



You probably didn't know that...
(small hidden features in Alfresco)

David Ciamberlano



Are you a curious person?

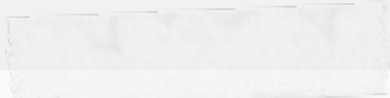
Alfresco hides many small less known features that maybe you'll never need to use... but that might prove useful on more than one occasion. So...

you probably didn't know that...

You can change the unchangeable

In Alfresco there are a few properties that **cannot be changed** (... for a good reason!):

- creator of a node (**cm:creator**)
- creation date (**cm:created**)
- modifier (**cm:modifier**)
- modification date (**cm:modified**)
- [...]



Mimetype: Adobe PDF
Document

Size: 639 KB

Creator: zak

Created Date: Thu 1 Dec
1988 00:00:00

Modifier: admin

Modified Date: Thu 28 Dec
2017 18:44:52

You can change the unchangeable

these properties belongs to the **cm:auditable** aspect (in *contentModel.xml*).

In a ideal world we wouldn't need to change them...
... but unfortunately we don't live in a ideal world.

The "good news" is that there is a way to change these values...
And it's (relatively) simple!

You can change the unchangeable

A behavior prevents these metadata from being changed.
All we need is to **disable** that behavior before the update.

```
props.put(ContentModel.PROP_CREATOR, newCreator);  
props.put(ContentModel.PROP_CREATED, newCreated);  
props.put(ContentModel.PROP_MODIFIER, newModifier);  
props.put(ContentModel.PROP_MODIFIED, newModified);  
  
policyBehaviourFilter.disableBehaviour(nodeRef, ContentModel.ASPECT_AUDITABLE);  
try {  
    nodeService.setProperties(nodeRef, props);  
}  
finally {  
    policyBehaviourFilter.enableBehaviour(nodeRef, ContentModel.ASPECT_AUDITABLE);  
}
```

We can disable the **auditable** behavior for a specific node **immediately before** the `setProperty()`...

You can change the unchangeable

A behavior prevents these metadata from being changed.
All we need is to **disable** that behavior before the update.

```
props.put(ContentModel.PROP_CREATOR, newCreator);  
props.put(ContentModel.PROP_CREATED, newCreated);  
props.put(ContentModel.PROP_MODIFIER, newModifier);  
props.put(ContentModel.PROP_MODIFIED, newModified);  
  
policyBehaviourFilter.disableBehaviour(nodeRef, ContentModel.ASPECT_AUDITABLE);  
try {  
    nodeService.setProperties(nodeRef, props);  
}  
finally {  
    policyBehaviourFilter.enableBehaviour(nodeRef, ContentModel.ASPECT_AUDITABLE);  
}
```

We can disable the **auditable** behavior for a specific node immediately before the `setProperty()`...

...and **re-enable** it soon after (note the *finally* block)!

Import an acp preserving the uuid of the nodes

To rely on the uuid to retrieve a document from an external application is **not a best-practice...**

but it happens often!

There's a way to preserve the original uuid when we need to import an acp.

Import an acp preserving the uuid of the nodes

we can create a new action, extending the
OOTB importer class
(**ImporterActionExecutor**)...



```
public class NewImporterActionExecutor extends ImporterActionExecutor {
```

... and override its **executeImpl**
method...



```
@Override  
public void executeImpl(Action ruleAction, NodeRef actionedUponNodeRef) {
```


Import an acp preserving the uuid of the nodes

we can create a new action, extending the
OOTB importer class
(**ImporterActionExecutor**)...



```
public class NewImporterActionExecutor extends ImporterActionExecutor {
```

... and override its **executeImpl**
method...



```
@Override  
public void executeImpl(Action ruleAction, NodeRef actionedUponNodeRef) {
```



...in which we find the
importservice.importView() call.

To preserve the uuid, we have to use a
custom modified **ImporterBinding**



```
ACPIImportPackageHandler importHandler = new ACPIImportPackageHandler(acpFile,  
    (String) ruleAction.getParameterValue(PARAM_ENCODING));  
importService.importView(importHandler, new Location(importDestination),  
    NewImporterBinding, importerProgress: null);
```

* to see the complete code: <https://gist.github.com/david-ciamberlano>

Import an acp preserving the uuid of the nodes

For example, we can unzip an acp, open its xml descriptor and modify the uuid of one of its nodes...

acp xml descriptor

```
<cm:title>
| <view:mlvalue view:locale="en_GB">Project Objectives.ppt</view:mlvalue>
</cm:title>
<cm:creator>mjackson</cm:creator>
<sys:node-uuid>alfresco-0000-0000-0000-xdevcon2018x</sys:node-uuid>
<cm:name>Project Objectives.ppt</cm:name>
<sys:store-protocol>workspace</sys:store-protocol>
```

Import an acp preserving the uuid of the nodes

Node Information

Reference workspace://SpacesStore/alfresco-0000-0000-0000-xdevcon2018x
Primary Path /app:company_home/st:sites/cm:import-test/cm:documentLibrary/cm:app-destination/cm:Project_x0020_Objectives.ppt
Type cm:content
Parent workspace://SpacesStore/1e361a21-74f1-40eb-b4d1-7bad6f25973d

Properties (13)

Name	Type	Value	Residual	Actions
cm:created	d:datetime	03 Mar 2011 11:31:30 GMT+0100 (CET)	false	Delete
cm:title	d:mltext	Project Objectives.ppt	false	Delete
cm:creator	d:text	mjackson	false	Delete
sys:node-uuid	d:text	alfresco-0000-0000-0000-xdevcon2018x	false	Delete
cm:name	d:text	Project Objectives.ppt	false	Delete
sys:store-protocol	d:text	workspace	false	Delete
cm:content	d:content	contentUrl=store://2017/12/30/16/28/4636de01-6071-4ccd-b662-b4317de99b40.bin mimetype=application/vnd.ms-powerpoint size=2117632 encoding=UTF-8 locale=it_ id=541	false	Delete
sys:store-identifier	d:text	SpacesStore	false	Delete
sys:node-dbid	d:long	1328	false	Delete
sys:locale	d:locale	en_GB	false	Delete
cm:modifier	d:text	mjackson	false	Delete
cm:owner	d:text	admin	false	Delete
cm:modified	d:datetime	03 Mar 2011 11:31:31 GMT+0100 (CET)	false	Delete

After the import of the acp, we will find a node in Alfresco with the previously modified uuid

Residual?
what's that
column for?

Residual properties

In Alfresco, properties may exist that do not belong to any registered model.

They are called **residual properties** and are useful, for example, to handle the case of a property removed from a type or an aspect.

Suppose we have an aspect with a property called *acme:test-residual* applied to a certain node:

Name	Type	Value	Residual
acme:test-residual	d:text	I will be residual...	false

what if we remove that property from the aspect?

Residual properties

In Alfresco, properties may exist that do not belong to any registered model.

They are called **residual properties** and are useful, for example, to handle the case of a property removed from a type or an aspect.

Suppose we have an aspect with a property called *acme:test-residual* applied to a certain node:

Name	Type	Value	Residual
acme:test-residual	d:text	I will be residual...	false

what if we remove that property from the aspect?

The property continue to exists for the node... but is marked as *residual*.

Name	Type	Value	Residual
acme:test-residual		I will be residual...	true


Residual properties

You can also create a "residual property" programmatically, as if it were any other property...


JavaScript input

Freemarker input

Script execution parameters



```
1 var xnode = search.findNode("workspace://SpacesStore/d4c61ac4-dale-471f-87ac-925d4063d9fa");
2
3 xnode.properties["whoami"] = "wtf";
4 xnode.save();
5
```



Name	Type	Value	Residual	Actions
app:editInline	d:boolean	true	false	Delete
cm:modifier	d:text	admin	false	Delete
cm:modified	d:datetime	30 Dec 2017 01:30:03 GMT+0100 (CET)	false	Delete
cm:whoami	<none>	wtf	true	Delete

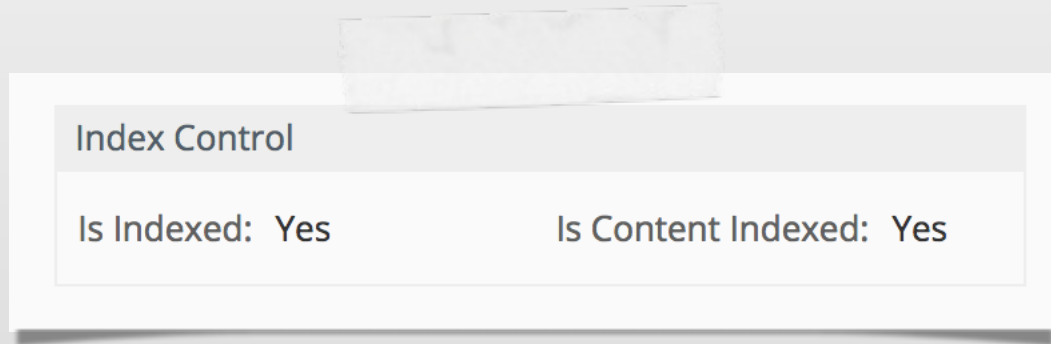
* "whoami" does not belongs to any model

Less known aspects - cm:indexControl

You can control the indexing of contents in Alfresco applying the aspect **cm:indexControl** to a node.

The aspect exposes 2 properties:

- **cm:indexed** (controls whether or not the node is indexed)
- **cm:isContentIndexed** (controls the full-text indexing of the node)



Less known aspects - cm:indexControl

Properties values combinations:

<i>cm:indexed</i>	<i>cm:isContentIndexed</i>	<i>indexing</i>
<i>True</i>	<i>True</i>	<i>metadata and content</i>
<i>True</i>	<i>False</i>	<i>metadata but not content</i>
<i>False</i>	<i>-</i>	<i>no</i>

Less known aspects - sys:hidden

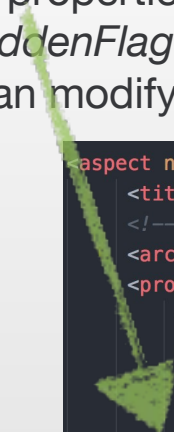
The **sys:hidden** aspect (defined in **systemModel.xml**) allows you to hide a node from a client or service (such as share, webdav, cifs, ...).

```
<aspect name="sys:hidden">
  <title>Hidden</title>
  <!-- Explicitly turn off archiving for all nodes with this aspect -->
  <archive>false</archive>
  <properties>
    <!-- A bit mask encoding whether clients
         (such as CIFS, WebDav, Share, ...) can see the node -->
    <property name="sys:clientVisibilityMask">=
    <!-- Is this file hidden regardless of file name patterns -->
    <property name="sys:hiddenFlag">=
    <!-- Should the hidden aspect cascade to child nodes? -->
    <property name="sys:cascadeHidden">=
    <!-- Should the index control aspect cascade to child nodes? -->
    <property name="sys:cascadeIndexControl">=
    <property name="sys:clientControlled">=
  </properties>
</aspect>
```

Less known aspects - sys:hidden

sys:hidden has 5 properties

(*clientVisibility*, *hiddenFlag*, *cascadeHidden*, *CascadeIndexControl*, *clientController*)
with which we can modify its behavior.



```
aspect name="sys:hidden">
  <title>Hidden</title>
  <!-- Explicitly turn off archiving for all nodes with this aspect -->
  <archive>false</archive>
  <properties>
    <!-- A bit mask encoding whether clients
    (such as CIFS, WebDav, Share, ...) can see the node -->
    <property name="sys:clientVisibilityMask">=
    <!-- Is this file hidden regardless of file name patterns -->
    <property name="sys:hiddenFlag">=
    <!-- Should the hidden aspect cascade to child nodes? -->
    <property name="sys:cascadeHidden">=
    <!-- Should the index control aspect cascade to child nodes? -->
    <property name="sys:cascadeIndexControl">=
    <property name="sys:clientControlled">=
  </properties>
</aspect>
```

Less known webscripts - export a site

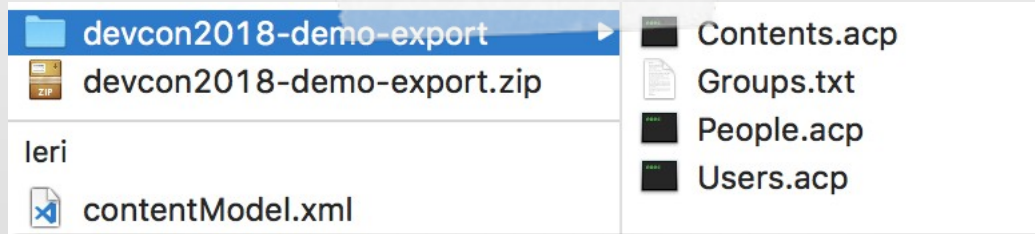
It's possible to export the whole content of an Alfresco Site with one command:

GET: /alfresco/service/api/sites/{site-shortname}/export

```
david@david-mac:~/Desktop$ curl "http://localhost:8080/alfresco/service/api/sites/devcon2018-demo/export"
-u admin:admin -o devcon2018-demo-export.zip
```

% Total	% Received	% Xferd	Average	Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left	Speed
100	8265k	0	8265k	0	0	1104k	0	--:--:-- 0:00:07 --:--:-- 1141k

```
david@david-mac:~/Desktop$
```

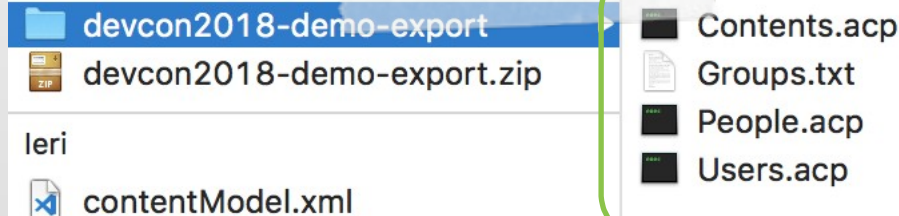


Less known webscripts - export a site

It's possible to export the whole content of an Alfresco Site with one command:

GET: /alfresco/service/api/sites/{site-shortname}/export

```
david@david-mac:~/Desktop$ curl "http://localhost:8080/alfresco/service/api/sites/devcon2018-demo/export"
-u admin:admin -o devcon2018-demo-export.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 8265k    0 8265k    0     0  1104k      0  --:--:--  0:00:07 --:--:-- 1141k
david@david-mac:~/Desktop$
```



Less known webscripts - dictionary & classes

To check quickly if a custom model has been correctly installed we can use (guess?) a webscript:

GET /alfresco/service/api/dictionary

GET /alfresco/service/api/classes/{type-name}

```
david@david-mac:~/Desktop$ curl "http://localhost:8080/alfresco/service/api/
classes/acme_document" -u admin:admin
{
  "name": "acme:document",
  "isAspect": false,
  "isContainer": false,
  "title": "ACME Document",
  "description": "",
```





You probably didn't know that...
(small hidden features in Alfresco)

David Ciamberlano
Thank you!