Remove Dimensions From Planning LCM Extracts

Problem

I am currently working with a client that is updating a planning application and one of the changes is to remove a dimension. After the new application was setup and the hierarchies were modified to meet the objectives, migrating artifacts was the next step. As many of you know, if you try to migrate web forms and composite forms, they will error during the migration due to the additional dimension in the LCM file. It wouldn't be a huge deal to edit a few XML files, but when there are hundreds of them, it is extremely time consuming (and boring, which is what drove me to create this solution).

Assumptions

To fully understand this article, a basic understanding of XML is recommended. The example below assumes an LCM extract was run on a Planning application and it will be used to migrate the forms to the same application without a CustomerSegment dimension. It is also assumed that the LCM extract has been downloaded and decompressed.

Solution

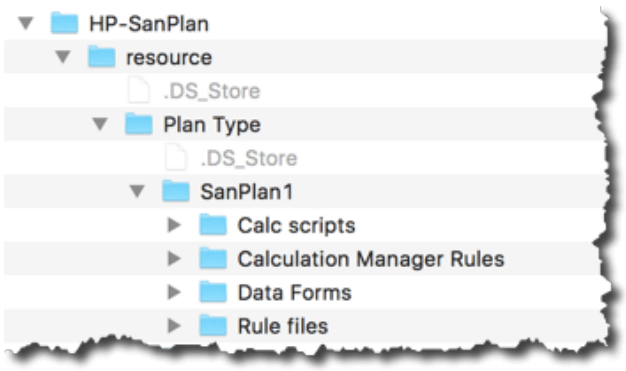
I have been learning and implementing PowerShell scripts for the last 6 months and am overwhelmed by how easy it is to complete complex tasks. So, PowerShell was my choice to modify these XML files in bulk.

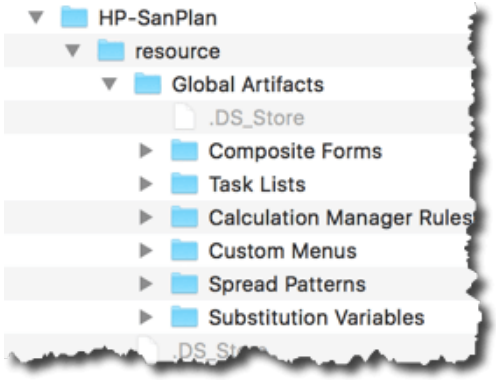
It would be great to write some long article on how smart this solution is and overwhelm you with my whit, but there is not much too it. A few lines of PowerShell will loop through all

the files and remove the XML tags related to a predefined dimension. So, let's get to it.

Step 1 – Understand The XML

There are two folders of files we will look to. Forms are under the plan type and the composite forms are under the global artifacts. Both of these are located inside the resource folder. If there are composite forms that hold the dimension in question as a shared dimension, both will need to be impacted. Scripts will be included to update both of these areas.





Inside each of the web form files will be a tag for each dimension, and it will vary in location based on whether the dimension is in the POV, page, column, or row. In this particular example, the CustomerSegment dimension is in the POV section. What we want to accomplish is removing the <dimension/> tag where the *name* attribute is equal to CustomerSegment.

For the composite forms, the XML tag is slightly different, although the concept is the same. The tag in composite form XML files is <sharedDimension/> and the attribute is

dimension, rather than name.

Step 2 – Breaking Down the PowerShell

The first piece of the script is just setting some environment variables so the script can be changed quickly so that it can be used wherever and whenever it is needed. The first variable is the path of the Data Forms folder to be executed on. The second is the dimension to be removed.

```
# Identify the source of the Data Forms folder and the dimension to be removed
# List all files, recursively, that exist in the path above
$files = Get-ChildItem $lcmSourceDir -Recurse |
where {$_.Attributes -notmatch 'Directory'} |
```

The next piece of the script is recursing through the folder and storing the files in an array. There is a where statement to exclude directories so the code only executes on files.

```
# List all files, recursively, that exist in the path above
$files = Get-ChildItem $lcmSourceDir -Recurse |
where {$_.Attributes -notmatch 'Directory'} |
```

Step 3 – Removing The Unwanted Dimension

The last section of the script does most of the work. This will loop through each file in the $files array and

1. Opens the file
2. Loops through all tags and deletes any <dimension/> tag with a name attribute with a value equal to the $dimName

variable
3. Saves the file

```
# Loop through the files and find an XML tag equal to the dimension to be removed
Foreach-Object {


$xml = Get-Content $_.FullName
$node = $xml.SelectNodes(“//dimension”) |
Where-Object {$_.name -eq $dimName} | ForEach-Object {
# Remove each node from its parent
[void][/void]$_.ParentNode.RemoveChild($_)
}
$xml.save($_.FullName)
Write-Host “($_.FullName) updated.”
}
```

Executing The Logic On Composite Forms

The above concepts are exactly the same to apply the same logic on composite forms files in the LCM. If this is compared to the script applied to the web forms files, there are three differences.

1. The node, or XML tag, that needs to be removed is called *sharedDimension*, not *dimension*. (highlighted in red)
2. The attribute is not *name* in this instance, but is called *dimension*. (highlighted in red)
3. We have added a counter to identify whether the file has the dimension to be removed and only saves the file if it was altered. (highlighted in green)

The Script

```
$lcmSourceDir = "Z:\Downloads\KG04\HP-SanPlan\resource\Global Artifacts\Composite Forms"
$dimName = "CustomerSegment"
# List all files
$files = Get-ChildItem $lcmSourceDir -Recurse | where
```

```
{$_.Attributes -notmatch 'Directory'} |
# Remove CustomerSegment
Foreach-Object {
  # Reset a counter to 0 - used later when files is saved
  $fileCount = 0

$xml = Get-Content $_.FullName
$node = $xml.SelectNodes("//sharedDimension") | Where-Object
{$_.dimension -eq $dimName}  | ForEach-Object {
#Increase the counter for each file that matches the criteria
    $fileCount++
# Remove each node from its parent
[void][/void]$_.ParentNode.RemoveChild($_)
}
# If the dimension was found in the file, save the updated
contents.
  if($fileCount -ge 1) {
$xml.save($_.FullName)
Write-Host "$_.FullName updated."
    }
}
```

Summary

The first script may need to be run on multiple plan types, but the results is an identical folder structure with altered files that have the identified dimension removed. This can be zipped and uploaded to Shared Services and used to migrate the forms to the application that has the dimension removed.

The scripts above can be copied and pasted into PowerShell, or the code can be Downloaded.