the source for 3rd party systems. Therefore, we need to provide a destination mapping for this filestore.
	`#StorageMap.properties` `source.default=legacy_01`
File Copy	If the content has not been already moved and we are planning to move the content using EMA see FileCopier. This is applicable for merge-content, moving content to a new filestore and moving content from 3rd party system scenarios.  File List log file generated by the Ingestor must be renamed before each run so that we don't override the existing file from a previous run.
Default files	**When** 1. We migrate from a 3rd party system. 2. We migrate from an older version of Documentum to a new version. In this scenario there can be mandatory attributes that are in the newer version but were not present in the older version. **How to Create** User can use DefaultFileCreator Tool to create the default files. In case the requirement is only to **override few attributes with constant values** transform can be little slower as it works traversing each row one by one. Ingestor has a way to handle this. Prefix the default value (in the defaults file) with an "*". To null a value, use the word "null".
	`i_latest_flag=*0`
	 1. The folder where the default files are stored **must be provided in the classpath of the Ingestor** so it picks up the default files. 2. EMA looks for the defaults file for a given type anywhere in the classpath: defaults_<type name>.properties 3. This file typically contains only single value attributes, no repeating attributes. (The files generated by DefaultFileCreator have repeating attributes too as they can be used in EMA API which supports repeating attribute defaults) 4. We can use #include statement to build inheritance of default values. 5. E.g. if we have a type called corp_doc which has a parent type dm_document we can use the below line in the "defaults_corp_doc.properties" file to include all defaults properties from the parent type: 6. #include <path to the file>/defaults_dm_document.properties 7. To add a blank value for a string use 8. Attribute=\<blank> 9. Where <blank> is an actual space " " 10. For Integer/Boolean we use 0 for false and 1 for true. Default values for a type are output to the log4j log file during initialization of the

	injector, if you are having issues, check that the defaults are being correctly loaded.
Custom attributes (single/ repeating) having object id's in them	To provide custom attributes which have object id values in them (both single and repeating) we need to provide the information of those attributes in the properties file. This is the same file which can contain the command line options. Provide the file as a VM argument while running the Ingestor like -Doptions.default="E:/ema/IngestorProperties.properties" The convention to be followed is: `id.<type>.single=attr1,attr2` `id.<type>.repeating=rattr1,rattr2` Where <type> = the object type having the attribute with object id.
"Lost+ Found" folder	If you ingest an object who's parent folder is not part of the data set, the ingestor will place that object (e.g. earlier version of a doc moved into the cabinet we extracted) into a folder specified by the --lost-found "<path>" parameter (e.g. –lost-found "/Temp/Lost+Found"  For the documents that landed in the Lost & found folder there was no easier way to figure out the original folder where the particular document resided in the source location. To overcome this Ingestor now generates a LostFound List file which has the document object id, its original folder id and the new folder object id assigned.  Folder must exist before the ingestor runs as Ingestor will not create it.
Restart/ Delta mode	Restart mode is to be used when Ingestion fails and you have to rerun it after fixing the issue. This mode checks for the existence of DB rows before executing an INSERT statement. If a row exists, it is skipped (DELTA mode would execute an UPDATE) Delta mode is to be used for "delta migrations" which means it would have updated documents as well as new documents as part of the delta. When you are in Restart/Delta mode you have to provide "--preload-dbs" with all the db's that were earlier provided in the "--dump-ids".  If preload-dbs are not provided for ingestion with restart/delta mode then ingestion will start from scratch and new object id's would be assigned to each document.  Delta type and delta column are used for 3rd party migrations, where a specific field identifies the document in the target system. These options should not be used for Documentum-to-Documentum deltas.

	Keep ID mode	The object ID's are retained for all types except for dm_user, dm_group, dm_acl, dm_format, dm_folder, dm_cabinet. For these • If the object id is already occupied in the target then a new object ID is assigned. • If the object already exist in target then it is not replaced in the target (also entries in MongoDB are removed so that ingestion for them is not performed) unless that type is specified in the replace list option. Here is how the repository is to be configured while creating it: 1. Manipulate objectID counter in headscript script before repository installation. 2. Keep target repository ID same as source repository ID. 3. To find out how much IDs should be reserved please refer to: https://inside.dell.com/docs/DOC-222996?sr=stream&ru=21386 4. Code to add in headstart script before repository installation.
		`QueryID = dmAPIGet("apply,c,0000000000000000,NEXT_ID_LIST,TAG,I,<primary type> ,HOW_MANY,I, <count>")` `If QueryID <> "" Then` `status = dmAPIExec("close,c," & QueryID)` `Else` `Print dmAPIGet("getmessage,c")` `Print "Failed to extend ID"` `DmExit(-1)` `End If`
		Check the PPT deck (@EMA SyncP): "**KT_EMA-Retaining Object IDs.pptx**" for more details.
	Trim fields	There is a need to trim fields for DB2 to Oracle/SQL Server conversions. To achieve this override functionality has been added to EMA ETL IngestManager (as well as Cloner). See "Overrides.properties" file in the samples folder. Specify the override properties file by adding -DOVERRIDES=<filename> to the java command line running IngestManager Note: Clone JAR file MUST be included to run IngestManager
	Special Considerations	When any of dm_acl, dm_user, dm_group or dm_format is being migrated we need to consider if the existing object present in the target systm is to be replaced or not. By default EMA does not replace the existing object with the same name in the target. For the objects not existing in the target it creates new objects. In case you want to override the objects existing in the target you have to use the **delta** mode with **replace** option

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-rn	--repository-name	repo name	Yes	Name of target repository
-ru	--repository-username	username	Yes	Repository username
-rp	--repository-password	password	Yes	Repository password
-td	--target-driver	driver class	Yes	Target JDBC driver to use for connection
-tc	--target-connection	connect string	Yes	Target JDBC connection string
-tu	--target-user	username	Yes	Target JDBC username
-tp	--target-password	password	Yes	Target JDBC password
-mh	--mongo-host	hostname	Yes	Mongo DB Hostname
-mp	--mongo-port	port number	Yes	Mongo DB Port number

	-md	--mongo-db	db name	Yes	Mongo DB Database name
	-mu	--mongo-user	mongo user	Yes	Mongo Admin username
	-mpa	--mongo-password	mongo password	Yes	Mongo Admin password
	-sm	--storage-map	map file	Yes	Storage map if copying content
	-di	--dump-ids	mongo db name	Yes	Dump out the object ID map after ID assignment
	-dr	--dry-run		No	Runs ingestion w/o committing changes
	-nb	--no-batch		No	Submit single INSERTs instead of batches
	-thc	--thread-count	num threads	No	Number of threads to use for ingestion
	-mrg	--merge-content		No	Merge filestore from source to target
	-lf	--lost-found	folder path	No	Folder for documents when the parent folder is not migrated
	-fcl	--file-copy-list	filename	No	Location of the file copy list to be generated
	-ihc	--ignore-host-check		No	Ignore hostname check
	-rl	--replace	types seperated by ,	No	Replace the object if they already exist. Types can be dm_user,dm_group,dm_acl, dm_format, dm_folder, dm_cabinet (can be used only in delta mode)
	-mo	--mode		No	Takes value as DELTA | RESTART | INGEST (default)
	-pds	--preload-dbs	list of mongo db's	No	Load object IDs from existing Mongo DB(s)
	-k	--keep-ids		No	New IDs are not allocated to migrated objects
	-kf	--keep-ids-for	list of tags	No	 Retain object IDs for specific types. Tags are first 2 characters of the object ID, e.g. 09 (document), 0b (folder), 0c (cabinet)
	-dt	--delta-type	object type	No	Object type for which we have the Delta
	-dc	--delta-column	column name	No	Unique column of the Object type for Delta
	-h	--help		No	Show this text

Scenario

Help

java -cp "EMAIngestManager -1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.ema.ingestor.IngestManager --help*

Providing properties file for parameters

Options can be added to a properties file instead of providing them on the command line.In case an option is provided both in properties file as well as in command line, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/IngestorProperties.properties" -cp "EMAIngestManager-1.6.0.jar;EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config; C:/EMA/EMA1.6.0/dependency-jars/" com.emc.ema.ingestor.IngestManager*

IngestorProperties.properties

```
repository-name=test15
repository-username=Administrator
repository-password=Thom2807
target-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
target-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test15_docbase
target-user=sa
target-password=Thom2807
storage-map=E:/ema/sameFilestore.storagemap
mongo-host=localhost
```

	mongo-port=27017 mongo-db=Extractor_Cabinet mongo-user=admin mongo-password=Thom2807
Ingestion (SQL Server)	*java -cp "EMAIngestManager-1.6.0.jar;EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu sa -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md ExtractorDB_10 -mu admin -mpa Thom2807 -sm "C:\EMA\storagemap.properties" -di ID_BACKUP*
Ingestion (Oracle Database)	*java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar ; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -td oracle.jdbc.driver.OracleDriver -rn OracleDb -ru Administrator -rp Thom2807 -tc jdbc:oracle:thin:@10.8.58.48:1521:ORCL -tu source -tp source -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -sm "C:\EMA\storagemap.properties" -di ID_BACKUP*
Delta Mode (Documentum to Documentum migration)	**During the first Ingestion dump the ID's** *java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar; E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md NODELTA_TEST -mu admin -mpa Thom2807 -sm "C:\EMA\storagemap.properties" --dump-ids DELTA_BACKUP*
	Run the Ingestor in Delta mode *java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md NODELTA_TEST -mu admin -mpa Thom2807 -sm "C:\EMA\storagemap.properties" -di ID_BACKUP -mo delta --preload-dbs DELTA_BACKUP*
Delta Mode (3rd party to Documentum migration)	*java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md NODELTA_TEST -mu admin -mpa Thom2807 -sm "C:\EMA\storagemap.properties" -di ID_BACKUP -mo delta -dc object_name -dt um_invoice -mo delta*

Restart Mode	*java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md NODELTA_TEST -mu admin -mpa Thom2807 -sm "C:\EMA\storagemap.properties" -di NEW_BACKUP -mo restart --preload-dbs RESTART_BACKUP*
Dry Run mode  No commit of changes done on the target system	*java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu sa -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md ExtractorDB_10 -mu admin -mpa Thom2807 -di ID_BACKUP --dry-run*
Preserving object id's of the source system	 Refer to "**KT_EMA-Retaining Object IDs.pptx**" present @ SyncP
Replacing objects of particular type if they already exist in target	*java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu sa -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md ExtractorDB_10 -mu admin -mpa Thom2807 -mo delta -di ID_BACKUP --replace dm_user*
Providing name of filecopylist  A new parameter "--file-copy-list" is added where user can provide the filename along with the path where the file should be generated.	*java -cp "EMAIngestManager-1.6.0.jar; EMA-API-1.6.0.jar; EMA-Clone-1.6.0.jar;E:/Documentum/config;C:/ema/config; C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.ema.ingestor.IngestManager -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu sa -tp Thom2807 -mh 127.0.0.1 -mp 27017 -md ExtractorDB_10 -mu admin -mpa Thom2807 -di ID_BACKUP -fcl Testfilelist*

DOCUMENTUM DELTA MIGRATIONS

Because of the volume of data being migrated, and restrictions on production system outages, it is sometimes necessary to perform an initial migration and then subsequently one of more delta migrations. In this context, a delta migration refers to migrating a specific part of the source data, usually defined by being data that has been added and/or modified since the initial migration, and merging this into the already migrated data in the target.

Deltas will typically handle use cases such as:

- New folder structures added to the system
- New documents added to the system
- Modifications to existing folder structures
- Modifications to existing documents
- New versions of existing documents

Some other delta use cases are NOT current supported by EMA-Migrate. These include:

- Deleting documents or folders
- Modifying relationships between documents
- Modifying virtual document structures

If your use case requires any of these use cases, please consult with the EMA team and we will determine a way forward for your current situation. Deleting objects is planned for EMA 1.7, but in the meantime other solutions will need to be found.

The following sections describe the particular configuration required at each stage for a Delta Migration.

Preparing for Delta Migrations

As the source system will stay live and will have constant changes to documents as it is still in use, it is very helpful to take a snapshot of the source DB and migrate the bulk of the data from this snapshot rather than from the live system.

This gives the following advantages:

- Extraction processing has no effect on the running production system performance
- Extraction/Transformation/Ingestion verification can be done against a fixed starting point

Typically we would recommend a short downtime of the source system, during which a copy is taken to a separate DB instance from which we can run extraction processes. This ensures consistency across all tables in the Documentum Schema

Delta Extraction

While it is technically possible to extract all data and reconcile what has/has not changed at ingestion time, it makes sense to extract only data that has been modified since a specific point in time (when the initial migration took place, or when the DB snapshot was taken (see above).

To control the data that is extracted, use a “where clause” that includes a timestamp, e.g. s.r_modify_date > DATE(‘12/25/2016’). The extraction process will extract all sysobjects that satisfy the criteria into Mongo for further processing, reference objects as for initial extractions, and objects related to the sysobjects that are extracted (relations, virtual document structure, ACLs and content objects).

Note: in contrast to the initial extraction, it is not necessary to extract the folder structure of for all documents (if it has not changed). Linking of documents to the correct folder in the target can be handled during ingestion (see below).

Delta Transformation

Transformation is for the most part not affected when running deltas: the same transformation rules will apply, and the same configuration files (properties/javascript/java) can be used for the transformation.

Delta Ingestion

During ingestion, we must allow for the possibility that the object being ingested already exists. As this is time-consuming if we know it will not be the case, a parameter is used to switch on this mode of operation: --mode DELTA. In delta mode, when objects are ingested, we first check to see if the object already exists. If the object exists, we update the DB with the new object values. If the object does not already exist, we insert the object into the target system.

There are two types of check for the object already existing:

- Object ID already in use: Object ID must be unique, so the existing object is overwritten by the new object
- Object with other source of uniqueness already exists (e.g. Folders, ACLs): existing objects are overwritten ONLY if the object type is included in the --replace list parameter

This can become quite complex in cases where we are merging different repositories with overlapping folder or ACL structures. Analysis is required in each migration project to determine the correct approach for the situation, please consult with your regional migration practice lead and the EMA team to confirm the approach before proceeding.

Preloading ID Databases: In cases where we are relying on the pre-existence of objects from a previous migration, the --preload-dbs parameter is used to specify the various object ID DBs in Mongo that contain the objId/newObjId pairings from all previous runs. Preloading the DBs in this way enables EMA to replace references to objects that were not part of the delta extraction to the correct new object IDs in th target.

Operating in keep-ids mode: If we are keeping object IDs for all objects (i.e. including folders) then we do not need to preload IDs from previous runs because the IDs do not change anyway.

Clone Migration flag: If we are migrating data from/to repositories with the same docbase ID, but not preserving all object IDs, then add this flag as a java parameter to the command line: -DCLONE_MIGRATION=true. This tells the system to retain object ID references during ingestion that are to objects of the same docbase ID, as the assumption is that these objects will already be present in the target.

Delta File Copy

In Documentum, if new content is added, as a new document, a new version, or saved as the same version, a new file is always created in the filestore. However, as we extract content objects for all modified objects, even if the modification was to meta-data only, we may create a file copy list that includes files that were already copied in the initial migration. These files will be copied again, but we can omit the parameter –overwrite so that existing files are left in place, and new content files will be copied.

If we are using a legacy filestore approach, whereby the folder structure and file names are retained from source to target, we can use robocopy or similar tools to copy all files added after the cut off date of the extraction.

EMA-API

Definition	To extract data from a third party source like a CSV, YAML or database where Documentum is not the source system, we need to build our own custom adaptor to work in place of Extractor. EMA provides a flexible framework to build our own custom adaptors.
How	**Step 1** - Create a Java project in your workspace **Step 2 -**Add a dependent jar named "EMA-API-1.6.0 jar" to your project (Check for updates to the jar in the PATCHES directory in SyncP). **Step 3 -** Create an Adaptor class and make it extend EMAAdaptor abstract class (this class is part of the EMA-API jar and provides basic method to initialize your source connections, get data row by row and close the connections). **Step 4 -** As the first point in your adaptor, create an instance of EMAFactory class. EMAFactory class is a supporting class that helps build Folders, documents etc. **Step 5 -**Write your custom code.

Familiarity with EMA-API

Steps	Description	Code
1.	Create an Adaptor class and make it extend EMAAdaptor class	`public class CSVAdaptor extends EMAAdaptor {}`
2.	Overload the method of EMAAdaptor class. There are three abstract functions in this class to parse, validate parameters and print usage for the application.	`@Override` `public void parseParameters(String[] args) {` `while (ix < args.length) {` `// what does this param signify?` `if (args[ix].equals("-mh") || args[ix].equals("--mongo-host")) {` `setMongoHost(args[++ix]);` `}` `else if (args[ix].equals("-mp")` `}} // Add other parameters according to the parameters your application accepts`
3.	Parse command line parameters, validate it and jump to the actual run.	`public static void main(String[] args) {` `lgr.info("Starting CSVAdaptor processing.. ");` `CSVAdaptor ing = new CSVAdaptor();` `ing.parseParameters(args);` `ing.run();` `}`
4.	Create an Instance of EMAFactory class. There are other overloaded versions of EMAFactory constructor available in EMA-API jar.	`public void run() {` `if (!validate()) {` `printUsage();` `return;}` `long start = System.currentTimeMillis();` `try {` `// init mongo connection` `try {` `EMAFactory emaf;` `if (this._mongoUsername == null) {` `emaf = new EMAFactory(this._mongoHost,` `this._mongoPort,getSettingsFile(), this._mongoDB);` `} else {` `emaf = new` `EMAFactory(this._mongoHost,this._mongoPort,getSettingsFile(),` `this._mongoDB, this._mongoUsername,this._mongoPassword);` `}`

	5.	A ReaderFactory class is provided to provide object of the reader you are reading data from. Currently only CSV and Database reader objects are supported	SourceReader reader = ReaderFactory.*getReader*(getSource(), bean
		getReader() has many overloaded versions to provide an object according to Source. If you are reading data from a source other than CSV or database, modify the ReaderFactory class and create your own Reader class implements SourceReader and overload its three methods according to your source.	**public class** CSVTypeReader **implements** SourceReader { @Override **public void** init() **throws** IOException { String fileToRead = getFileName(); **if** (!fileToRead.substring(fileToRead.lastIndexOf(".")) .equalsIgnoreCase(".csv")) { *lgr*.error("Cannot read a file other than CSV. Please specify a CSV file and re-run the Adaptor"); System.*exit*(0); } *lgr*.info("Reading input CSV File :" + fileToRead); csvReader = **new** CSVReader(**new** InputStreamReader(**new** FileInputStream(fileToRead), "UTF-8"), csvSeperator, CSVParser.*DEFAULT_QUOTE_CHARACTER*, '\0'); csvRowHeader = csvReader.readNext(); } @Override **public** Map<String, String> getNextRow() **throws** IOException { Map<String, String> rowData = **new** HashMap<String, String>(); String[] csvRowData = csvReader.readNext(); **if** (csvRowData == **null**) { **return null**; } **for** (**int** i = 0; i < csvRowHeader.length; i++) { rowData.put(csvRowHeader[i], csvRowData[i]); } **return** rowData; } @Override **public void** close() **throws** IOException { **if** (csvReader != **null**) { csvReader.close(); } }
	6.	To create a folder hierarchy, we can use FolderCache as follows :  This will create a cabinet named 'Migration', with a subfolder 'Product 1' and two folders under.	FolderCache folderCache = emaf.newFolderCache(); Folder folder = folderCache.getCreateFolder("/Migration/Product 1/Distribution Manual"); Folder folder1 = folderCache.getCreateFolder("/Migration/Product 1/EQMS"); folderCache.writeAll();
	7.	To create a document under EQMS folder with version 1.0, we can use newDocument() method to create the document.	Document doc = emaf.newDocument("<Name of Document>"); doc.setParentFolder(folder1); doc.setRepeatingAttribute("r_version_label","1.0"); doc.write();

	8.	To create a new version of the document, created in step 7, use cloneForNewVersion() method. This method has two parameters: 1. *Version_label*: the version of which you want to create a new version. For eg : If your document already has 2 versions namely 1.0 and 2.0 and you want to create a new version 3.0 then you will mention 2.0 in version_label parameter. 2. *keepContent* : this copies the content to the new version if set to true.	Document doc1 = (Document) doc.cloneForNewVersion("1.0", **false**); doc1.write ();
	9.	To create a virtual document using EMA-API, we can use the method addVirtualDocumentComponent() method. It has two overloaded version, one that accepts the object to be added as component, other one accepts the object id/chronicle id of the component to be added.	Containment childComp = doc.addVirtualDocumentComponent("<object id of child component>"); // Set version_label of Containment object childComp.setVersionLabel(versionLabel); childComp.write();
	10.	To create a relation, we use newRelation() method of EMAFactory class.	Relation newRelation = emaf.newRelation("Migrated Relation"); // The parameter signifies the relation_type in Documentum. newRelation.setParentId(parentId); // r_object_id of parent newRelation.setChildId(childId); // r_object_id chronicle_id of child //Set version_label to child version label newRelation.setChildLabel(versionLabel); newRelation.setPermanentLink(**true**); newRelation.write();
	11.	To clear mongoDB collections before next run, use the method	emaf.clearDocumentCollections();

File System Adaptor

Definition

A sample adaptor called FileSystemAdaptor (*FileSystemAdaptor.zip*) is provided in the "samples" directory. This can be used as a starting point while developing any new adaptor. FSA extracts data from a file system. Data and hierarchy of document remain the same and some attributes which are not present with files can be provided at time of ingestion by using default files.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-mh	-mongo-host	hostname	Yes	Mongo DB Hostname
-mp	--mongo-port	port number	Yes	Mongo DB Port number
-md	--mongo-db	db name	Yes	Mongo DB Database name
-mu	--mongo-user	user name	Yes	Mongo DB Username
-mpw	--mongo-password	password	Yes	Mongo DB Password
-d	--directory	file system path	Yes	Root directory to ingest
-s	--settings	settings file	Yes	Settings file for ingestion
-dr	--destination-root	folder path	No	Root destination folder in the repository
-ir	--include-fsroot	true|false	No	Include root folder from the specified folder you are importing
-ms	--migrate-security	true|false	No	Translate file/folder security to Documentum ACLs. Default: false
-dt	--document-type		No	Default is dm_document
-ft	--folder-type			Default is dm_folder
-h	--help		No	Show this text

Importing (Eclipse)

Step 1	Unzip FileSystemAdaptor.zip present in the samples directory.
Step 2	Open eclipse and go to File -> Import.
Step 3	From import window go to General -> Existing Projects into Workspace and press next.
Step 4	In root directory select the path to FileSystemAdaptor.
Step 5	Press finish.
Step 6	Right-Click on project folder and choose properties. Go to java Build Path present in menu on left hand side. Choose Libraries tab from top and click on add external JARs. Add EMA-API-1.6.0.jar and EMAIngestManager-1.6.0.jar and your project is all set to run.

Scenario

Providing properties file for parameters

Options can be added to a properties file instead of providing them on the command line.In case an option is provided both in properties file as well as in command line, the command line option value will override the

In eclipse VM argument :
-Doptions.default="E:/ema/FileSystemAdaptorProperties.properties"

FileSystemAdaptorProperties.properties

```
mongo-host=localhost
mongo-port=27017
mongo-db=Extractor
mongo-user=admin
mongo-password=Thom2807
directory=C:\\temp\\Test
setting=C:\\demo\\fsa.properties
destination-root=path to set
document_type=dm_document
folder_type=dm_folder
```

	value provided in the properties file.	
	Program arguments	*--mongo-host "localhost" --mongo-port 27017 --mongo-db "Extractor" --mongo-user "admin" --mongo-password "Thom2807" --directory "C:/temp" --settings "C:/temp /fsa.properties" --destination-root "/temp" --document-type "dm_document" --folder-type "dm_folder"*

CLONER

Definition Used when there is a database change from Oracle/DB2 to SQL Server.
It copies all tables, handles aspect property data and all object IDs, workflow states, remain intact
It does not:

- Upgrade schema to new version
- Content needs to be copied separately
- Install aspects in the target
- Migrate to different repository version. (Repository versions must be exactly the same)
- Create custom indexes in the target. They need to be reapplied after cloning

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-sd	--source-driver	Driver	Yes	Source JDBC driver to use for connection
-sc	--source-connection	Connection string	Yes	Source JDBC connection string
-su	--source-user	Username	Yes	Source JDBC username
-sp	--source-password	password	Yes	Source JDBC password
-td	--target-driver	Driver	Yes	Target JDBC driver to use for connection
-tc	--target-connection	Connection	Yes	Target JDBC connection string
-tu	--target-user	Username	Yes	Target JDBC username
-tp	--target-password	Password	Yes	Target JDBC password
-stp	--source-table-prefix	Source Prefix	Yes	Source table prefix. For Oracle the value is like “SchemaName.”
-ttp	--target-table-prefix	Target Prefix	Yes	Target table prefix. For SQL server the value is like “dm_databaseName_docbase.dbo.”
-cv	--create-vstamp		No	Create the entry in the vstamp table needed by SQL Server
-iv	--invalidate-views		No	Invalidate views in target
-rdo	--rename-dar-objects		No	Change dmc_dar object names to avoid problems updating DARs with Aspect types
-xa	--exclude-audit		No	Exclude dm_audit* tables from the cloning process
-ao	--audit-only		No	Clone only the dm_audit* tables
-ta	--truncate-audit		No	Truncate dm_audittrail* tables ? valid only if the --exclude-audit option is used
-aw	--audit-where-clause		No	Where clause applied to migration of dm_audit* tables
-fc	--force-clone		No	Continue cloning even in case of errors
-th	--thread-count	number of threads	No	Number of threads to use (default = 10)
-rck	--reset-crypto-key		No	Reset crypto keys (use if aek.key changed)
-tdo	--truncate-dmi-object		No	Truncate the dmi_object table
-io	--index-only		No	Only realign the indexes in the target database
-to	--test-only		No	Run DB connection and basic clone tests ONLY
-nb	--no-batch		No	Submit single INSERTs instead of batches
-si	--skip-indexes		No	Skip index realignment in the target database
-tm	--table-map	filename	Yes	Write the aspect table map to this file
-ig	--ignore-list	list of tables	No	List of tables to ignore during cloning
-co	--copy-list	list of tables	No	List of tables to clone, ignore everything else!!
-de	--default-list	override file	No	Override file to override certain columns
-bl	--batch-limit	rows per batch	No	Number of rows in a batch, default 10000
-cl	--commit-limit	batches per commit	No	Number of batches before a commmit, default 10000

	-rf	--report-file		No	Name of the report file to be generated
	-st	--segment-tables	list of tables	No	List of tables to split into multiple tasks during cloning
	-dr	--dry-run		No	Run in dry run mode without committing any changes
	-ibv	--ignore-basic-validation		No	No validation tests are run
	-h	--help		No	Show this text

Scenario	**Help**	*java -cp "";"C:\EMA\EMA1.6.0\EMAClone-1.6.0.jar";"C:\EMA\EMA1.6.0\dependency-jars*";"C:\EMA\EMA1.6.0\EMAIngestManager-1.6.0.jar";"C:\Documentum\config" com.emc.monorail.clone.Cloner --help*
	Basic Cloning (Oracle to SQL Server)	*java -cp "";"C:\EMA\EMA1.6.0\EMAClone-1.6.0.jar";"C:\EMA\EMA1.6.0\dependency-jars*";"C:\EMA\EMA1.6.0\EMAIngestManager-1.6.0.jar";"C:\Documentum\config" com.emc.monorail.clone.Cloner -sd "oracle.jdbc.driver.OracleDriver" -sc "jdbc:oracle:thin:@DCTMORA:1521:ORCL" -su "clone" -sp "password" -td "com.microsoft.sqlserver.jdbc.SQLServerDriver" -tc "jdbc:sqlserver://DCTMSQL;databaseName=DM_clone_docbase" -tu "clone" -tp "password" -stp "CLONE." -ttp "DM_clone_docbase.*

How to Clone	**Step 1**	Create target repository. We do not need to use the same aek.key file as the cryptographic keys will be reset later in the process in any case (**works for 6.7 SP2 and earlier only**). The target repository should have the same name and docbase ID as the source repository.
	Step 2	Create types, registered tables etc. in the target repository. This can be achieved by DAR files, scripts etc. In order for EMA to clone registered table data, the table structure should be in target.
	Step 3	Modify index dm_message_archive_s.d_1f00468380000518 to be non-unique (use SQL Server Management Console to do this). Leaving the index as unique causes a duplicate error when cloning.
	Step 4	Run the cloner tool. A sample batch file "*Clone.bat*" is provided in the samples directory. This will produce a report summary of the cloning process in a file "clone_report.txt" (see example in Appendix1, with notes explaining errors and discrepancies.
	Step 5	Reset cryptography keys on target repository. Connect to 10.64.218.82, open SQL Server Management Console, open the resetCryptoKeys.sql script (in Administrator's My Documents folder), and run the script
	Step 6	Continue with post-clone steps

Appendix 1: Sample clone report with manually added notes (in green). Check "*EMAClone-Report.pdf*" in samples folder.

Appendix 2: Overrides.properties file with manual added notes (in green):

```
/* format is <tablename>.<column>.<action>=newvalue
   action=ifnull:            replace with newvalue if the column value is null, or "", or " "
   action=default:           replace with newvalue regardless of current value of the column
   action=regexp(<exp>):     if value matches exp, then replace with newvalue
   */
dm_format_s.default_storage.ifnull=0000000000000000
dm_acl_s.r_template_id.ifnull=0000000000000000
dm_acl_s.r_alias_set_id.ifnull=0000000000000000
dm_sysobject_s.r_alias_set_id.ifnull=0000000000000000
dm_sysobject_s.r_policy_id.ifnull=0000000000000000
dm_group_s.group_directory_id.ifnull=0000000000000000
dmi_expr_code_s.parent_id.ifnull=0000000000000000
dm_server_config_s.ldap_config_id.ifnull=0000000000000000
dm_server_config_s.alias_set_id.ifnull=0000000000000000
dm_policy_s.app_validation_id.ifnull=0000000000000000
dmi_registry_s.policy_id.ifnull=0000000000000000
dm_process_s.calendar_id.ifnull=0000000000000000
dm_process_s.sd_element_default_acl.ifnull=0000000000000000
dm_workflow_s.parent_id.ifnull=0000000000000000
```

```
class.override=com.emc.monorail.clone.db.override.OverrideDefinition
class.ifnull=com.emc.monorail.clone.db.override.IfNullDefinition
class.regexp=com.emc.monorail.clone.db.override.RegExpDefinition
```

Appendix 3: resetCryptoKeys script to reset cryptographic keys to use a new aek.key file:

```
UPDATE dm_docbase_config_s               SET i_crypto_key        ='';
UPDATE dm_docbase_config_s               SET i_ticket_crypto_key ='';

DELETE FROM dmi_object_type              WHERE r_object_id in (SELECT r_object_id FROM dmi_vstamp_s WHERE
i_application = 'dm_docbase_config_crypto_key_init');
DELETE FROM dmi_vstamp_s    WHERE r_object_id in (SELECT r_object_id FROM dmi_vstamp_s WHERE i_application =
'dm_docbase_config_crypto_key_init');
DELETE FROM dmi_object_type              WHERE r_object_id in (SELECT r_object_id FROM dmi_vstamp_s WHERE
i_application = 'dm_docbase_config_ticket_crypto_key_init');
DELETE FROM dmi_vstamp_s    WHERE r_object_id in (SELECT r_object_id FROM dmi_vstamp_s WHERE i_application =
'dm_docbase_config_ticket_crypto_key_init');

DELETE FROM dm_sysobject_s                WHERE r_object_id in (SELECT r_object_id FROM
dm_public_key_certificate_s WHERE key_type=1);
DELETE FROM dm_sysobject_r WHERE r_object_id in (SELECT r_object_id FROM dm_public_key_certificate_s WHERE
key_type=1);

DELETE FROM dm_public_key_certificate_s WHERE key_type=1;

DELETE FROM dm_sysobject_s                WHERE r_object_id in (SELECT r_object_id FROM dm_cryptographic_key_s
WHERE key_type=1);
DELETE FROM dm_sysobject_r WHERE r_object_id in (SELECT r_object_id FROM dm_cryptographic_key_s WHERE
key_type=1);

DELETE FROM dm_cryptographic_key_s      WHERE key_type=1;
```

Morph

Definition Used to do mass object type changes within a repository. Object ids remain the same, inflight workflow retain their state and audit trail stays intact.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-rn	--repository-name	repo name	Yes	Name of target repository
-ru	--repository-username	username	Yes	Repository username
-rp	--repository-password	password	Yes	Repository password
-td	--target-driver	driver class	Yes	Target JDBC driver to use for connection
-tc	--target-connection	connect string	Yes	Target JDBC connection string
-tu	--target-user	username	Yes	Target JDBC username
-tp	--target-password	password	Yes	Target JDBC password
-mh	--mongo-host	hostname	Yes	Mongo DB Hostname
-mp	--mongo-port	port number	Yes	Mongo DB Port number
-md	--mongo-db	db name	Yes	Mongo DB Database name
-mu	--mongo-username	username	Yes	Mongo DB Username
-mpw	--mongo-password	password	Yes	Mongo DB Password
-ot	--old-type	old object type	Yes	Old Object Typese
-nt	--new-type	new object type	Yes	New Object Typese
-wh	--where-clause	DQL where clause	No	Where clause to identify documents to morph
-dl	--detach-lifecycle		No	Detach selected objects from their lifecycle
-da	--detach-aspects	Aspect list	No	Detach selected aspects (or ALL) from the objects
-mr	--merge-relations	Relation list	No	List of relation types whose metadata should be merged in
-tf	--transform-file	Transform file	No	Properties file defining property transformations
-dr	--dry-run		No	Dry-run extract/transform, no DB changes
-nb	--no-batch		No	Run ingestion in no-batching mode, good for debugging issues
-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Morph –help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/MorphProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Morph*

MorphProperties.properties

```
repository-name=test15
repository-username=Administrator
repository-password=password
target-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
target-
connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test15_
docbase
target-user=sa
target-password=password@123
mongo-host=localhost
mongo-port=27017
mongo-db=ExtractorDB
```

	mongo-user=admin mongo-password=Thom2807 old-type=old_type new-type=new_type
Running	*java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.Morph -rn Test15 -ru Administrator -rp Thom2807 -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu sa -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet --transform-files "C:/EMA/EMA 1.6.0/sample files/transformSample.properties" -ot old_type -nt new_type*

FileCopier

Definition

For content copy when the content has not been already copied before Ingestion is run, FileCopier is used. It picks up the FileList that is generated by the Ingestor and runs a multi-threaded copying process. In case there are issues copying some files a retry file list is also generated with these files information.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-sp	--source-prefix	src-prefix	No	Prefix to source file names
-dp	--dest-prefix	dst-prefix	No	Prefix ro dest file names
-bs	--buffer-size	buffer-size	No	Stream buffer size
-tc	--thread-count	num threads	No	Number of threads to use for filecopy
-th	--threshold	num	No	Max size to use channels
-os	--os-threshold	num	No	Max size to use bytestream
-sc	--os-script	filename	No	OS script for file copy
-ov	--overwrite		No	By default this is true
-l	--list	filename	Yes	File name having the source and destination file
-s	--separator	seperator	No	Required if the separator is other then pipe |
-no	--no-copy		No	Files are not copied. Used when you want to do a checksum check after files are copied
-chk	--checksum	option	No	Do checksum depending on the option. Option can take vales all, numpc or number e.g. 10pc will check checksum for 10 percent of the documents
-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.FileCopier --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/FileCopierProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.FileCopier*

FileCopierProperties.properties

```
source-prefix=E:\\Documentum\\data\\cloneme\\content_storage_01\\00002329\\
dest-prefix=E:\\Documentum\\data\\test9\\content_storage_01\\00012cd1\\
thread-count=2
list=C:\\ema\\logs\\file_copy_list.log
```

Runing without checksum check

java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.FileCopier --thread-count 4 --list "E:/ema/logs/file_copy_list.log" -sp "E:/Documentum/data/cloneme/content_storage_01/00002329/" -dp "E:/Documentum/data/test9/content_storage_01/00012cd1/"*

Runing with checksum check for all documents

java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.FileCopier --thread-count 4 --list "E:/ema/logs/file_copy_list.log" -sp "E:/Documentum/data/cloneme/content_storage_01/00002329/" -dp "E:/Documentum/data/test9/content_storage_01/00012cd1/" -chk "all"*

	 "all" is the default if no value is provided for checksum option. Other values it can take are: Percentage: To select a percentage of documents to be checked use something like -chk "30pc".In this case 30 percent of the documents would be checked. Number: To select the number of documents to be checked use something like -chk "30" .In this case 30 documents would be checked.
Only checksum check with no file copy  In case you want to run checksum check after you have copied the file using file copier.	*java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.FileCopier --thread-count 4 --list "E:/ema/logs/file_copy_list.log" -sp "E:/Documentum/data/cloneme/content_storage_01/00002329/" -dp "E:/Documentum/data/test9/content_storage_01/00012cd1/" --no-copy -chk "all"*

Replatform

Definition Update hostname entries in configurations settings stored inside the repository. Can also be used to modify configurations when moving from Unix to Windows and vice versa.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-d	--driver	driver class	Yes	JDBC driver to use for connection
-c	--connection	connect string	Yes	JDBC connection string
-u	--user	username	Yes	JDBC username
-p	--password		Yes	JDBC password
-sh	--source-home	home-dir	No	Source DM_HOME environment variable value
-th	--target-home	home-dir	No	Target DM_HOME environment variable value
-sd	--source-db		No	Source Database Name
-td	--target-db		No	Target Database Name
-so	--source-os	os	Yes	Source Operating System (<os> = 'windows' or 'unix')
-to	--target-os	os	Yes	Target Operating System (<os> = 'windows' or 'unix')
-shn	--source-hostnames	list	Yes	Source Hostnames (<list> = comma separated list)
-thn	--target-hostnames	list	Yes	Target Hostnames (<list> = comma separated list)
-pr	--preview		No	Preview mode (run script w/o committing)
-co	--commit		No	Commit mode (run and commit)
-sc	--script	filename	No	Script mode (run script format)
-sf	--script-format		No	Script format <lang> = 'plsql' or 'sql'
-sfi	--script-file		No	Script file to open
-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Replatform --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/ReplatformProperties.properties" -cp "EMATools-1.6.0.jar; EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Replatform*

ReplatformProperties.properties

```
driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test15_docbase
user=sa
password=password@123
source-os=windows
target-os=windows
source-hostnames=HOSTOLD
target-hostname=HOSTNEW
source-home=Test2
target-home=Test15
```

Running

java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/ ;C:/Documentum/config" com.emc.monorail.configtools.Replatform --driver "oracle.jdbc.OracleDriver" --connection "jdbc:oracle:thin:10.8.50.39:1521" --user user --password "Thom2807" --source-hostnames HOSTOLD --target-hostnames HOSTNEW --script -sf "plsql" -sfi "C:/Temp/run.sql" --source-os windows --target-os windows*

Folder Structure Generator

Definition Used to generate a folder structure based on a simple text file containing a list of folders. A sample file "*folderListSample.txt*" is provided in the samples directory.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-mh	-mongo-host	Host	Yes	Mongo DB Hostname
-mp	--mongo-port	Port number	Yes	Mongo DB Port number
-md	--mongo-db	Database name	Yes	Mongo DB Database name
-mu	--mongo-user	User name	Yes	Mongo DB Username
-mpw	--mongo-password	Password	Yes	Mongo DB Password
-f	--filename	Filename	Yes	Folder list filename
-p	--prefix	Prefix	No	Folder name prefix
-h	--help		No	Show this text

Here's a sample of the folder list file format. The first header line "folderpath" has to be there, and is case sensitive.

```
folderpath
/esdms/Solicitations/Solicitation 1/tab 2
/esdms/Solicitations/Solicitation 1/tab 8
```

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.FolderStructureGenerator --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/FolderStructureGeneratorProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.FolderStructureGenerator*

FolderStructureGeneratorProperties.properties

```
mongo-host=localhost
mongo-port=27017
mongo-db=ExtractorDB
mongo-user=admin
mongo-password=Thom2807
filename=C:\\temp\\folderListSample.txt
```

Running

java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.FolderStructureGenerator -mh "localhost" -mp 27017 -md "ExtractDB" -mu "admin" -mpw "Thom2807" --filename "c:/temp/folderListSample.txt"*

Link Count Update

Definition Update the link count of a folder. Required when there is a transformation which moves documents to different folders. It is not required when documents are not moved to new folders as part of transformation or if D2-Core job will be used to apply auto-linking rules.

Parameters	**Short Option**	**Long Option**	**Argument**	**Mandatory**	**Description**
	-td	--target-driver	Driver	Yes	Target JDBC driver to use for connection
	-tc	--target-connection	Connection String	Yes	Target JDBC connection string
	-tu	--target-user	Username	Yes	Target JDBC username
	-tp	--target-password	Password	Yes	Target JDBC password
	-os	--output-script	Script	No	Filename to use for script output
	-of	--output-format	Format	No	Format to use for output format. By default Oracle is used.
	-wc	--where-clause	Sql clause	No	Restrict the folders to be reprocessed
	-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.LinkCountUpdateTool --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/LinkCountUpdateProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.LinkCountUpdate*

LinkCountUpdateProperties.properties

```
target-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
target-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test15_docbase
target-user=sa
target-password=password@123
output-script=C:\\temp\\test.sql
output-format=oracle
```

Running

java -cp "C:/EMA/EMA1.6.0/EMATools-1.6.0.jar;C:/EMA/EMA1.6.0/EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/, C:/Documentum/config" com.emc.monorail.tools.LinkCountUpdateTool --target-driver "oracle.jdbc.OracleDriver"--target-connection "jdbc:oracle:thin:@localhost:1521:ORCL" --target-user "target"--target-password "password" --output-script "C:/Temp/updateFolders.sql" --output-format "oracle"*

DataVerifier

Definition Used to test the compatibility of a MongoDB database with the target DB schema into which it is intended to be ingested. This is a quicker and more efficient approach than running dry-runs until all INSERTs pass.

Checks

- String values do not exceed the length of the field in the DB
- Fixed length string fields (such as IDs) match the length of the values in Mongo
- Boolean values are 0 or 1
- Numeric fields do not contain text
- Required fields are present (not NULL). If the value is missing then it checks the default file for the particular attribute value. In case the value for the particular attribute is missing there too it will throw an error specifying that the value is null or not present.
- custom types and filestores are present in the target system.

Parameters	Short Option	Long Option	Argument	Mandatory	Description
	-rn	--repository-name	repo name	Yes	Name of target repository
	-ru	--repository-username	username	Yes	Repository username
	-rp	--repository-password	password	Yes	Repository password
	-td	--target-driver	driver class	Yes	Target JDBC driver to use for connection
	-tc	--target-connection	connect string	Yes	Target JDBC connection string
	-tu	--target-user	username	Yes	Target JDBC username
	-tp	--target-password	password	Yes	Target JDBC password
	-mh	--mongo-host	hostname	Yes	Mongo DB Hostname
	-mp	--mongo-port	port number	Yes	Mongo DB Port number
	-md	--mongo-db	db name	Yes	Mongo DB Database name
	-mu	--mongo-user	mongo user	Yes	Mongo Admin username
	-mpa	--mongo-password	mongo password	Yes	Mongo Admin password
	-t	--type		No	Object type to verify New If not provided then Data verifier will run for all types
	-a	--attributes		No	New To verify a particular attribute.
	-iv	--ignore-verifiy		No	If user want to ignore verify
	-v	--verbose		No	Show all erroneous data rows
	-sm	--storage-map	storage-map	No	Storage map, include to check filestores.
	-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.DataVerifier --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties

java -Doptions.default="E:/ema/DataVerifierProperties.properties" -cp "E:/Documentum/config;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.DataVerifier*

DataVerifierProperties.properties

```
repository-name=test15
repository-username=Administrator
repository-password=password
target-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
target-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test15_docbase
target-user=sa
```

file.	target-password=password@123 mongo-host=localhost mongo-port=27017 mongo-db=ExtractorDB mongo-user=admin mongo-password=Thom2807 type=dm_type
Running	*java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;DefaultFiles;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.DataVerifier -rn test1 -ru Administrator -rp Thom2807 -td "com.microsoft.sqlserver.jdbc.SQLServerDriver" -tc "jdbc:sqlserver://win2008x64base:1433;databaseName=DM_test1_docbase" -tu user -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract -t "type_verify" –v*
Running for specific Attribute	*java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;DefaultFiles;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.DataVerifier -rn test1 -ru Administrator -rp Thom2807 -td "com.microsoft.sqlserver.jdbc.SQLServerDriver" -tc "jdbc:sqlserver://win2008x64base:1433;databaseName=DM_test1_docbase" -tu user -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract -t "type_verify" -a "owner_name" -v*
Running for all the types	*java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;DefaultFiles;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.DataVerifier -rn test1 -ru Administrator -rp Thom2807 -td "com.microsoft.sqlserver.jdbc.SQLServerDriver" -tc "jdbc:sqlserver://win2008x64base:1433;databaseName=DM_test1_docbase" -tu user -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract -v*

Compare

Definition

Used to compare an object pre/post-transformation. New properties, deleted properties, and modified properties will be displayed in the output with before/after values.

Users can do comparison of objects by providing a percentage or a number. Random objects are chosen and compared to verify the correctness. Earlier one had to provide IDs or an id list for the comparison.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-mh	-mongo-host	Host	Yes	Mongo DB Hostname
-mp	--mongo-port	Port	Yes	Mongo DB Port number
-md	--mongo-db	Database	Yes	Mongo DB Database name
-mu	--mongo-user	Username	Yes	Mongo DB Username
-mpw	--mongo-password	Password	Yes	Mongo DB Password
-o	--object-id	ID	No	Object ID to compare. Mandatory if --file or -rc is not provided
-ot	--orig-type	Type name	No	Original object type. Mandatory if --file or is not provided
-nt	--new-type	Type name	No	New object type. Mandatory if --file is not provided
-bt	--base-type	Type name	No	Base type: document, folder, or group
-f	--file	Filepath	No	File with a list of object ids to be compared
-rc	--random-check	Option	No	Option can take values numpc or number. E.g. 10p will compare 10% of objects.
-h	--help		No	Show this text

Use one of -f, -o or -rc at a time only.
Sample file "*idlist.txt*" has been added to the samples folder.

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Compare --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/CompareProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Compare*

CompareProperties.properties

```
mongo-host=localhost
mongo-port=27017
mongo-db=ExtractorDB
mongo-user=admin
mongo-password=Thom2807
orig-type=old_type
new-type=new_type
object-id=09XXXX
```

Compare only a particular object id

java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Compare -mh localhost -mp 27017 -md Extract_Cabinet -mu admin -mpw Thom2807 -bt document -ot old_type -nt new_type -o 09XXXX*

Compare number of random objects

-rc "30pc" will compare 30 percent of total objects of that type.

java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.Compare -mh localhost -mp 27017 -md Extract_Cabinet -mu admin -mpw Thom2807 -bt document -ot old_type -nt new_type -rc "30"*

Providing idlist	*java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.Compare -mh localhost -mp 27017 -md Extract_Cabinet -mu admin -mpw Thom2807 -bt document -f "C:\EMA\EMA1.6.0\samples\idlist.txt"*

Default File Creator

Definition Used to generate default files (required during Ingestion) for the types specified.

r_page_cnt has the value of 0 generated by default. Change it to 1 for the types with content.

Parameters	**Short Option**	**Long Option**	**Argument**	**Mandatory**	**Description**
	-u	--user	username	Yes	Username
	-r	--repository	Repository name	Yes	Repository
	-p	--password	password	Yes	Password
	-d	--domain	Domain	No	Domain name for repository connection
	-a	--all		No	Extract all type
	-l	--list		No	Extract these type only
	-xl	--exclude-list		No	Extract all types except these
	-e	--expression		No	Extract types matching a pattern
	-xe	--exclude-expression		No	Extract types that do not match a pattern
	-f	--file		No	Extract types listed in a file
	-xf	--exclude-file		No	Exclude types listed in a file
	-dp	--defaults-path	path	No	Default path for file where default file will be generated (C:/temp/DefaultsFiles)
	-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.DefaultFileCreator --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/DefaultFileCreatorProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/;* E:/Documentum/config" *com.emc.monorail.tools.DefaultFileCreator*

DefaultFileCreatorProperties.properties

```
user=Administrator
password=password
repository=cloneme
list=type1,type2,type3
defaults-path=C://EMA1.6.0//DefaultsFiles
```

Generate for all types

java -cp *"E:/Documentum/config;.;EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*"* com.emc.monorail.tools.DefaultFileCreator --repository Test15 --user dmadmin --password Thom2807 --all

Generate for only types specified

java -cp "E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar*;C:/EMA/EMA1.6.0/dependency-jars/**"* com.emc.monorail.tools.DefaultFileCreator --repository Test15 --user dmadmin --password Thom2807 --list "type1,type2,type3"

Exclude generate for types specified

java -cp *"E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*"* com.emc.monorail.tools.DefaultFileCreator --repository Test15 --user dmadmin --password Thom2807 --exclude-list "type1,type2,type3"

Generate types using an

java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-

expression	*1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.DefaultFileCreator --repository Test15 --user dmadmin --password Thom2807 --expression dm_xml*
Generate for types provided in file	*java -cp "E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.DefaultFileCreator --repository Test15 --user dmadmin --password Thom2807 --file "C:\temp\abc.txt"*  Here's a sample of the file format to extract types provided in file. Write types in different lines. dm_acl dm_folder dm_document

Type Extractor

Definition Used to extract the types present in the source system. It generates a dql file which can be run in the target system to create the corresponding types.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-u	--user	Username	Yes	Source repository username
-r	--repository	Repository name	Yes	Source repository name
-p	--password	password	Yes	Source repository password
-d	--domain	domain	No	Domain Name
-a	--all		No	Extract all type aspects
-l	--list		No	Extract these type only
-xl	--exclude-list		No	Extract all types except these
-e	--expression		No	Extract types matching a pattern
-xe	--exclude-expression		No	Extract types that do not match a pattern
-f	--file		No	Extract types listed in a file
-xf	--exclude-file		No	Exclude types listed in a file
-g	--go		No	Append Go for the update query generated. Used for aspects
-h	--help		No	Show this text

Scenario

Help

```
java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.TypeExtractor --help
```

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

```
java -Doptions.default="E:/ema/TypeExtractorProperties.properties"  -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;E:/Documentum/config;" com.emc.monorail.tools.TypeExtractor
```

TypeExtractorProperties.properties

```
user=Administrator
password=password
repository=cloneme
list=type1,type2,type3
domain=windows NT domain name
```

Extract all types

```
java -cp "E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.TypeExtractor --repository Test15 --user dmadmin --password Thom2807 --all --go >> "c:/temp/types.dql"
```

Extract only types specified

```
java -cp "E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.TypeExtractor --repository Test15 --user dmadmin --password Thom2807 --list "type1,type2,type3" --go >> c:/temp/types.dql
```

Exclude extraction of types specified

```
java -cp "E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.TypeExtractor --repository Test15 --user dmadmin --password Thom2807 --exclude-list "type1,type2,type3" --go >> "c:/temp/types.dql"
```

Extract types providing an expression

```
java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.TypeExtractor --repository Test15 --user dmadmin --password
```

	Thom2807 --expression dm_xml *--go >> c:/temp/types.dql*
Extract types provided in file	java -cp *"E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*"* com.emc.monorail.tools.*TypeExtractor* --repository Test15 --user dmadmin --password Thom2807--file *"*C:\temp\abc.txt*"--go >> c:/temp/types.dql*  Here's a sample of the file format to extract types provided in file. Write types in different lines. dm_acl dm_folder dm_document

Audit Trail Extractor

Definition Used to extract the audit trail from the system.

Parameters	**Short Option**	**Long Option**	**Argument**	**Mandatory**	**Description**
	-sd	--source-drive	Driver	Yes	Source JDBC driver to use for connection
	-sc	--source-connection	Connection String	Yes	Source JDBC connection string
	-su	--source-user	Username	Yes	Source JDBC username
	-sp	--source-password	Password	Yes	Source JDBC password
	-mh	--mongo-host	Host	Yes	Mongo DB Hostname
	-mp	--mongo-port	Port	Yes	Mongo DB Port number
	-md	--mongo-db	Database name	Yes	Mongo DB Database name
	-mu	--mongo-user	Username	Yes	Mongo DB Username
	-mpw	--mongo-password	Password	Yes	Mongo DB Password
	-wc	--where-clause	Sql clause	No	SQL where clause identifying sysobjects to extract
	-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.AuditTrailExtractor --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/AuditTrailExtractorProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.AuditTrailExtractor*

AuditTrailExtractorProperties.properties

```
source-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
Source-
connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_d
ocbase
source-user=sa
source-password=password@123
mongo-host=localhost
mongo-port=27017
mongo-db=Extractdb
mongo-user=admin
mongo-password=Thom2807
where-clause=<WHERE CLAUSE>
```

Normal Running

java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.AuditTrailExtractor -sd "com.microsoft.sqlserver.jdbc.SQLServerDriver" -sc "jdbc:sqlserver://win2008x64base:1433;databaseName=DM_test1_docbase" -su "test1" -sp Thom2807 -mh localhost -mp "27017" -md Extract_Cabinet -mu "admin" -mpw "Thom2807" -wc "<WHERE CLAUSE>"*

User Group Extractor

Definition Used to extract the users and groups in a particular system.

Parameters	Short Option	Long Option	Argument	Mandatory	Description
	-sd	--source-drive	driver	Yes	Source JDBC driver to use for connection
	-sc	--source-connection	Connection string	Yes	Source JDBC connection string
	-su	--source-user	Username	Yes	Source JDBC username
	-sp	--source-password	Password	Yes	Source JDBC password
	-mh	--mongo-host	Host	Yes	Mongo DB Hostname
	-mp	--mongo-port	Port	Yes	Mongo DB Port number
	-md	--mongo-db	Database name	Yes	Mongo DB Database name
	-mu	--mongo-user	username	Yes	Mongo DB Username
	-mpw	--mongo-password	Password	Yes	Mongo DB Password
	-uwc	--user-where-clause	Sql clause	No	SQL user where clause identifying sysobjects to extract
	-gwc	--group-where-clause	Sql clause	No	SQL group where clause identifying sysobjects to extract
	-h	--help		No	Show this text

Provide at least one of user where clause or group where clause.

Scenario

Help

```
java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.GroupUserExtractor --help
```

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

```
java -Doptions.default="E:/ema/GroupUserExtractorProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.GroupUserExtractor
```

GroupUserExtractorProperties.properties

```
source-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
Source-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase
source-user=test1
source-password=Thom2807
mongo-host=localhost
mongo-port=27017
mongo-db=Extractdb
mongo-user=admin
mongo-password=Thom2807
user-where-clause=owner_name like 'dmadmin'
group-where-clause=group_name like 'docu'
```

Extract both users and groups

```
java -cp "E:/Documentum/config;.EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.GroupUserExtractor -sd "com.microsoft.sqlserver.jdbc.SQLServerDriver" -sc "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test1_docbase" -su test1 -sp Thom2807 -mh localhost -mp 27017 -mu admin -mpw Thom2807 -md user_group_ext -uwc "owner_name like 'admin'" -gwc "group_name like 'docu'"
```

Extract users depending on a clause

```
java -cp "E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*"
```

	com.emc.monorail.tools.GroupUserExtractor -sd "com.microsoft.sqlserver.jdbc.SQLServerDriver" -sc "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test1_docbase" -su " test1" -sp Thom2807 -mh localhost -mp 27017 -md Extract_Cabinet -mu admin -mpw Thom2807 -md user_group_ext -uwc "owner_name like 'dmadmin'"
Extract groups depending on a clause	*java -cp "E:/Documentum/config;.;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.GroupUserExtractor -sd "com.microsoft.sqlserver.jdbc.SQLServerDriver" -sc "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test1_docbase" -su test1 -sp Thom2807 -mh localhost -mp 27017 -md Extract_Cabinet -mu admin -mpw Thom2807 -md user_group_ext -gwc "group_name like 'docu'"*

ACLExtractor

Definition Used to extract the ACLs from the system.

Parameters	**Short Option**	**Long Option**	**Argument**	**Mandatory**	**Description**
	-sd	--source-drive	Driver	Yes	Source JDBC driver to use for connection
	-sc	--source-connection	Connection	Yes	Source JDBC connection string
	-su	--source-user	Username	Yes	Source JDBC username
	-sp	--source-password	Password	Yes	Source JDBC password
	-mh	--mongo-host	Host	Yes	Mongo DB Hostname
	-mp	--mongo-port	Port	Yes	Mongo DB Port number
	-md	--mongo-db	Database name	Yes	Mongo DB Database name
	-mu	--mongo-user	Username	Yes	Mongo DB Username
	-mpw	--mongo-password	Password	Yes	Mongo DB Password
	-wc	--where-clause	Sql clause	No	SQL where clause identifying ACLs to extract
	-h	--help		No	Show this text

Scenario

Help

java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ACLExtractor --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/ACLExtractorProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ACLExtractor*

ACLExtractorProperties.properties

```
source-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
Source-
connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase
source-user=sa
source-password=password@123
mongo-host=localhost
mongo-port=27017
mongo-db=Extractdb
mongo-user=admin
mongo-password=Thom2807
where-clause=<WHERE CLAUSE>
```

Running

java -cp "E:/Documentum/config;.;E:\EMA;EMAIngestManager-1.6.0.jar;EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ACLExtractor -sd "com.microsoft.sqlserver.jdbc.SQLServerDriver" -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_cloneme_docbase -su sa -sp password@123 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md "acl_extract" -wc "<WHERE CLAUSE>"*

ExtractFileList

Definition Generates a file copy list from the MongoDB database that we get after extraction. Used when a local data copy is to be done. FileCopier is run for content copy after running this tool.
Scenario: In case migration is happening in different geographies and the content size is huge there might be a requirement that the data disk (with only the content files which are part of the migration) needs to be shipped.

Parameters	**Short Option**	**Long Option**	**Argument**	**Mandatory**	**Description**
	-sd	--source-drive	Driver	Yes	Source JDBC driver to use for connection
	-sc	--source-connection	Connection	Yes	Source JDBC connection string
	-su	--source-user	Username	Yes	Source JDBC username
	-sp	--source-password	Password	Yes	Source JDBC password
	-mh	--mongo-host	Host	Yes	Mongo DB Hostname
	-mp	--mongo-port	Port	Yes	Mongo DB Port number
	-md	--mongo-db	Database name	Yes	Mongo DB Database name
	-mu	--mongo-user	Username	Yes	Mongo DB Username
	-mpw	--mongo-password	Password	Yes	Mongo DB Password
	-sm	--storage-map	mapfile	Yes	Storage map if copying content
	-fc	--filecopy-list	file	No	FileCopy list that will be generated
	-h	--help		No	Show this text

Scenario

Help

```
java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.ExtractFileList --help
```

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

```
java -Doptions.default="E:/ema/AuditTrailExtractorProperties.properties"  -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.ExtractFileList
```

ExtractFileList Properties.properties

```
source-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
Source-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase
source-user=sa
source-password=password@123
mongo-host=localhost
mongo-port=27017
mongo-db=Extractdb
mongo-user=admin
mongo-password=Thom2807
storage-map=C:\EMA\EMA1.6.0\samples\changeFilestore.storagemap.properties
```

Running

```
java -cp "EMA-API-1.6.0.jar;EMATools-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.ExtractFileList -sd "com.microsoft.sqlserver.jdbc.SQLServerDriver" -sc "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase" -su "sa" -sp "Thom2807" -mh "127.0.0.1" -mu "admin" -mp "27017"  -mpa "Thom2807" -md "Extract_Cabinet" -sm "C:/EMA/EMA1.6.0/samples/extract-storagemap.properties" -fl "C:/EMA/extractFileCopyList.txt"
```

Encrypt Utils

Definition Used to encrypt passwords and can be used in all the EMA components. EMA components can now decrypt the passwords encrypted by this tool. Can be used for mongo, repository and database passwords.

Parameters	**Short Option**	**Long Option**	**Argument**	**Mandatory**	**Description**
	-ep	--encrypt-password	string to encrypt	No	Encrypts the string provided
	-dp	--decrypt-password	encrypted string	No	Decrypts the encrypted string
	-h	--help		No	Show this text
Scenario	**Help**		*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.EncryptUtils --help*		
	Encryption of Password		*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.EncryptUtils -ep Thom2807*		
	Decryption of Password		*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.EncryptUtils -dp DM_ENCR=B8yMJvZwYLKy8bEG1zZ8AQ==*		

Content Migrator  New

Definition This tool was developed to help moving data from Centera to Isilon. It requires the Centera metadata and content be extracted in CSV file and filesystem respectively.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-td	--target-driver	driver class	Yes	Target JDBC driver to use for connection
-tc	--target-connection	connect string	Yes	Target JDBC connection string
-tu	--target-user	username	Yes	Target docbase owner username
-tp	--target-password	password	Yes	Target Docbase owner password
-fs	--filstore	filestore	Yes	Target filestore name.
-fn	--filename	filepath	Yes	Name of csv files. If there are multiple files use | to separate.
-s	--separator	csv separator	Yes	Separator used in csv file to separate values
-bs	--batch-size	number	No	Size of a batch. Default Size is 10000.
-mv	--move		No	To move the files instead of doing copy
-sp	--source-prefix		No	Prefix to the source file path
-sc	--source-centera		No	In case Source system is centera
-idc	--id-column		No	Column number of ids in csv (Please consider column number starts from 0)
-pc	--path-column		No	Column number of path in csv (Please consider column number starts from 0
-h	--help		No	Show this text

Scenario

Help

java -Dlog4j.configuration="file:C:\ema\logs\log4j.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ContentMigrator --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

Running

java -Dlog4j.configuration="file:C:\ema\logs\log4j.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ContentMigrator -td "com.microsoft.sqlserver.jdbc.SQLServerDriver" -tc "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_cloneme_docbase" -tu "sa" -tp "password@123" -fs "filestore_01" -fn "C:\new.csv" -s "," -mv -idc "0" -pc "1"*

Reference Update (Beta) 

Definition

For smaller size migrations (typically < 1 million documents), we migrate all objects and relationships etc. in a single batch, and relationships etc. are automatically updated to newly allocated object IDs. In some cases, the size of the data requires us to run multiple batches, so it is not always possible to update all references to object IDs to the new values, as perhaps the referenced object has not yet been migrated. To address this issue we have added this tool, ReferenceUpdate, to act after completing the migration to create a script to update references in the target repository directly, with new object IDs retrieved via lookup class that can be customized. A simple database lookup based on a JDBC query is provided as an example.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-td	--target-driver	driver class	Yes	Target JDBC driver to use for connection
-tc	--target-connection	connect string	Yes	Target JDBC connection string
-tu	--target-user	username	Yes	Target docbase owner username
-tp	--target-password	password	Yes	Target Docbase owner password
-id	--id-lookup	Id lookup filename	Yes	Config file for ID Lookup
-c	--config	config filename	Yes	Configuration file for fields to update
-os	--output-script	filename	Yes	Filename used for script generatedby the tool
-of	--output-format	format name	Yes	By default, Oracle, or "sql".

Scenario

Help

java -Dlog4j.configuration="file:C:\ema\logs\log4j.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.RefUpdateTool --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

ID Lookup Properties file:

```
Classname=<java class to instantiate for ID lookup>

#Fields for the class
com.emc.monorail.repositorytools.idlookups.DatabaseIDLookup

driver=<JDBC driver class>
connection=<JDBC connection string>
username=<JDBC username>
password=<JDBC password>
tablename=<DB table to query>
oldidfield=<DB table column containing old ID>
newidfield=<DB table column containting new ID>
usecache=true|false
precachequery=<Query to optionally preload to the cache>
```

Example:

```
classname=com.emc.monorail.repositorytools.idlookups.DatabaseIDLookup
driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test20_docbase
username=Test20a
password=Thom2807
tablename=dm_sysobject_s
oldidfield=subject
newidfield=r_object_id
usecache=false
```

Config File Properties file:

	field.count=<number of fields to process> field.x=<DB Table name to check>.<DB Table Column to check>
	Example:
	field.count=4 field.0=dm_relation_s.child_id field.1=dm_relation_s.parent_id field.2=dmr_containment_s.parent_id field.3=dmr_containment_s.component_id
Running	*java -Dlog4j.configuration="file:C:\ema\logs\log4j.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*" com.emc.monorail.tools.RefUpdateTool -td "com.microsoft.sqlserver.jdbc.SQLServerDriver" -tc "jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_cloneme_docbase" -tu "sa" -tp "password@123" -id*

Connectivity Checker

Definition This tool checks connectivity and authentication for source/target databases, MongoDB, Content Server & File shares.

Parameters

Short Option	Long Option	Argument	Mandatory	Description
-td	--target-driver	driver class	Yes	Target JDBC driver to use for connection
-tc	--target-connection	connect string	Yes	Target JDBC connection string
-tu	--target-user	username	Yes	Target docbase owner username
-tp	--target-password	password	Yes	Target Docbase owner password
-fs	--filstore	filestore	Yes	Target filestore name.
-fn	--filename	filepath	Yes	Name of csv files. If there are multiple files use | to separate.
-s	--separator	csv separator	Yes	Separator used in csv file to separate values
-bs	--batch-size	number	No	Size of a batch. Default Size is 10000.
-mv	--move		No	To move the files instead of doing copy
-sp	--source-prefix		No	Prefix to the source file path
-sc	--source-centera		No	In case Source system is centera
-h	--help		No	Show this text

Scenario

Help

java -Dlog4j.configuration="file:C:\ema\logs\log4j.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ConnectionChecker --help*

Providing properties file for parameters

You can provide options in properties file instead of command line. In case an option is provided in both, the command line option value will override the value provided in the properties file.

java -Doptions.default="E:/ema/ConnectivityCheckerProperties.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ConnectionChecker*

ConnectivityCheckerProperties.properties

```
source-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
source-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test1_docbase
source-user=sa
source-password=Thom2807
source-filepath=\\INCSSINGHR19L2C\sharetest
target-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
target-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_test2_docbase
target-user=sa
target-password=Thom2807
target-filepath=\\INCSSINGHR19L2C\sharetest
repository-name=test1
repository-username=Admin1
repository-password=Thom2807
mongo-host=localhost
mongo-port=27017
mongo-db=Extractdb
mongo-user=admin
mongo-password=Thom2807
```

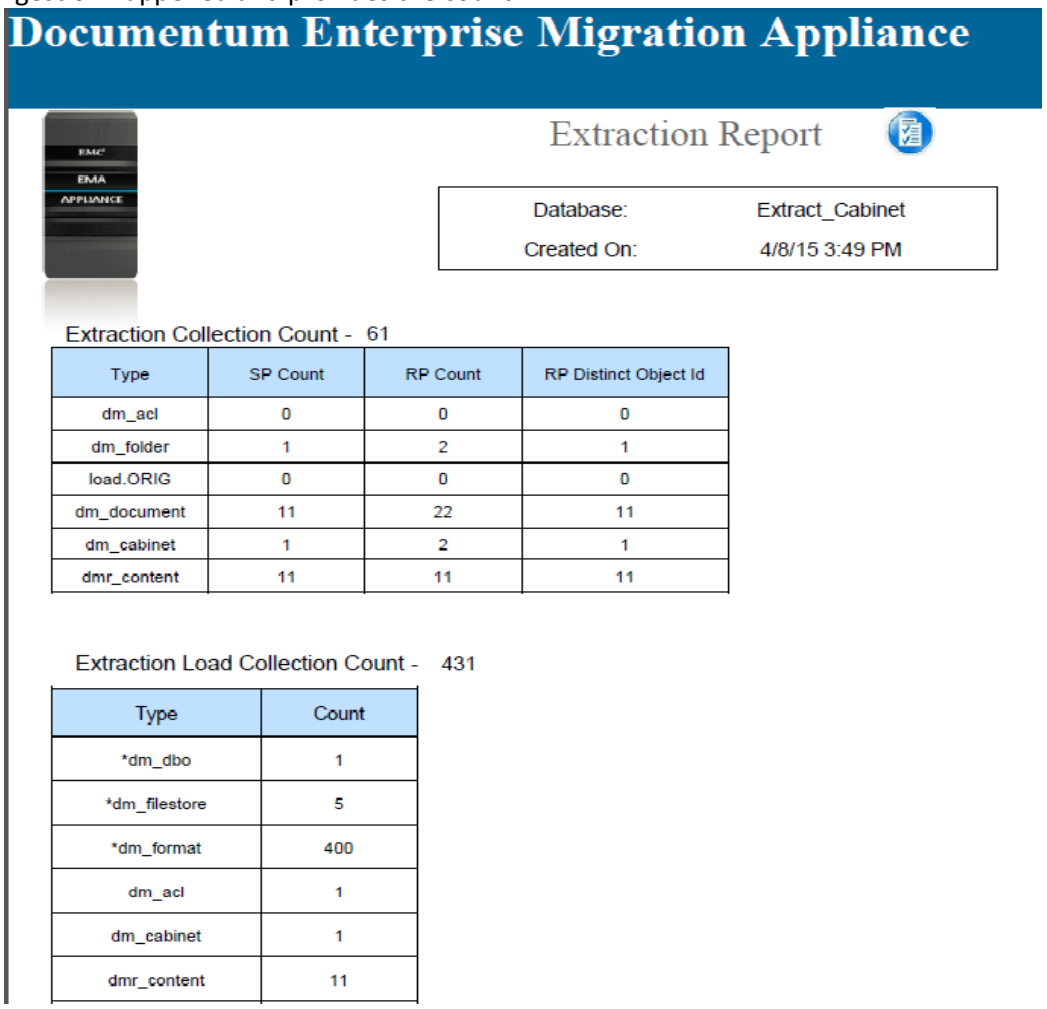
Running

java -Dlog4j.configuration="file:C:\ema\logs\log4j.properties" -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-

	1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/" com.emc.monorail.tools.ConnectionChecker -sd "com.microsoft.sqlserver.jdbc.SQLServerDriver" -sc "jdbc:sqlserver://192.168.147.138:1433;databaseName=DM_test1_docbase" -su "sa" -sp "Thom2807" -td "com.microsoft.sqlserver.jdbc.SQLServerDriver" -tc "jdbc:sqlserver://192.168.147.138:1433;databaseName=DM_test2_docbase" -tu "sa" -tp "Thom2807" -mh "127.0.0.1" -mu "admin" -mp "27017" -mpw "Thom2807" -md "Extract_Cabinet" -rn test1 -ru "Admin1" -rp "Thom2807" -sf "\\INCSSINGHR19L2C\sharetest" -tf "\\INCSSINGHR19L2C\sharetest"*

REPORT AND CONSISTENCY CHECKING

Definition	Used to generate report or consistency checking for Extraction, Transform and Ingestion.
Report Considerations	
General Report	For Extraction & Transform it reads the MongoDB database and provides the counts of all the respective collections. For Ingestion it reads the information from the database where the Ingestion happened and provides the count.
Consistency-Check Report	It checks the numeric count consistency (count of objects for each type) and data consistency (compare the attribute values of the objects). For Extraction it reads the MongoDB database and compares the source database. For Ingestion it reads the information from the mongo database and compares with the target database where the Ingestion happened.



Documentum Enterprise Migration Appliance

Extraction Report

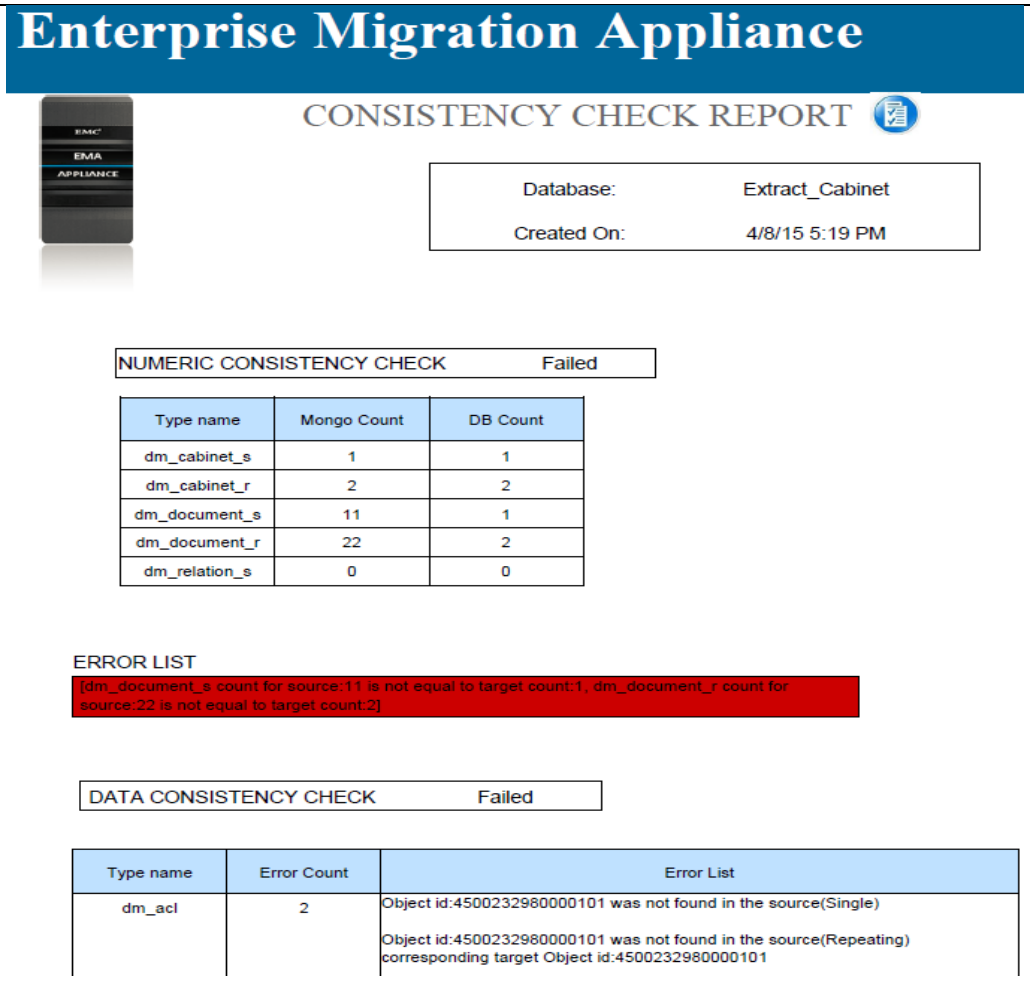
Database:	Extract_Cabinet
Created On:	4/8/15 3:49 PM

Extraction Collection Count - 61

Type	SP Count	RP Count	RP Distinct Object Id
dm_acl	0	0	0
dm_folder	1	2	1
load.ORIG	0	0	0
dm_document	11	22	11
dm_cabinet	1	2	1
dmr_content	11	11	11

Extraction Load Collection Count - 431

Type	Count
*dm_dbo	1
*dm_filestore	5
*dm_format	400
dm_acl	1
dm_cabinet	1
dmr_content	11



Enterprise Migration Appliance

CONSISTENCY CHECK REPORT

Database:	Extract_Cabinet
Created On:	4/8/15 5:19 PM

NUMERIC CONSISTENCY CHECK Failed

Type name	Mongo Count	DB Count
dm_cabinet_s	1	1
dm_cabinet_r	2	2
dm_document_s	11	1
dm_document_r	22	2
dm_relation_s	0	0

ERROR LIST

[dm_document_s count for source:11 is not equal to target count:1, dm_document_r count for source:22 is not equal to target count:2]

DATA CONSISTENCY CHECK Failed

Type name	Error Count	Error List
dm_acl	2	Object id:4500232980000101 was not found in the source(Single) Object id:4500232980000101 was not found in the source(Repeating) corresponding target Object id:4500232980000101

Parameters	Short Option	Long Option	Argument	Mandatory	Description
	-sd	--source-driver	driver class	No	Source JDBC driver to use for connection
	-sc	--source-connection	connect string	No	Source JDBC connection string
	-su	--source-user	Username	No	Source JDBC username
	-sp	--source-password	Password	No	Source JDBC password
	-td	--target-driver	Driver	No	Target JDBC driver to use for connection
	-tc	--target-connection	Connection String	No	Target JDBC connection string
	-tu	--target-user	Username	No	Target JDBC username
	-tp	--target-password	Password	No	Target JDBC password
	-mh	--mongo-host	Host	Yes	Mongo DB Hostname
	-mp	--mongo-port	Port	Yes	Mongo DB Port number
	-md	--mongo-db	Database name	Yes	Mongo DB Database name
	-mu	--mongo-user	Username	Yes	Mongo DB User name
	-mpa	--mongo-password	Password	Yes	Mongo DB User password
	-mo	--mode	Option	Yes	Takes value as Extract | Ingest
	-c	--cabinet-name		No	Extracted cabinet name. Multiple cabinets are seperated by pipe|operator
	-wh	--where-clause	Sql clause	No	SQL where clause used during extract
	-x	--exclude		No	Exclude objects in selected cabinets rather than include them
	-cc	--consistency-report		No	Run consistency check also

	-h	--help		No	Show this text
Scenario	**Help**		*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractorManager-1.6.0.jar" com.emc.monorail.tools.Report --help*		
	Extract Reporting	**Report from Mongo Database**	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet --mode extract*		
		Consistency-Report SQL Server Database	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet -c testcab --mode extract -cc*		
		Consistency-Report Oracle Database	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.31.160.75:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet -c testcab --mode extract -cc*		
		Where clause example  Where clause only takes SQL statement and NOT DQL.Potentially ambiguous fields should be prefixed by “s.”	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet -wh "(i_cabinet_id = '0c0004d28000b94b' and (r_modify_date > convert(DATETIME,'2013-01-01')and r_modify_date < convert(DATETIME,'2013-10-01')))" --mode extract -cc*		
		Consistency-Report from multiple cabinets	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md Extract_Cabinet -c "Resources|System|Temp|Templates|dmadmin|netvis_own" --mode extract -cc*		
		Exclude multiple cabinets  Interpret it as Extract Data from all cabinets not in the exclude list.	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md Extract_Cabinet -c "Resources|System|Temp|Templates|dmadmin|netvis_own"* ***-x*** *--mode extract -cc*		
	Transform Reporting	**Report from Mongo Database**	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -mh 127.0.0.1 -mp 27017 -mu*		

		admin -mpw Thom2807 -md Extract_Cabinet --mode transform
Ingest Reporting	**Consistency-Report SQL Server Database**	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet -c testcab --mode ingest -cc*
	Consistency-Report Oracle Database	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -sd oracle.jdbc.driver.OracleDriver -sc jdbc:oracle:thin:@10.31.160.75:1521:ORCL -su source -sp source -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet -c testcab --mode ingest -cc*
	Where clause example Where clause only takes SQL statement and NOT DQL.Potentially ambiguous fields should be prefixed by “s.”	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpw Thom2807 -md Extract_Cabinet -wh "(i_cabinet_id = '0c0004d28000b94b' and (r_modify_date > convert(DATETIME,'2013-01-01')and r_modify_date < convert(DATETIME,'2013-10-01')))" --mode ingest -cc*
	Consistency-Report from multiple cabinets	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md Extract_Cabinet -c "Resources|System|Temp|Templates|dmadmin|netvis_own" --mode ingest -cc*
	Exclude multiple cabinets Interpret it as Extract Data from all cabinets not in the exclude list.	*java -cp "EMATools-1.6.0.jar;EMA-API-1.6.0.jar;EMAIngestManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;EMAExtractManager-1.6.0.jar" com.emc.monorail.tools.Report -td com.microsoft.sqlserver.jdbc.SQLServerDriver -tc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test15_docbase -tu Test15 -tp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md Extract_Cabinet -c "Resources|System|Temp|Templates|dmadmin|netvis_own"* ***-x*** *--mode ingest -cc*

Log4j logging

Using your custom log4j.properties file

Option 1 – Add it to your classpath.

If there are multiple log4j files in your classpath the first file in the classpath gets loaded.

Option 2 – Add JVM parameter for it
-Dlog4j.configuration="file:D:\DEMA\config\log4j.properties"

Logging should be local, not over the LAN, regardless of the LAN speed and speed of the storage. Use a local drive for logs!

Sample log4j.properties file

```
---------------------------------------------------------------------------------------------
log4j.rootCategory=INFO, file, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.rootCategory=INFO, A1, F1
log4j.category.MUTE=OFF
log4j.additivity.tracing=false
log4j.category.tracing=DEBUG, FILE_TRACE
log4j.logger.com.emc.monorail.Morph=DEBUG, FILE2
log4j.additivity.com.emc.monorail.Morph=false
log4j.logger.com.emc.monorail.util=INFO, FCLOG
log4j.additivity.com.emc.monorail.util=false
log4j.logger.STATS=INFO, STATS
log4j.additivity.STATS=false
#------------------ CONSOLE -------------------------
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.threshold=INFO
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%d{ABSOLUTE} %5p [%t] %c - %m%n
#------------------ FILE -------------------------
log4j.appender.F1=org.apache.log4j.RollingFileAppender
log4j.appender.F1.File=E://ema//logs//log4j.log
log4j.appender.F1.MaxFileSize=10MB
log4j.appender.F1.layout=org.apache.log4j.PatternLayout
log4j.appender.F1.layout.ConversionPattern=%d{ABSOLUTE} %5p [%t] %c - %m%n
#------------------ FILE2 -------------------------
log4j.appender.FILE2=org.apache.log4j.RollingFileAppender
log4j.appender.FILE2.File=E://ema//logs//morphx.log
log4j.appender.FILE2.MaxFileSize=10MB
log4j.appender.FILE2.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE2.layout.ConversionPattern=%d{ABSOLUTE} %5p [%t] %c - %m%n
#------------------ FCLOG -------------------------
```

```
log4j.appender.FCLOG=org.apache.log4j.RollingFileAppender
log4j.appender.FCLOG.File=E://ema//logs//fcopy.log
log4j.appender.FCLOG.MaxFileSize=10MB
log4j.appender.FCLOG.layout=org.apache.log4j.PatternLayout
log4j.appender.FCLOG.layout.ConversionPattern=%d{ABSOLUTE} %5p [%t] %c - %m%n
#------------------- STATS -------------------------
log4j.appender.STATS=org.apache.log4j.FileAppender
log4j.appender.STATS.File=E://ema//logs//clone_stats.csv
log4j.appender.STATS.Append=false
log4j.appender.STATS.layout=org.apache.log4j.PatternLayout
log4j.appender.STATS.layout.ConversionPattern=[%t],%m%n
#------------------- FILE_TRACE -------------------------
log4j.appender.FILE_TRACE=org.apache.log4j.RollingFileAppender
log4j.appender.FILE_TRACE.File=E://ema//logs//trace.log
log4j.appender.FILE_TRACE.MaxFileSize=100MB
log4j.appender.FILE_TRACE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE_TRACE.layout.ConversionPattern=%d{ABSOLUTE} [%t] %m%n
```

MongoDB Basics

Intro to MongoDB

NoSQL Database, Schema less, Open Source, Free, Supports multiple languages (C, C++, Java, Python…)
Auto-Sharding for horizontal scalability, Built-in replication for high availability.

RDBMS Database	MongoDB Database
Table	Collections
Row	Documents
Where clause	Query selector
Order by	Sort 1 for ascending and -1 for descending
TOP	Limit & skip

Use	Command
Show all databases	show dbs
Select a database e.g. admin	use admin
Create a collection	db.createCollection("myCollection", { capped : false })
Create a document	db.myCollection.insert({"key1": "value1","key2": "value2"})
Read all documents from a collection	db.myCollection.find()
Read documents from a collection where a key has a particular value.	db.myCollection.find({"key1": "value1"})
Update document in a collection	db.myCollection.update({"key1": “value1"}, {$set: {"key2": “newValue2"}});
Delete a particular document in a collection	db.myCollection.remove({"key1": "value1"});
 MongoDB has a different behavior of update so we have to use $set modifier db.myCollection.update({name: 'Roooooodles'}, {weight: 590}) update found a document by name and replaced the entire document with the new document (the 2nd parameter). This is different than how SQL's update command works. When all you want to do is change the value of one, or a few fields, you are best to use MongoDB's $set modifier db.myCollection.update({name: 'Roooooodles'}, {$set: {weight: 590}})	
Backup MongoDB	mongodump --host localhost --port 27017 --username user --password pass --out E:/mongobackup/mongodump-2015-03-24
Restore MongoDB from a backup	mongorestore --host localhost --port 27017 --username user --password pass --out E:/mongobackup/mongodump-2015-03-24
 MongoDB has a role based authorization model now. If your backup/restore does not work try creating a new user with "backup" and "restore" roles assigned to it and see if that works. http://docs.mongodb.org/manual/reference/built-in-roles/#backup-and-restoration-roles	

Running scripts from MongoDB

Option 1 - Load it in Mongo shell
> load("c:\\temp\\test.js")

Option 2 – Provide the JavaScript file as a command line parameter
mongo admin -u admin -p Thom2807 c:\temp\test.js

Sample Scripts

Purpose	Code
Drop Database	print("----------Deleting Database starts----------"); print("Deleting ExtractorDB_2"); db = db.getSiblingDB('ExtractorDB_2'); db.dropDatabase(); print("----------Deleting Database completed----------");
Generate csv file with id Mapping (old object id and new object id) Script	var databaseName="Extract_Cabinet"; var delimiter=","; db = db.getSiblingDB(databaseName); print("Type,Old Object Id,New Object Id"); loadCursor = db.load.find(); while (loadCursor.hasNext()) { myDocument = loadCursor.next(); print(myDocument.type+delimiter+myDocument.objId+delimiter+myDocument.newObjId+delimiter); }
Count of each collection	Extract_Cabinet="Extract_Cabinet"; print("----------Extraction Report----------"); print("Mongo Database:"+Extract_Cabinet); db = db.getSiblingDB(Extract_Cabinet); collectionArray = db.getCollectionNames(); print("\nCollection Name"+"\t \t"+"Count"); print("----------"); for (var i = 0; i < collectionArray.length; i++) { collection = db.getCollection(collectionArray[i]); if(collectionArray[i] == "load" || collectionArray[i]=="system.indexes") { // we are not printing these collections count } else print(collectionArray[i]+"\t \t"+collection.count()); } print() print("----------Report Complete----------");
Creating _rp collection for a type (Required when migrating from earlier versions where there were no _r tables for a particular type)	print("Dropping and Creating dm_user_rp collection"); db.dm_user_rp.drop(); db.createCollection("dm_user_rp"); db.dm_user_sp.find().forEach(function (doc) { print("Before Insert " + doc.r_object_id); db.dm_user_rp.insert({ _class : "java.util.HashMap", r_object_id: doc.r_object_id, i_position: "-1", restricted_folder_ids : null }) print("After Insert " + doc.user_os_name); }); printjson({"List of r_object_id added to rp collection ":db.getCollection('dm_user_rp').distinct("r_object_id").sort()}) print("End of adding dm_user_rp collection");

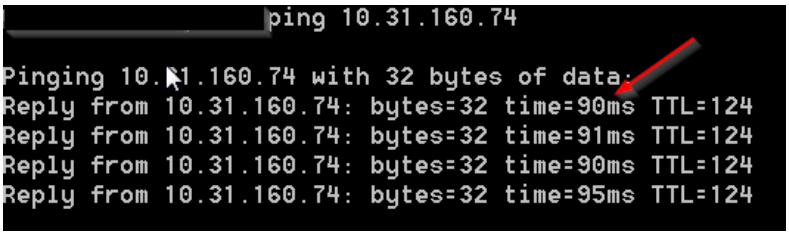
Customize Extractor XML file for your project

1. **Option-1** - Modifying the file directly in the Jar (using 7-zip. Right click on file and edit). The file is present at EMAExtractManager-X.X.X.jar\META-INF\spring\
2. **Option-2** - Create a folder path "**META-INF\spring**" inside a directory (e.g. E:\ema\config) which will be provided in the classpath. So the complete path in this case will be "E:\ema\config\META-INF\spring". Copy the **documentum-extractor-context.xml** file inside this path.

To run with the new extractor config file provide the directory path in the classpath
java -cp "EMAExtractManager-1.6.0.jar;C:/EMA/EMA1.6.0/dependency-jars/*;**E:\ema\config**" com.emc.ema.extractor.ExtractManager -sd com.microsoft.sqlserver.jdbc.SQLServerDriver -sc jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_Test2_docbase -su Test2 -sp Thom2807 -mh 127.0.0.1 -mp 27017 -mu admin -mpa Thom2807 -md ExtractorDB_10 -c "testCab"
Note: The directory path should be after the jar file to override the configuration

Performance Troubleshooting

- Logging level – Set logging level of INFO unless you are running for troubleshooting an issue
- Logs – Should be stored in the local system where EMA components are running and not to a remote system.
- Network Latency - In case you are seeing a slow Extraction & Ingestion there is a possibility that the network latency is in play here. Do a ping check to the source and target system and check the ping latency:



  Anything more than 30ms is bad and we need to try to reduce this. This can mean moving the migration box closer to the source/target system and in the same network instead of doing a VPN.

Extraction performance – If you are using a where clause for extracting a particular dataset adding indexes to the columns in where clause would help improve performance.

- Query Performance – Sometimes complex queries in the where clause case a slow in performance. See if the query can be improved. Another way is to use temporary tables which stores the output of the complex query and the temporary table is then used in the where clause.

Transformation performance – If you are using DB mappers and are querying either the source/target database it would help if the fields being queried for are indexed.

Ingestion performance –

- Memory - Since MongoDB is memory hungry and if you have it running on the same system as Ingestion you might see slow ingestions. For large dataset (multi-million) to ingest have a high memory (RAM). What we have observed is Ingestion performs quite well when you have a big heap (16GB / 32GB) allocated to it. Also consider moving MongoDB to a different system. Check deployment
- If your target database is Microsoft SQL Server change recovery mode of the database to "Simple" (normally "Full") for best performance. Check here
- If possible turn off transaction logs. If there is a problem, this means going to a backup, but it improves ingestion performance a lot on SQL Server.

FileCopier performance –

- If there is some QoS setting enabled turning it off can improve performance.

Automation

EMA Automation script "**EMAAutomation.bat**" present at the "**samples**" folder can be used to run the entire ETL process at once in a single go. It follows these steps Extraction-> Transform-> Ingestion. An error in any of the steps would prevent it from executing further steps.
Remember to update the batch file with the log file & the batch files location for Extract, Transform & Ingest.

Scripting

Properties file and batch files also support inheritance. Similar to default files "#include" can be used for properties file too.
e.g. For Properties file
MongoDB.properties:

```
mongo-host=localhost
mongo-port=27017
mongo-db=Extractdb
mongo-user=admin
mongo-password=Thom2807
```

ExtractorProperties.properties:

```
#include MongoDB.properties
source-driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
source-connection=jdbc:sqlserver://127.0.0.1:1433;databaseName=DM_cloneme_docbase
source-user=sa
source-password=password@123
cabinet-name=cab
```

e.g. For batch file
setenv.bat:

```
SET CLASSPATH=""
SET CLASSPATH=%CLASSPATH%;"C:\EMA\EMA1.6.0\EMATools-1.6.0.jar"
SET CLASSPATH=%CLASSPATH%;"C:\EMA\EMA1.6.0\EMA-API-1.6.0.jar"
SET CLASSPATH=%CLASSPATH%;"C:\EMA\EMA1.6.0\EMAIngestManager-1.6.0.jar"
SET CLASSPATH=%CLASSPATH%;"C:\EMA\EMA1.6.0\dependency-jars\*"

REM EMA Parameters
REM ------------------
SET LOG4j_FILE_PATH="C:\EMA\logs\log4j.properties"

REM Mongo Parameters
SET STG_DB_HOST="127.0.0.1"
SET STG_DB_PORT_NUMBER="27017"
SET STG_DB_NAME="Training"
SET STG_DB_USER_NAME="admin"
SET STG_DB_USER_PASSWORD="Thom2807"
```

compare.bat

```
CALL setenv.bat

SET CMP_ORIGINAL_TYPE="dm_document"
SET CMP_NEW_TYPE="dm_document"
…
```

TROUBLESHOOTING

MongoDB database does not startup properly. Mongo service disappear then reappear over and over again

- Sometimes there are some lock files created which needs to be deleted else they don't allow a restart of Mongo.
- Check the disk space also. Mongo checks on the minimal disk size required for its processing and if it is less than it does not start.
- Launch mongod from command line instead of as a service

  mongod.exe --dbpath="C:\Program Files\MongoDB\Server\3.0\data" --logpath="C:\Program Files\MongoDB\Server\3.0\log\mongo.log"

com.emc.monorail.Ingestor - Waiting for X tasks to complete

- If the earlier message you see is: .emc.monorail.engine.IngestionEngine - Docbase: xxx, User: dmadmin and the waiting message keeps on going on for long you might have exhausted the session limit. Try increasing the session limit
- Break the data into batches (aroud 5 million documents)
- If the target system is Oracle (SQL Server is already taken care by EMA) –
  Have a "EngineConfig.properties" file placed in your class path. It should have the following lines:
  commit-by-batch=true
  batch-size=4000

unique constraint (XXX) violated
OR
Cannot insert duplicate key row in object 'dbo.<table>' with unique index 'XXX.

- You might be running Ingestion again from a previous failed Ingestion. Revert your database to the original state.
- Check if the mongo single collections(_sp) has duplicate entries.
- If you are running Ingestion and are retaining the object ids this might arise if you have not reserved adequate number of object ids. Check the "*KT_EMA-Retaining Object IDs.ppt*" in SyncP for more details.
- If you see this issue after migration in any of your application then restart your application server.

Cannot insert null

- Do you have defaults file in place and have added it to the class path?
- Open the defaults files with notepad (not word pad/write) and make sure the properties are all on separate lines - we had an issue with this whereby there was an error formatting the file when editing with word pad.

"String or binary data would be truncated" exception

- Run DataVerifier for the type for which the exception occurs.
- Do you have defaults file in place and have added it to the class path?
- In case you have created a default file for a particular type and the default value that has been assigned is a single space, the escape needs to be followed by a single space (shown here with an "X" instead of a space).
  custom_attr1=\X
- If your problem is still not solved try running the ingestion in --no-batch mode as it will output the parameters that are failing.

Restarting transform throws error (com.mongodb.CommandFailureException:{"errmsg":"target namespace exists"})

- Run transform again with "-ct"(--clear-target) config. More info about –ct is in help menu of transform

org.springframework.jdbc.support.SQLErrorCodesFactory - SQLErrorCodes loaded: [DB2, Derby, H2, HSQL, Informix, MS-SQL, MySQL, Oracle, PostgreSQL, Sybase, Hana]
org.springframework.dao.DataIntegrityViolationException: PreparedStatementCallback; SQL [INSERT into BATCH_JOB_EXECUTION_PARAMS(JOB_EXECUTION_ID, KEY_NAME, TYPE_CD, STRING_VAL, DATE_VAL, LONG_VAL, DOUBLE_VAL, IDENTIFYING) values (?, ?, ?, ?, ?, ?, ?, ?)]; data exception: string data, right truncation; nested exception is java.sql.SQLDataException: data exception: string data, right truncation

- This error message occurs when a lengthly where clause provided is provided.

To Fix this

- Open the Extractor jar file using 7zip and look for schema-hsqldb.sql file. Right click on the file and select "Edit" option.
- Look for the table BATCH_JOB_EXECUTION_PARAMS and column STRING_VAL. Increase the VARCHAR value to a higher number from the current value and save the file

FileCopier- IllegalArgumentException- Source file not found

- Check the path shown in the error message
- If the path seems correct check if the files are present in the particular path.
- If the path contains current directory path appended at the start then check the file path in filecopy list. Does it contain extra characters like double quotes "

Transform- Rerunning transform throws "source namespace does not exist" error

- Use –ct option in command. It will clear the collection from previous run .NEW ones and convert .ORIG to _sp and _rp.

Cloner- Logs showing error "[DM_OBJECT_W_GET_ATTR_TYPE_ERROR_NAME]warning: "attempt to read value of wrong type from attribute 0 (name object_name)"

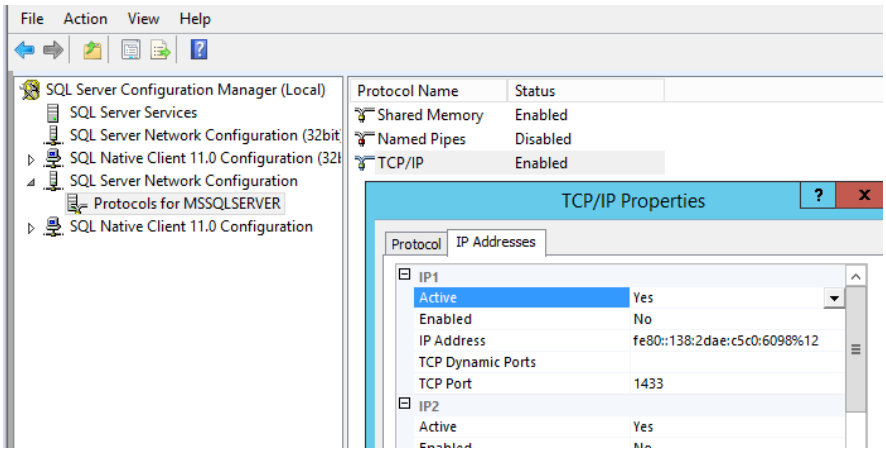
- Did you create the vstamp entry and invalid the views? These are command line options that should normally be used when moving from Oracle to SQL Server.

ERROR [main] com.emc.ema.extractor.documentum.DocumentumExtractor - Deleted object Root version(in /Temp folder) not present in dataset: 0901821880062e7f
FATAL [main] com.emc.ema.extractor.documentum.DocumentumExtractor - Where clause does not include all versions of all documents!!

- Use a where clause to include the particular cabinet to extract, the particular root document and the temp cabinet. E.g. s.i_cabinet_id = '<cabinet ID>' OR s.r_object_id = '<root document ID>' OR s.r_object_id = '<object ID of the Temp cabinet>'

com.microsoft.sqlserver.jdbc.SQLServerException: The TCP/IP connection to the host 127.0.0.1, port 1433 has failed. Error: "Connection refused: connect. Verify the connection properties. Make sure that an instance of SQL Server is running on the host and accepting TCP/IP connections at the port. Make sure that TCP connections to the port are not blocked by a firewall.
at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDriverError(SQLServerException.java:190)

- Check in "SQL Server Connection Manager" that TCP/IP is enabled and the TCP port is set correctly in TCP/IP properties



FATAL [main] com.emc.monorail.repositorytools.DataTicketProvider - Replenish failed: Is the filestore initialized properly?
FATAL [main] com.emc.monorail.repositorytools.DataTicketFactory - DataTicketProvider failed
…
java.lang.Exception: DataTicketProvider failed to provide new Data tickets
at com.emc.monorail.repositorytools.DataTicketFactory.getID(DataTicketFactory.java:31)
at com.emc.monorail.Ingestor.run(Ingestor.java:973)

This happens when the ticket counter has not been initialized - to do this, you will need to add a document to the system and save the content to this filestore. This has to be done before we run the ingestion.
To do this, you can create a new object with API or DFC, and set the a_storage_type to the filestore name before saving, then the file will be copied to the directory structure, and the ticket will be initialized.
To check if the filestore is fully initialized:

1. Get the object ID of the filestore object
2. Search for a row in dmi_sequence_s table that matches the last 4 digits of the filestore object ID
3. If there is a row, then this should be working. If there isn't a row, add a piece of content manually.

FAQ

Q. How many threads the extractor runs by default? Is it common for the extractor to create 40+ parallel db connections?

Ans: Extractor uses the Spring batch framework and have heavily parallelized code so that multiple extractions of different types (including _sp & _rp) can be run in parallel for high performance. The creation and maintenance of Threads is handled by the framework. Typically we have seen 20/25 threads running. For a complex query we have seen Oracle creating multiple sessions for a single query. Hence you might see 40+ db sessions.

Q. How does the EMA Ingestor compute the extension for a particular file?

Ans: Here is how the computation for extension happen

1. Look for dos_extension for the particular format in target by querying the dm_format_s table
2. If o/p of point 1 is null/empty check the dos_extension from the source format object (if the format also is extracted and is being migrated)
3. If o/p of point 2 is null/empty blank is used as an extension

Q. Source repositories (Oracle) range from versions 5.2.5 to 6.5 and needs to be upgraded to 7.X (SQL Server).

- **Can we perform the upgrade of the repository to version 7.x at the same time as performing the DB Conversion or is that recommended to be a step after running EMA by using the normal repository upgrade script?**

  Ans: We can't go from 5.2.5 to 7.x directly. We need to go to 5.3 sp6 or some stepping stone. In other situations. It might be easier to use the ETL approach.
- **Is there any way to bring across a certain range of content (e.g. only 1 business cabinet or last 1 year of content) as part of using the EMA Cloner?**

  Ans: No, this is not possible with Cloner. Cloner takes the whole repo, in one go. There is no option to take the last year or single cabinet. If we want to do that, it's an ETL (which can jump from 5.2.5 to 7.x directly, which might be an advantage)

Q. What are the considerations to consider when using EMA-Migrate to pull content out of 5.2.5 (Oracle backend DB) and straight into 7.x SQL Server?

Ans: Here are some points to consider

- Creating dm_user_rp collection as it does not exist in earlier versions. Check creating _rp collection

Q. How do we handle deleted documents in the source system?

Ans: Currently EMA does not handle deleted documents in the source. Contact the EMA team if you have this requirement.

The following query can be used to get the list of objects ID deleted and using SQL.

SELECT dm_audittrail.audited_obj_id FROM dm_audittrail_sp dm_audittrail WHERE ((dm_audittrail.event_name='dm_destroy') and (dm_audittrail.time_stamp>'01-AUG-2016'))

Run API script in target based on list of objects returned.

destroy,c,<list of audited_obj_ids>

Q. How can delta of dmr_containment objects be done?

Ans: We cannot delta containment objects, we could only overwrite what is already there, as there is no time stamp on the objects.

Q. How many threads the extractor runs by default? Is it common for the extractor to create 40+ parallel db connections?

Ans: Extractor uses the Spring batch framework and have heavily parallelized code so that multiple extractions of different types (including _sp & _rp) can be run in parallel for high performance. The creation and maintenance of Threads is handled by the framework. Typically we have seen 20/25 threads running. For a complex query we have seen Oracle creating multiple sessions for a single query. Hence you might see 40+ db sessions.

Q. Documents created by one user could not be found (via search) by another user. However, they can see each other docs by navigating directly to a folder.

Ans: Here is what was done (when the source was 5.2.5 and target was 7.2):

1. Dm_user alias set and home_docbase have values from the source. I set alias set to '0000000000000000' and docbase to PFA72 (actual target repo name).
2. Dm_group alias set pointing to the source (I set it to '0000000000000000'), group_global_unique_id is empty (I made it docname:group name, PFA72:pfa in my case), group_native_room_id is empty (I set it to '0000000000000000'), group_directory_id is empty (set to '0000000000000000').
3. Dm_acl custom ACLs missing some data, again they were originated in 5.2, repeating values for the r_application_permit and r_permit_type where not inserted according to the existing values of r_accessor_name. 5.2 doesn't have these attributes. So, I updated one permission set and rest of repeating attributes were added successfully.