Concept - Migration from DMS to DAM

Introduction

DMS module do not exist anymore in M5. All binary content should be handled by the DAM module. In order to let existing projects move to M5, we provide several migration tasks.

1. Migrate DMS or any Data repository to the DAM repository.
   Update folder and Data node type and properties.
2. Update Content Nodes properties having DMS references (UUID link from content to DMS).
   Adapt the content property in order to work with the new DAM module and asset handling.
3. Update Data Nodes properties having DMS references (UUID link from content to DMS).
   Adapt the content property in order to work with the new DAM module and asset handling.
4. Migrate Uploaded Data (in the content tree) to the DAM repository.
   Copy binary & create DAM asset.
   Adapt the content property in order to work with the new DAM module and asset handling.
5. Migrate binary date uploaded within the FCK editor (Rich text editor)
   Copy binary & create DAM asset.
   Adapt the link defined in the current text content.
6. Migrate Uploaded Data File (in any workspace) to the DAM repository. For File uploaded with a 4.5.x file or dataFile control.
   Copy binary & create DAM asset.
   Replace the content node by a property of the same name pointing to the newly created asset.
7. Clean Content structure
   Remove all reference to binary content that are not part of the DAM repository.

Tasks

Migrate Binary data to DAM repository

GOAL: Migrate DMS (or other Data repository) into the DAM repository.

REQUIREMENTS:

- All node identifier should be kept (identifier for folder or dataNode are the same in DMS or DAM repository)
- Keep the DMS/Data folder structure
- Possibility to add parent folder on the DAM repository
- Possibility to migrate partial DMS tree structure.

STEPS:

1. Copy DMS/Data specific repository's paths (`List<"/demo-project", "/demo-doc">`) into the target DAM repository and the specified sub-path.
2. Change the property's of Folder node and primaryType (`mgnl:folder`)
3. Change the property's of Asset node and primaryType (`mgnl:asset`)

Class: `MoveDataWorkspaceToDamMigrationTask`
Constructor attributes:

Property	Description	Default value	Valid values
`taskName`	Task name used by the reporting tool, and to log/display informations related to this task.		String

`taskDescription`	Task Description used to display informations in the admin. central update view.		String
`originalPathsList`	List of the data path to migrate. For example from the DMS repository : `Arrays.asList("/demo-project", "/demo-doc")` will only migrate the content of this two folders.		*List<String>*
`targetSubPath`	Subroot folder that will contain the migrated DMS folders. If `targetSubPat=subRoot`, then the DAM repository will have the following structure : `subRoot/deom-project/...` If set to null, the exact same folder structure will be duplicate.		`String`
`dataRepository`	Source repository name. Typically `dms`		`String`

Update Content Nodes properties from DMS/Data to DAM references

GOAL: Change content node reference from DMS to DAM module.

REQUIREMENTS:

- All image / video / audio that reference DMS content should be link to DAM.
- Change property values from jcr identifier to dam compositId for Link's or image gallery.
- Possibility to migrate partial Content tree structure.

STEPS:

1. Iterate the Content nodes and in case of a DMS reference change and remove property
    a. DMS reference
        i. `image=dms`
        ii. `imageDmsUUID=dmsuuid`
    b. DAM reference
        i. `image=jcr:damIdentifier`

Class: `ChangeWebsiteDmsReferenceToDamMigrationTask`
Constructor attributes:

Property	Description	Default value	Valid values
`taskName`	Task name used by the reporting tool, and to log/display informations related to this task.		`String`
`taskDescription`	Task Description used to display informations in the admin. central update view.		`String`
`contentRepository`	Content repository name. Typically `website`		`String`
`contentPathsList`	List of the content path to handle. For example from the `website` repository : `Arrays.asList("/demo-project")` will only handle the content of this folder.		*List<String>*

This migration task extends `AbstractPropertyValueSearchDamMigrationTask`. This abstract class exposes utility methods allowing to retrieve a list of nodes matching property value definition.
For example, if you would to get a list of Nodes under '/demo-project/article' that have property value equal to 'dmsUUID', simply extends `AbstractPropertyValueSearchDamMigrationTask`:

```
public class CustomMigrationTask extends AbstractPropertyValueSearchDamMigrationTask {
...
    private void handlePath(String path) throws RepositoryException {
        // Create Query
        Node rootNode = contentSession.getNode(path);
        if (rootNode.hasNodes()) {
            // set the property to search
            super.setPropertyValue("dmsUUID")
            // create the query String based on the root path. Query will use the value defined
previously.
            String query = super.createQuery(path);
            // execute the query on the specified contentSession
            QueryResult result = super.executeQuery(query, contentSession);
            if (result != null) {
                NodeIterator nodeIterator = result.getNodes();
                while (nodeIterator.hasNext()) {
                    Node node = nodeIterator.nextNode();
```

Update Data Nodes properties from DMS/Data to DAM references

GOAL: Change data node reference from DMS to DAM module.

REQUIREMENTS:

- All image / video / audio **from the DATA workspace** that reference DMS content should be link to DAM.
- Change property values from jcr identifier to dam compositId .
- Possibility to migrate partial Content tree structure.

STEPS:

1. Iterate the DATA nodes and in case of a DMS reference change or remove property (from a. to b.)
    a. DMS reference
        i. `imgUUID=21f33ddf-10b0-430b-b09d-f311921c86d4`
        ii. `img=/demo-project/img/bk/Opener/v-light-4`
    b. DAM reference
        i. `imgUUID=jcr:21f33ddf-10b0-430b-b09d-f311921c86d4`
        ii. `img=/dms-migrated-images/demo-project/img/bk/Opener/v-light-4`

Class: `ChangeDataDmsReferenceToDamMigrationTask`
Constructor attributes:

Property	Description	Default value	Valid values
`taskName`	Task name used by the reporting tool, and to log/display informations related to this task.		`String`
`taskDescription`	Task Description used to display informations in the admin. central update view.		`String`
`contentRepository`	Content repository name. Typically `website`		`String`
`contentPathsList`	List of the content path to handle. For example from the `website` repository : `Arrays.asList("/demo-project")` will only handle the content of this folder.		*`List<String>`*
`identifierPropertyName`	Name of the property that contains the UUID reference to the dms document.	`imgUUID`	`String`
`pathPropertyName`	Name of the property that contains the path of the referenced dms document	`img`	`String`

Move uploaded binary from content(website) to DAM repository

GOAL: Move the binary data stored under the content repository (`website`) to the DAM repository.

REQUIREMENTS:

- All image / video / audio that are stored in the content repository are moved under the DAM repository.
- Create and initialize a new Asset Node based on the moved binary Node.
- From the content Node, refer this Asset as a standard DAM resource.
- Remove the old uploaded binary data from the content repository.

STEPS:

Iterate the Content nodes and in case of an upload content is found:

1. Copy the binary node to the DAM repository
    a. Create the DAM folder
    If the binary node was found under `demo-project/about/subsection-articles/article/content/01` this folder will be created in the DAM repository.
2. Create an Asset Node based on the information stored in the moved binary node.
3. Add the binary node as child of the Asset Node
4. Change the property of the content Node in order to use the DAM module to refer the Binary data.
5. Remove the old binary node stored under the content repository.

Class: `MoveContentToDamMigrationTask`
Constructor attributes:

Property	Description	Default value	Valid values
`taskName`	Task name used by the reporting tool, and to log/display informations related to this task.		String
`taskDescription`	Task Description used to display informations in the admin. central update view.		String
`contentRepository`	Content repository name. Typically `website`		`String`
`contentPathsList`	List of the content path to handle. For example from the `website` repository : `Arrays.asList("/demo-project")` will only handle the content of this folder.		*`List<String>`*
`targetSubPath`	Subroot folder that will contain the migrated uploaded binary. If `targetSubPat=subRoot`, then the DAM repository will have the following structure : `subRoot` `/demo-project/about/subsection-articles/article/content/01` `...` If set to null, the exact same folder structure will be duplicate.		`String`
`searchPropertyValue`	Property value used to identify a node having an associated binary node. If null '`upload`' is used.	upload	`String`

Migrate binary data uploaded within the FCK editor (Rich text editor)

GOAL: Move the binary data uploaded within the FCK editor to the DAM repository.

REQUIREMENTS:

- All binaries stored using the FCK Editor are moved under the DAM repository.
- Create and initialize a new Asset Node based on the moved binary Node.
- From the text content, refer this Asset as a standard DAM resource (Update the internal link).
- Remove the old uploaded binary data from the content repository.

STEPS:

Iterate the Content nodes and in case of a FCK Editor uploaded content is found:

1. Copy the binary node to the DAM repository
    a. Create the DAM folder
    If the binary node was found under `demo-project/about/subsection-articles/article/content/01` this folder will be created in the DAM repository.
2. Create an Asset Node based on the information stored in the moved binary node.
3. Add the binary node as child of the Asset Node
4. Change the link stored within the text property.

5. Remove the old binary node stored under the content repository.

Class: `MoveFCKEditorContentToDamMigrationTask`
Constructor attributes:

Property	Description	Default value	Valid values
`taskName`	Task name used by the reporting tool, and to log/display informations related to this task.		String
`taskDescription`	Task Description used to display informations in the admin. central update view.		String
`contentRepository`	Content repository name. Typically `website`		`String`
`contentPathsList`	List of the content path to handle. For example from the `website` repository : `Arrays.asList("/demo-project")` will only handle the content of this folder.		*`List<String>`*
`targetSubPath`	Subroot folder that will contain the migrated uploaded binary. If `targetSubPat=subRoot`, then the DAM repository will have the following structure : `subRoot` `/demo-project/about/subsection-articles/article/content/01` `...` If set to null, the exact same folder structure will be duplicate.		`String`
`searchPropertyValue`	Property value used to identify a node having an associated binary node. If null ', `repository:'` is used.	,repository:	`String`

Migrate Uploaded Data File (in any workspace) to the DAM repository. For File uploaded with a 4.5.x file or dataFile control.

Migrate uploaded Data File (from any workspace) to the DAM repository

GOAL: Move the binary data uploaded with the file or dataFile control from any workspace to DAM as asset.

REQUIREMENTS:

- All binaries stored using the file or dataFile control are moved under the DAM repository.
- Create and initialize a new Asset Node based on the moved binary Node.
- Remove the original binary.
- Add a property containing the asset composite id key.

STEPS:

Search

- In a workspace
- For certain sub paths
- All resources node with certain name having a binary property

For each

1. Copy the binary node to the DAM repository
    a. Create the DAM folder
    If the binary node was found under `data/myType/photo/image` this folder will be created in the DAM repository.
2. Create an Asset Node based on the information stored in the moved binary node.
3. Add the binary node as child of the Asset Node
4. Add a property (same name as the node name) containing the new asset composite key. .
5. Remove the old binary node stored under the content repository.

Class: `MoveFileContentToDamMigrationTask`
Constructor attributes:

Property	Description	Default value	Valid values
`taskName`	Task name used by the reporting tool, and to log/display informations related to this task.		String

`taskDescription`	Task Description used to display informations in the admin. central update view.		String
`contentRepository`	Content repository name. Typically `data`		`String`
`contentPathsList`	List of the content path to handle. For example from the `website` repository : `Arrays.asList("/myType")` will only handle the content of this folder.		*`List<String>`*
`targetSubPath`	Subroot folder that will contain the migrated uploaded binary. If `targetSubPat=subRoot`, then the DAM repository will have the following structure : `subRoot /demo-project/about/subsection-articles/article/content/01 ...` If set to null, the exact same folder structure will be duplicate.		`String`
`searchFieldName`	Name of the content node to search .Typically this is the name of the file or dataFile field.		`String`

Clean content(website) for DAM repository

GOAL: Clean content repository and remove all Binary references not part of the DAM workspace.

REQUIREMENTS:

- Remove all orphan binary references (A TextImage component having a Jcr identifier for the image not found in the DAM repository).
- Create DAM compositeId for link content reference.

STEPS:

1. Create a NodeVisitor (get all nodes having one of the following properties name *'image/video/flash/logoImg/printLogoImg/teaserImg'*)
    a. If the property value is a DAM compositId and this id refer to an existing asset, keep the link, otherwise remove the property and log.
2. Create a NodeVisitor (get all nodes having the following property name *'link'*)
    a. If this link refer to a DAM binary, create a compositeId.

Class: `CleanContentForDamMigrationTask`
Constructor attributes:

Property	Description	Default value	Valid values
`taskName`	Task name used by the reporting tool, and to log/display informations related to this task.		String
`taskDescription`	Task Description used to display informations in the admin. central update view.		String
`contentRepository`	Content repository name. Typically `website`		`String`
`contentPathsList`	List of the content path to handle. For example from the `website` repository : `Arrays.asList("/demo-project")` will only handle the content of this folder.		*`List<String>`*

Own migration task

`CleanContentForDamMigrationTask` extend `AbstractCleanContentForDamMigrationTask`. If you want to create your own clean task, using visitor pattern, just extends `AbstractCleanContentForDamMigrationTask` , and override the abstract method.
Visitor Pattern is quite easy to use: Create your Visitor, and VisitorAction method. For every Node reaching the Visitor condition, call the VisitorAction.

```
public class CustomCleanContentForDamMigrationTask extends AbstractCleanContentForDamMigrationTask {
  ...
    /**
     * Override in order to add your custom visitors.
     */
    @Override
    protected List<NodeVisitor> addCustomVisitors() {
        List<NodeVisitor> visitorList = new ArrayList<NodeVisitor>();
        visitorList.add(createCustomNodeVisitor());
        ...


    private NodeVisitor createCustomNodeVisitor() {
        NodeVisitor cleanUpVisitor = new NodeVisitor() {
            private String prefix = "video";
            @Override
            public void visit(Node node) throws RepositoryException {
                // First filter Nodes based on their Type
                if (NodeUtil.isNodeType(node, NodeTypes.Page.NAME) || NodeUtil.isNodeType(node,
NodeTypes.Component.NAME)) {
                                        // This return all properties having a name starting with video
                    PropertyIterator propertyIterator = node.getProperties(prefix);
                    while (propertyIterator.hasNext()) {
                        Property property = propertyIterator.nextProperty();
                        // Take Action
                        createCustomNodeVisitorTakeAction(node, property);
                    }
                                ....         private void createCustomNodeVisitorVisitorTakeAction(Node
node, Property property) throws RepositoryException {
          try {
               // Take Action
```

Usage

Since M5, all modules are responsible to perform the required migration. For example, STK demo project module (for Magnolia 4.5.x), bootstrapped files for DMS and also Content (Website workspace).
Website contents (demo-project, demo-features) have references to DMS.
It is on the responsibility of the demo project module to perform the migration of DMS to DAM and to handle the Content migration.

Let's have a look to the version handler of this module (`DemoProjectVersionHandler`):

```
.addTask(
  // Conditional task: Only to perform if DMS module is installed (should be the case..)
  new IsModuleInstalledOrRegistered("Migrate DMS Repository", "Migrate only if DMS is installed", "dms",
           // Move Binary from DMS to DAM workspace
       new MoveDataWorkspaceToDamMigrationTask("Migration task: Migrate DMS content to DAM", "Migrate DMS to
DAM", Arrays.asList("/demo-project"), null, "dms")))
.addTask(
  new IsModuleInstalledOrRegistered("Migrate DMS Repository", "Migrate only if DMS is installed", "dms",
       // Change Content references from DMS to DAM
       new ChangeDataReferenceToDamMigrationTask("Migration task: Migrate Identifier ID to point to DAM",
"magnolia-demo-project", "website",
           Arrays.asList("/demo-   project", "/demo-features"))))
.addTask(
  // Move Content Binary to DAM workspace
  new MoveUploadedContentToDamMigrationTask("Migration task: Migrate Uploaded content to DAM repository",
"magnolia-demo-project", "website",
      Arrays.asList("/demo-project","/demo-features"), "/moved_uploaded"))
.addTask(
  // Clean Content repository
  new CleanContentForDamMigrationTask("Migration task: Clean Content repository", "magnolia-demo-project",
"website", Arrays.asList("/demo-project", "/demo-features")))
```

Your custom module will have to perform the same steps.

⚠️ Always run ChangeDataReferenceToDamMigrationTask after MoveDataWorkspaceToDamMigrationTask .