White paper: Advanced scripting - Outputjob

Jelle Laarman 17-02-2016

Introduction

Output job is the way to get consistent output out of Altium designer, this is a portable document which we can reuse across projects.

With an output job you can generate the standard outputs for Assembly and Fabrication like Gerber, NC drill, pick and place and ODB++ files.

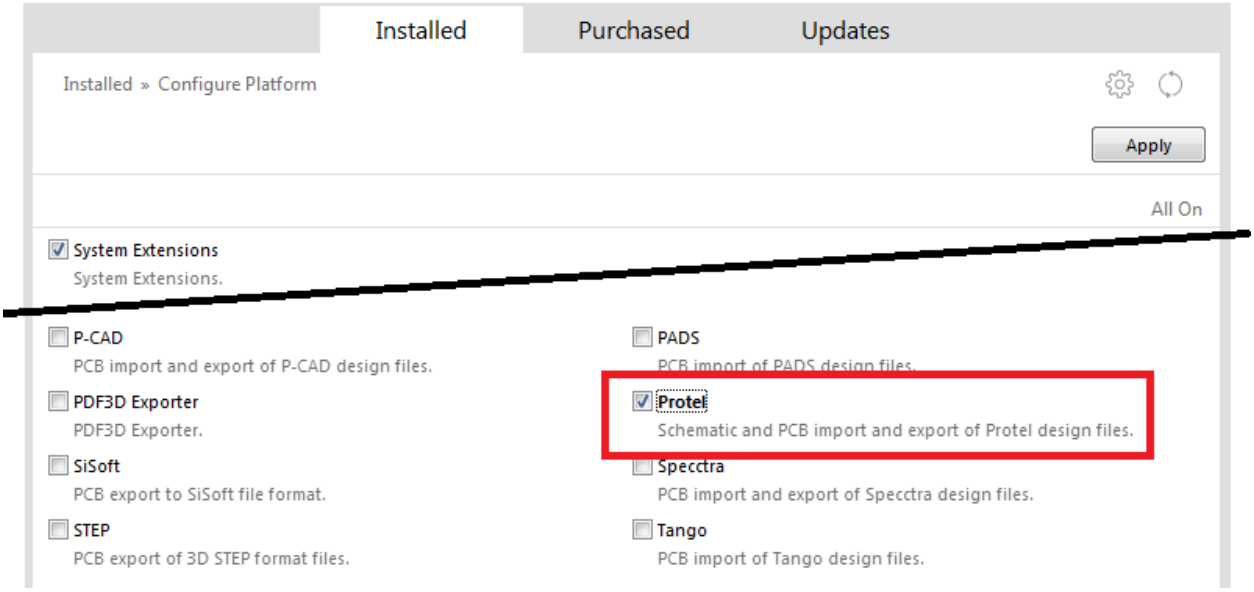
Sometimes we want special output from our CAD software.
This might be done to support some dedicated validation process, or to generate production machine specific files.
Altium supports this through the scripting possibility inside the outputjob.
This way you customized outputs are integrated in your standard workflow without additional manual intervention.

In this whitepaper we will set up an outputjob script which generates an ASCII PCB file.
This is not a standard output of the output job, but many post-processing software uses this type of file for checks or visualization.

This shows Altium Beyond Macro scripting capabilities

Prerequisite

ASCII PCB file >>exporter



Check this by Opening a PcbDoc and see if the Entry File>>Export>> Protel PCB 2.8 ASCII exists, if it is there your good to go.

Start Scripting

Now that we have installed the needed plugins for our goals, we can start scripting.

The best place to start scripting is to see how it is done in the examples, this gives a good feeling and idea on how to approach your scripting challenge. The examples can be found at:
https://techdocs.altium.com/display/SCRT/Script+Examples+Reference

ASCII Output Specific

The most important part of the script is of course the actual exporting of the ASCII file.
Luckily there is a standard plugin available which takes care of the hard work here.

Some research shows that the way to save/export is done through the command:

```
MyServerDocument.DoFileSave('Protel PCB 2.8 ASCII(*.pcb)');
```

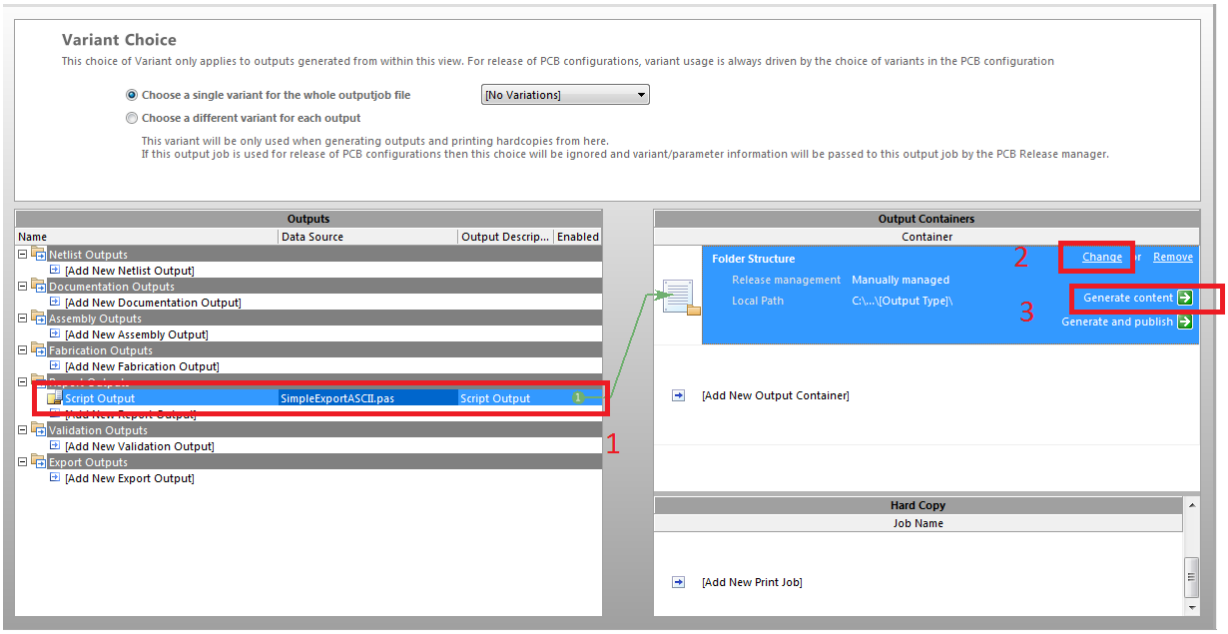
This Command will save the currently active document, represented by the "MyServerDocument" variable.
So our next task is to make sure that our Focused document is the PcbDoc that we want to save, and that it is already on the location where we want it to go:

```
Procedure Generate(Parameters : String);
Var
    CurrWorkSpace        : IWorkSpace; // An Interface handle to the current workspace
    PCBProject           : IProject;   // An Interface handle to the current Project
    MyServerDocument     : IServerDocument;
    i                    : Integer;
    Path                 : TDynamicString;
Begin
   CurrWorkSpace := GetWorkSpace;
   If CurrWorkSpace = Nil Then Exit;
   PCBProject := GetWorkspace.DM_FocusedProject;
   If PCBProject = Nil Then Exit;
   For i := 0 to PCBProject.DM_LogicalDocumentCount - 1 Do //find the PcbDoc
   Begin
      if PCBProject.DM_LogicalDocuments(i).DM_DocumentKind = 'PCB' Then
      Begin
         MyServerDocument := PCBProject.DM_LogicalDocuments(i);
      End;
   End;
   Path := MyServerdocument.DM_FullPath;
   //Close the DOC first or export will fail
   ResetParameters;
   AddStringParameter('ObjectKind','ProjectDocuments');
   RunProcess('WorkspaceManager:CloseObject');
   //Copy the orginal PCBdoc to a new location
   MyServerDocument := Client.OpenNewDocument('PCB',Path, PCBoutFile,FALSE);
   MyServerDocument.DoFileSave('');
   Client.ShowDocument(MyServerDocument);
   //save the copy
   MyServerDocument.DoFileSave('Protel PCB 2.8 ASCII(*.pcb)');//saves as ASCII
   //close the newly created doc
   Client.CloseDocument(MyServerDocument);
   //delete it from Project
   PCBProject.DM_RemoveSourceDocument(PCBoutFile);
   //And add the original
   PCBProject.DM_AddSourceDocument(Path);
End;
```

So, this should do it.

Outputjob Specific

This is the layout of the outputjob document, Most Altium user use this document in a “define once, use many” fashion.



For our script there are 3 important “events”

1- Configure
2- PredictFileOutputNames
3- Generate

1-Configure

This event, which can be fired by double click, RMB>>Configure or ALT+ENTER at the selected outputer (1) would usually open the properties of the selected outputter, so here we could make a Dialog window with settings, but our script has no additional settings.

Selecting this will call:

```
Function Configure(Parameters : String) : String;
```

From our script.

The configure function get a String as parameter, and in returns a String (a function in Delphi always returns a value, if no return value is desired then a procedure is utilized).

The Parameter(s) are supplied AND saved by the outputjob document.
The writer of the script determines what goes in these parameters, by returning the new string value via the “Result” variable.

The OutputJob document in a human readable ASCII document, and the Script call get the following line:

Configuration1_Item1=|RECORD=ScriptView|SCRIPTPARAMETERS=wop

This is because the script has returned “wop” in the configure Function.
Of course the next call of the configure Function would get the parameter “wop”.
Using this inter script communication system the scripter can pass his own parameters around, needed for his desired functionality.

Our script has no settings only a location where the output file will go, but, more on that in the next part.

2- PredictFileOutputNames

This Function is called when the “change” (or RMB>> Properties) function is clicked in the ouput container.

```
Function PredictOutputFileNames(Parameters : String) : String;
```

After this script function is executed the usual folder and filename configuration dialog is displayed using the script provided data.

The script should supply the filename, which is the same as when the data is generated, but does not generate the data function.

[note: don’t use Showmessage to debug here, it is not supported]

In our script, we like our output to have the same name as the original pcbdoc, but now with the extension ‘pcb’ which is the default for a protel 2.8 ASCCI file.

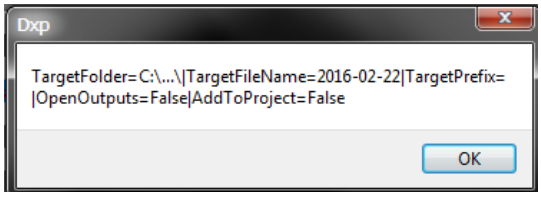
So, all we need to do is predict the filename, we could go get the name of the pcbdoc in the project and strip the ‘’doc”.

```
Function PredictOutputFileNames(Parameters : String) : String;
    Var
    CurrWorkSpace        : IWorkSpace; // An Interface handle to the current workspace
    PCBProject           : IProject;   // An Interface handle to the current Project
    MyServerDocument     : IServerDocument;
    i                    : Integer;
    FileName             : TDynamicString;
Begin
   CurrWorkSpace := GetWorkSpace;
   If CurrWorkSpace = Nil Then Exit;
   PCBProject := GetWorkspace.DM_FocusedProject;
   If PCBProject = Nil Then Exit;
   For i := 0 to PCBProject.DM_LogicalDocumentCount - 1 Do //find the PcbDoc
   Begin
      if PCBProject.DM_LogicalDocuments(i).DM_DocumentKind = 'PCB' Then
      Begin
         MyServerDocument := PCBProject.DM_LogicalDocuments(i);
      End;
   End;
   //Get the filename of the PcbDdoc, and make extension .pcb for the ASCII file
   FileName := MyServerdocument.DM_FileName;
   //Strip the doc from pcbdoc, so its pcb.
   FileName := StringReplace( FileName, 'PcbDoc', 'pcb',1);
   Result := Filename;
End;
```

3-Generate

The Generate Procedure (it has no return value) is where the actual work is done.

```
Procedure Generate(Parameters : String);
```



The parameter which are passes here are all settings for the script

These are delivered using the output job and the settings that ware entered in the “configure” function.

The passed parameters are pretty self-explanatory, we only need to write the code that takes these parameter and acts on it.

```
Procedure Generate(Parameters : String);
Var
    CurrWorkSpace        : IWorkSpace; // An Interface handle to the current workspace
    PCBProject           : IProject;   // An Interface handle to the current Project
    MyServerDocument     : IServerDocument;
    S                    : String;
    Path                 : String;
    i                    : Integer;
    FileName             : TDynamicString;

    //Now the vars that we need to interprent (Outjob supplied)
    AddToProject          : Boolean;
    OpenOutputs           : Boolean;
    TargetFileName        : String;
    TargetFolder          : String;
    TargetPrefix          : String;
Begin
    //initialize the variables
    OpenOutputs          := True;
    AddToProject         := True;
    TargetFileName       := '';
    TargetFolder         := '';
    TargetPrefix         := '';
    //fill local variables with parameter values
    If GetState_Parameter(Parameters, 'OpenOutputs,S) Then OpenOutputs :=
StringsEqual(S, 'True');
    If GetState_Parameter(Parameters, 'AddToProject', S) Then AddToProject :=
StringsEqual(S, 'True');
    If GetState_Parameter(Parameters, 'TargetFileName', S) Then TargetFileName := S;
    If GetState_Parameter(Parameters, 'TargetFolder', S) Then TargetFolder := S;
    If GetState_Parameter(Parameters, 'TargetPrefix', S) Then TargetPrefix := S;

   CurrWorkSpace := GetWorkSpace;
   If CurrWorkSpace = Nil Then Exit;
   PCBProject := GetWorkspace.DM_FocusedProject;
   If PCBProject = Nil Then Exit;
   For i := 0 to PCBProject.DM_LogicalDocumentCount - 1 Do //find the PcbDoc
   Begin
      if PCBProject.DM_LogicalDocuments(i).DM_DocumentKind = 'PCB' Then
      Begin
         MyServerDocument := PCBProject.DM_LogicalDocuments(i);
      End;
   End;
   Path := MyServerdocument.DM_FullPath;
```

```
   If StringsEqual(TargetFileName, '') Then //no filename, then take the existing name
   Begin
      //Get the filename of the PcbDdoc, and make extension .pcb for the ASCII file
      TargetFileName := MyServerdocument.DM_FileName;
      //Strip the extension, we will add it later
      TargetFileName := StringReplace( TargetFileName, '.PcbDoc', '',1);
   End;
   //Close the DOC first or export will fail
   ResetParameters;
   AddStringParameter('ObjectKind','ProjectDocuments');
   RunProcess('WorkspaceManager:CloseObject');
   //Copy the orginal PCBdoc to the new location
   MyServerDocument := Client.OpenNewDocument('PCB',Path,
TargetFolder+TargetPrefix+TargetFileName+'.pcb',TRUE);
   MyServerDocument.DoFileSave('');
   Client.ShowDocument(MyServerDocument);
   //save the copy as ascii
   MyServerDocument.DoFileSave('Protel PCB 2.8 ASCII(*.pcb)');
   //Handle the extra options
   If AddToProject = False Then
PCBProject.DM_RemoveSourceDocument(TargetFolder+TargetPrefix+TargetFileName+'.pcb');
   If OpenOutputs = False Then Client.CloseDocument(MyServerDocument);
   //And add the original back
   PCBProject.DM_AddSourceDocument(Path);
End;
```

Summary

As we have shown in this whitepaper it is really easy to get you very own specific output type integrated into the standard Altium flow