Supercharge PBCS with PowerShell

Last year I presented an in-depth overview on PowerShell and how it can be utilized in the Hyperion environment. I have been asked many times to share it. The presentation is a technical presentation and is meant to provide a strong introductory level foundation for anybody that wants to start using PowerShell to automate repetitive tasks. I have built a large library of shared functions that can be used to automate PBCS and ePBCS, and I plan to share pieces of this in future posts.

For now, anybody that is interested in learning PowerShell, or has used it and doesn’t know why some things work and others don’t, this might prove to be a valuable resource.

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.

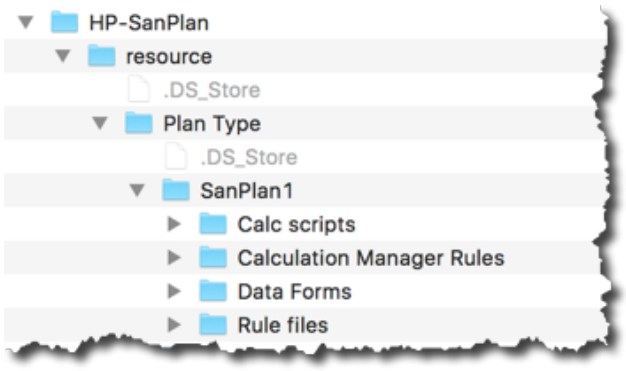
Solution

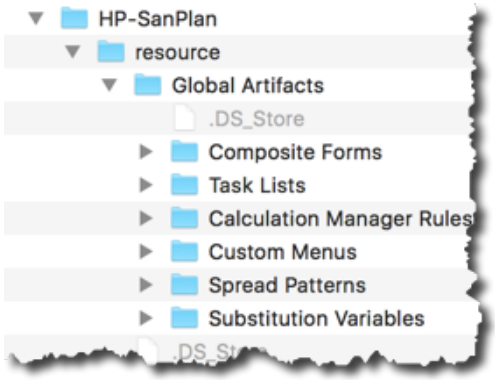
I have been learning and implementing PowerShell scripts for the last 6 months and am overwhelmed by how easy it is to complete complex tasks. So, PowerShell was my choice to modify these XML files in bulk.

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.
[crayon-5b272f4935466138635192/]
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.
[crayon-5b272f4935475794073537/]

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

[crayon-5b272f493547c602557087/]

```
$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

[crayon-5b272f4935483015102987/]

```
$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.

Use PowerShell to split large files by month/year for data loads into FDMEE on PBCS

If you are using PBCS, you may run into some challenges with large files being passed through FDMEE. Whether performance is an issue or you just want to parse a file my month/year, this script might save you some time.

The Challenge

I recently had the need to break apart a file. The source provided one large text file that included 2 years of data that was needed to populate the history of an employee metrics application. The current process loaded files by month and we wanted to be able to piggy back off the existing scripts to load and process data in FDMEE and the monthly Planning data pushes to the ASO reporting cube. So, the need break the data file into seperate files by month and year was required. The file was delimited and formatted like the following.

```
Entity,Year,Scenario,Period,Account,Date,Employee,Pay Code,JobNumber,Data
BU1005,FY15,Actual,Feb,Pay Amount,02/02/2015,V1398950,P105,,108.10
BU1005,FY15,Actual,Feb,Pay Amount,02/03/2015,V1398950,P105,,108.92
```

The goal was to have a file for every unique month and year combination that included only the lines of the relevant time periods. The header of the file also had to exist in each of the smaller files. Since we were working on a Windows machine, we used PowerShell to script the solution.

Powershell Script Directions

The script is pretty simple to use and understand. Update the script as follows.

1. Create a new text file with a ps1 extension and paste the following into that file.
2. Update the srcFile variable to point to the file to be parsed.
3. Update the startYear to the first year in the file to be extracted.
4. Update the currentYear variable to the last year in the file to be extracted.
5. Update the ProcessName to a meaningful word or phrase that will be used to create the file name.
6. Save the file and execute it like any other PowerShell script.

This will produce 12 files for each year with the header line and the data related to the month and year that represents the year and month in the file name.

Disclaimer

I welcome feedback on improving performance and will give credit to anybody that can improve on this. I am NOT an expert in PowerShell and I am sure there are faster ways to accomplish this. This created 12 files (1 year / 12 months) from a file that includes 7.8 million records and completed in 24 minutes. So, this is pretty reasonable for one-off requests, but might need attention if it was a repeatable need.

This was developed using PowerShell 5 and some functions do not work in earlier adoptions of the software.

Powershell Script

[crayon-5b272f49360e4646283583/]
[crayon-5b272f49360ef319896124/]

Conclusion

Hopefully this will benefit the community. As I create more scripts like this, I plan to share them.