|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| LEIF TRONSTADS PLASS 4  NO-1337 SANDVIKA  TEL. +47 67 57 21 20  FAX. +47 67 57 21 01  **Technical Note**  **Guide for APDL Submit app** | | | | |
| **DATE** | **PREPARED BY** | | **CHECKED BY** | **NO. OF PAGES** |
| **18-Sep-2015** | **Initials** | | **Initials** | **14** |
| C:\Users\1990anli\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E80990AF.tmp | | **DOCUMENT TITLE:**  Guide for APDL Submit app | | |
| **PROJECT NAME:**  CON-18-SIE-008 APDL Application Developement | | |
| **Name of the client** | | **DOCUMENT NUMBER:**  CON-18-SIE-008 APDL Application Developement\_TN01 | | |
| **PREPARED BY:**  Anders Froman Lindeskov, M.Sc. Dipl.ing  Anders.Lindeskov@edrmedeso.com | | | | |

Contents

[1. Introduction 3](#_Toc518650952)

[1.1. Definition and Abbreviations 3](#_Toc518650953)

[2. The installation 4](#_Toc518650954)

[2.1. How to include the script in the Available Extensions. 4](#_Toc518650955)

[3. Using the APDL App. 5](#_Toc518650956)

[4. Adding and duplication multiple APDL submit routine to project. 9](#_Toc518650957)

[5. References 10](#_Toc518650958)

[Appendix A. Source code 11](#_Toc518650959)

[Appendix A.1. XML code 11](#_Toc518650960)

[Appendix A.1.1. Python code 13](#_Toc518650961)

# Introduction

This document is the user guideline for the Siemens Gamesa RE Workflow APDL submit applet for the ANSYS Workbench simulation environment.

The applet has been custom developed to match the Siemens Gamesa RE current file handling and project Workflow in during simulations with customised APDL routines.

## Definition and Abbreviations

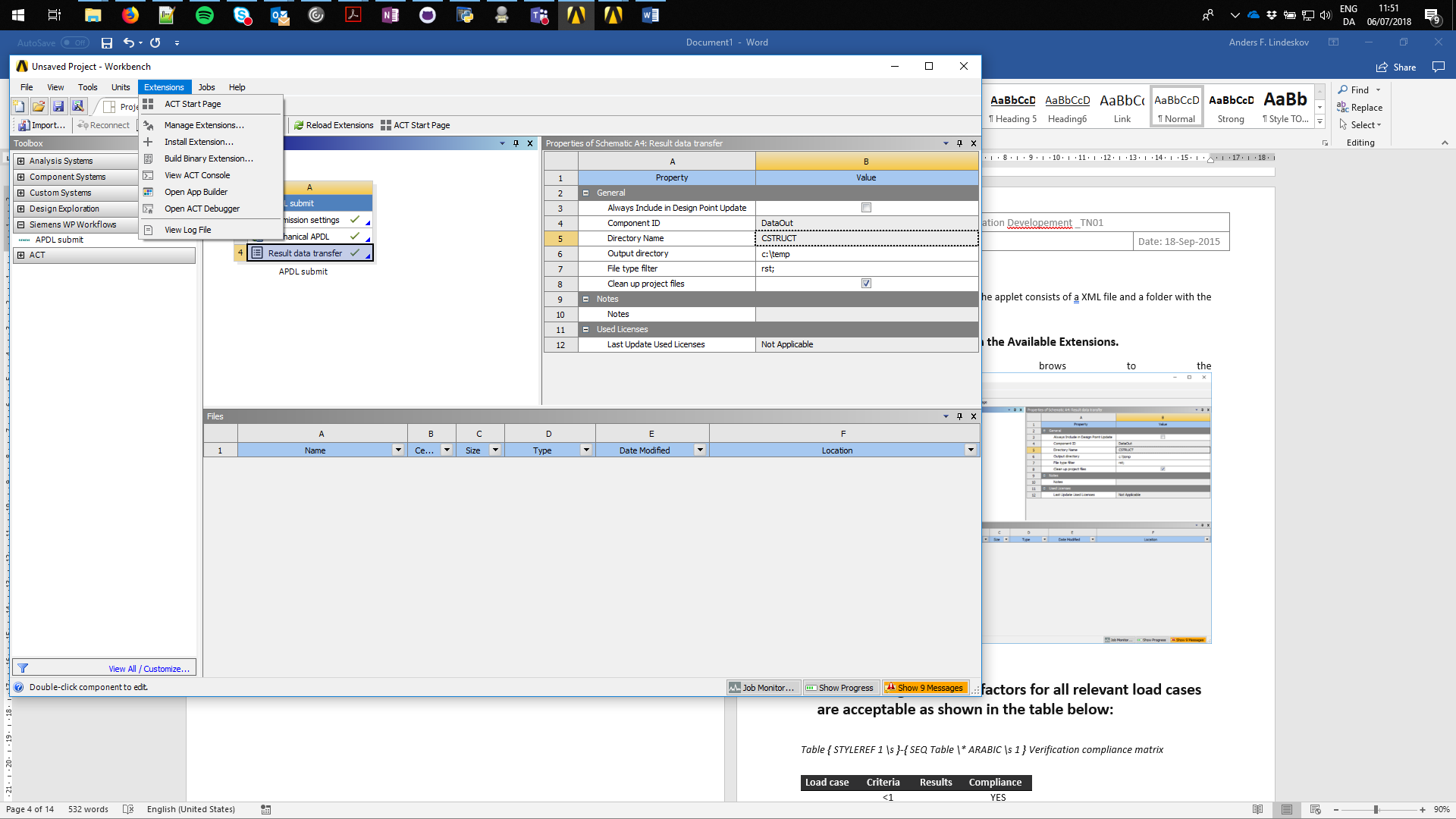
|  |  |
| --- | --- |
| WB | ANSYS Workbench |
| FE | Finite Element |
| N/A | Not Applicable |
| App | Application |

# The installation

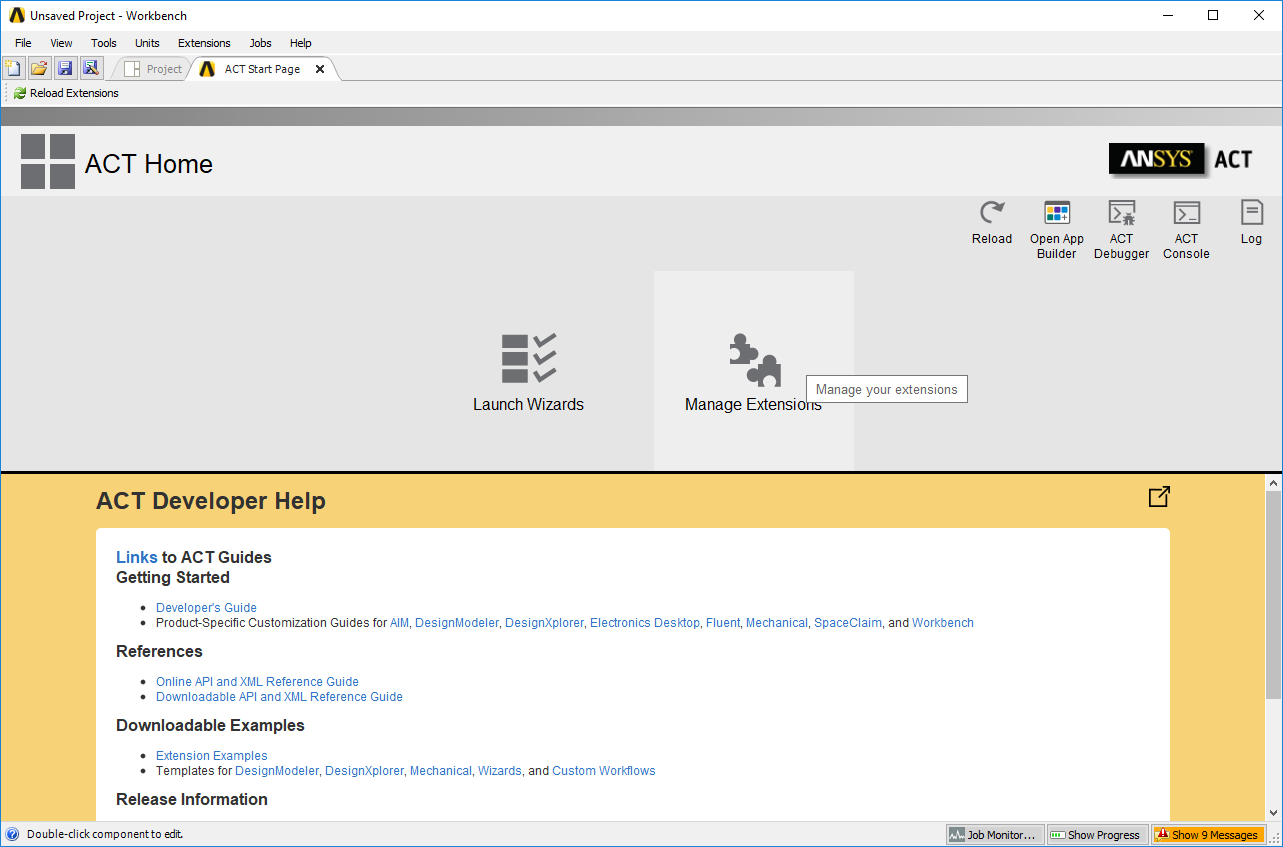
The is provided in script readable from. The applet consists of a XML file and a folder with the same Name.

## How to include the script in the Available Extensions.

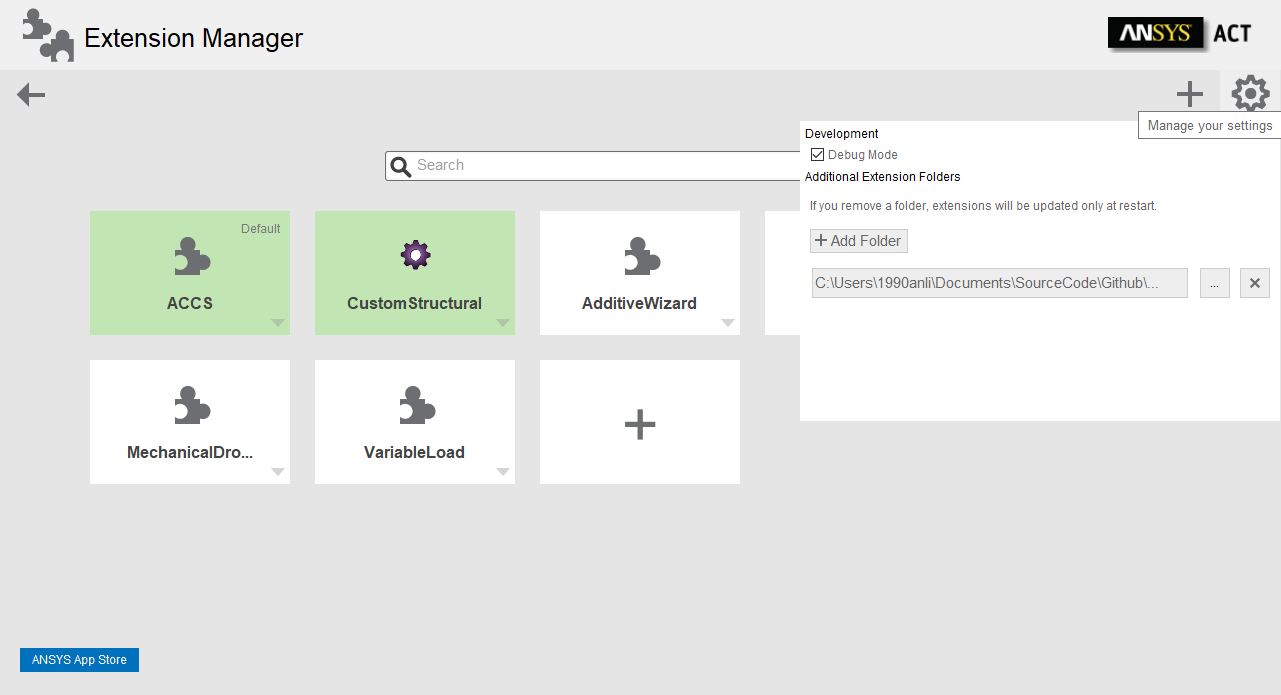
Open WB and brows to the ACT Start Page in the Extensions menu.



Click on the Manage Extensions icon.

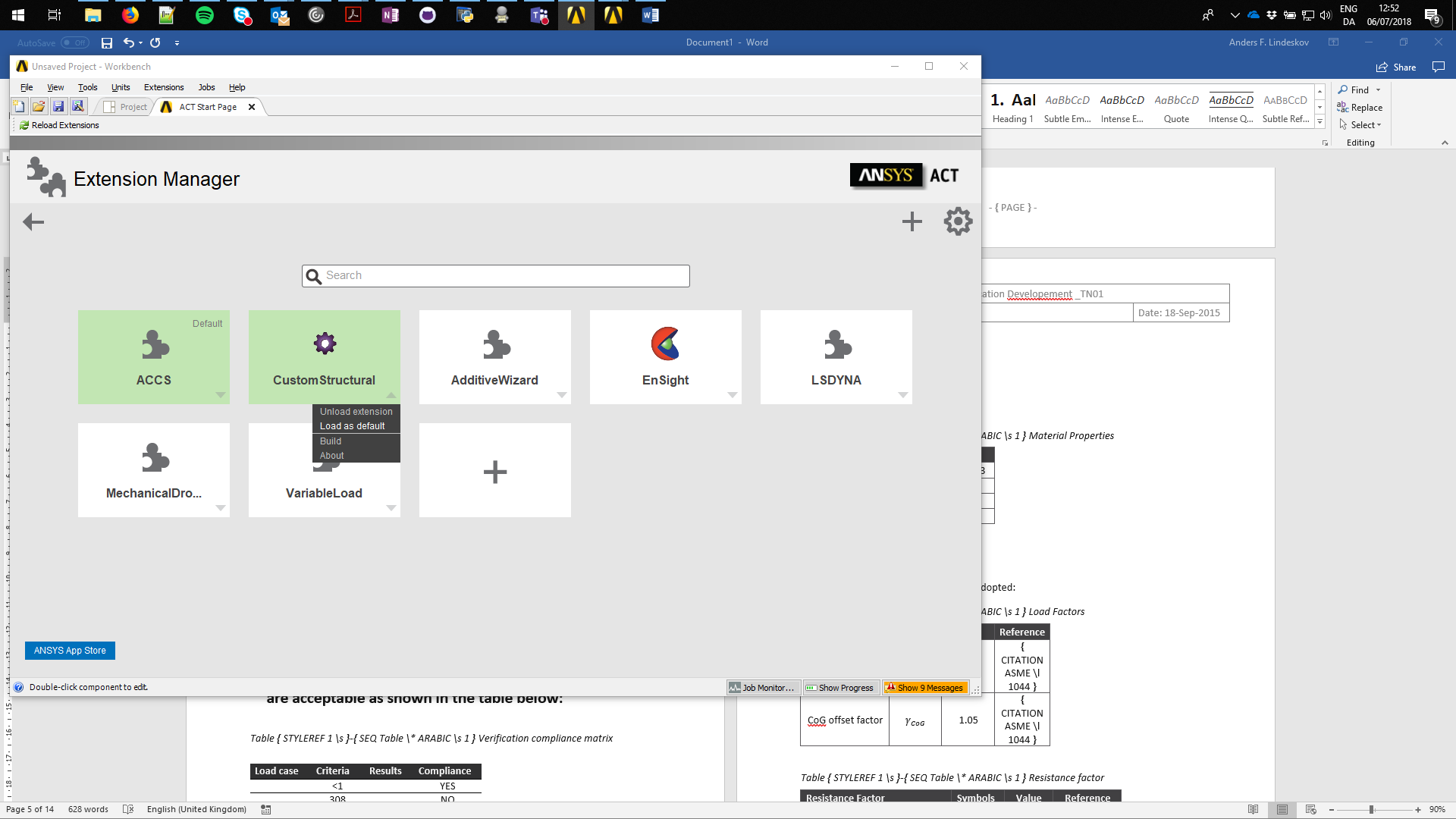


In the Extensions Manager, click on the gear wheel in the upper right corner to add Additional Extension Folders.



Select Add Folder and brows the Personal, local directory storing the XML applet file and folder. Point on the XML and select Open.

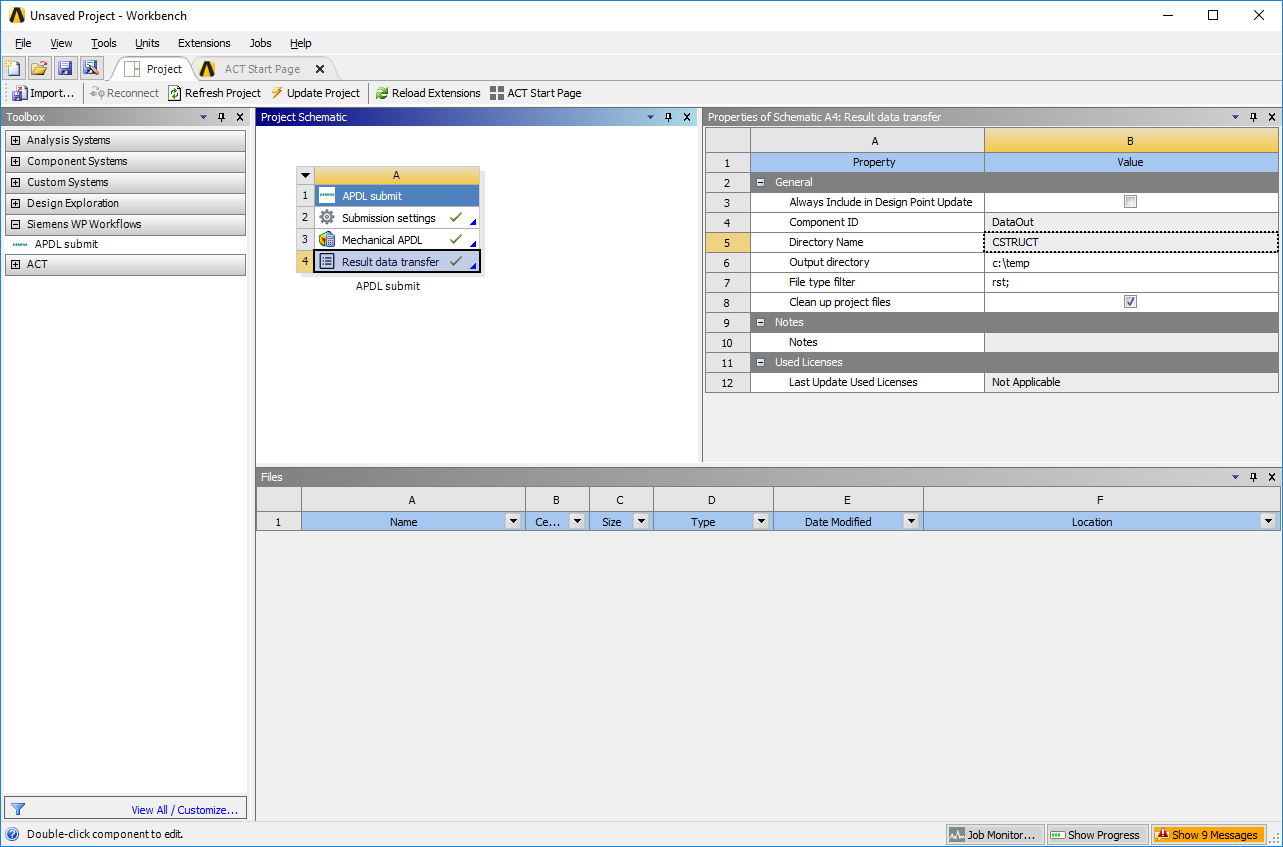
The Custom Structural App should now appear in the overview sheet of the Extension Manager. Click on the Custom Structural app, and it becomes green, indication the App extension is now included in the WB project.



To load the app as default. Select the small downwards arrow in the lower right corner of the Custom Structural app. Navigate in the showing dropdown menu to the Load as Default option.

# Using the APDL App.

After successfully instalment of the App in the Extensions Manager, the Siemens WP Workflow index is visible in the Toolbox menu to the left in the WB main project page overview. Extending the Siemens WP workflow make the APDL submit task routine visible.

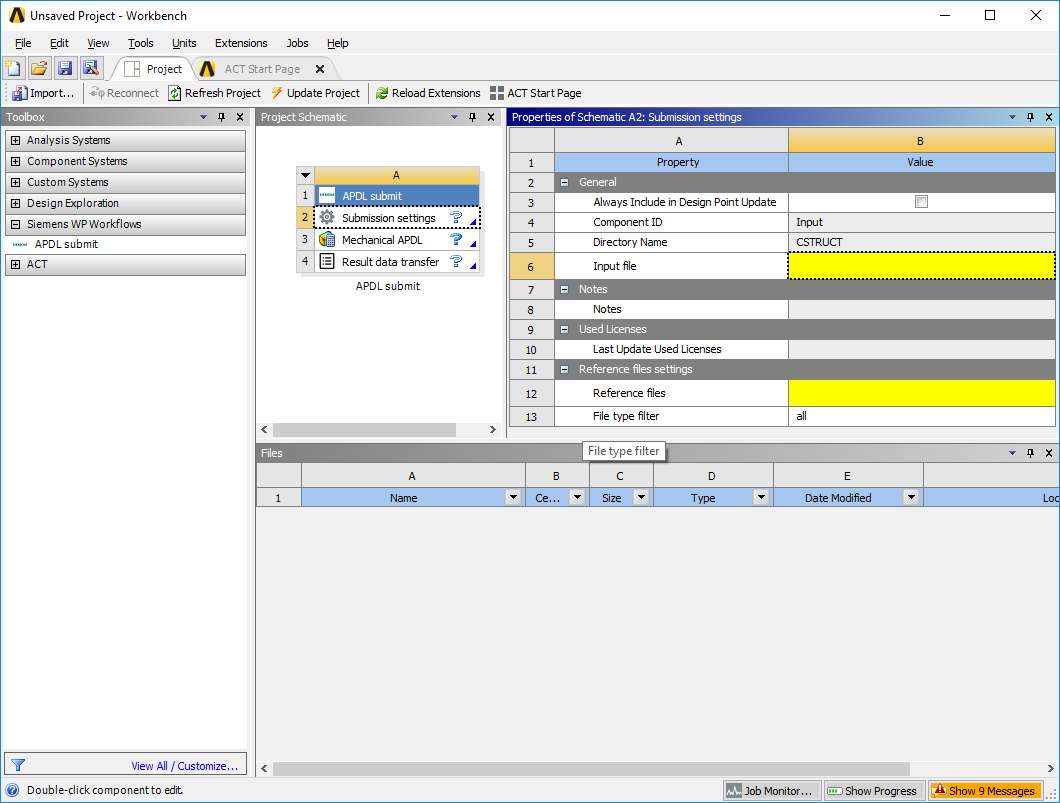


Drag and drop or double click on the task routine to include in the Project workflow page.

The APDL submit consists of three tasks;

1. a submission setting,
2. a Mechanical APDL routine
3. a Result Data transfer.

## Submission Settings

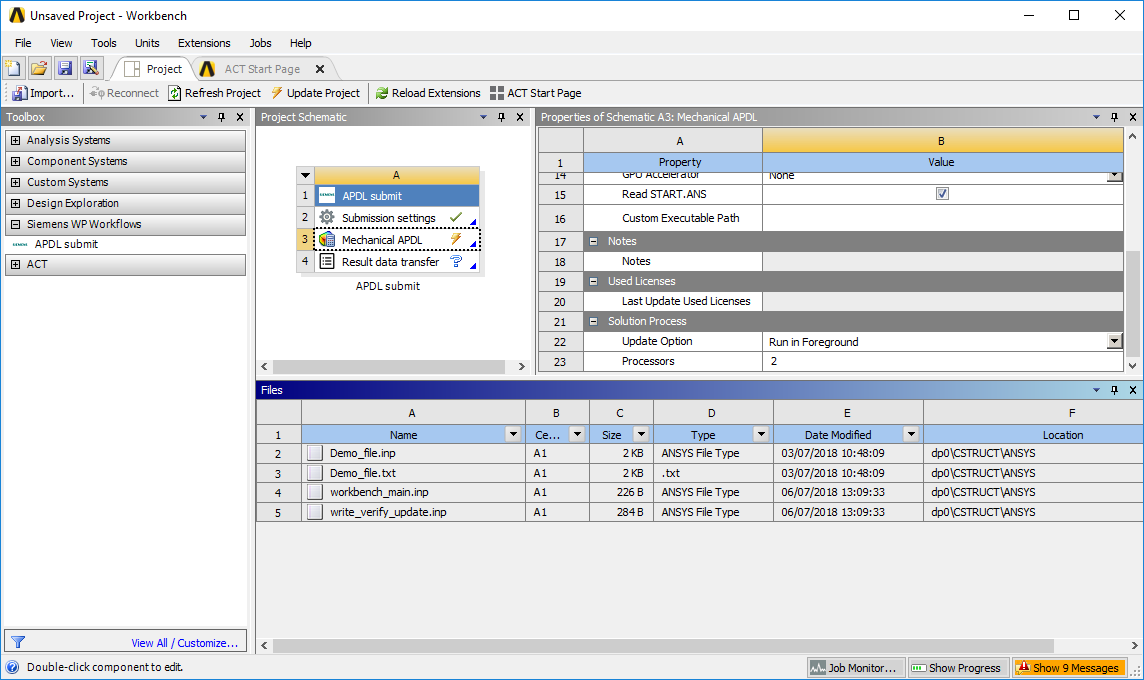


The submission setting needs three user inputs; an input file path (*mandatory*), a references files director location (*Optional*) and optional a reference file filter (*Optional*).

If the Reference files location is not manual set, the script uses the root path of the selected input file. (Flat file structure)

The reference file filters options are displayed below in Table XXX. The filters can be combined with in lists with either comma (“,”) or semi comma (“;”) as delimiter.

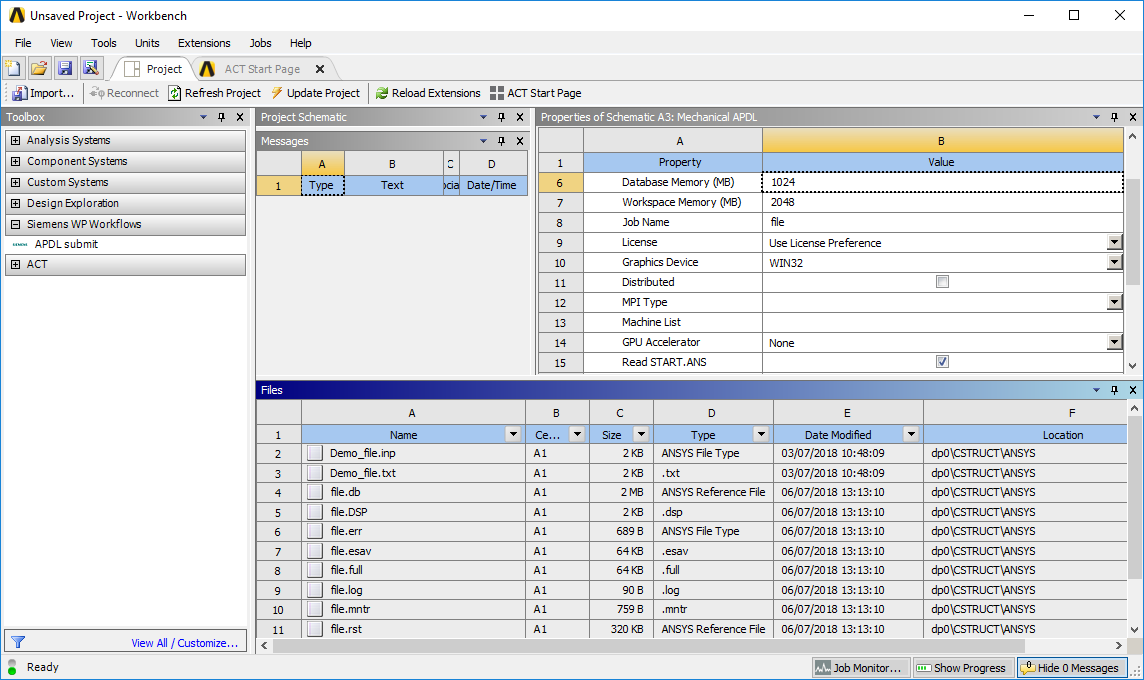
|  |  |
| --- | --- |
| **Reference Filter option** | **Description** |
| 'txt’ | Text files |
| 'inp' | Ansys input files (ASCII format) |
| 'xml' | XML- Extented Meta Language (ASCII format) |
| 'db' | ANSYS Database file (Binary format) |
| 'rst' | ANSYS Result Database file (Binary format) |
| 'log' | ANSYS Log file (ASCII format) |
| 'out' | ANSYS Solver output file (ASCII format) |
| 'all', '\*', '' or '\*.\*’ | All files |



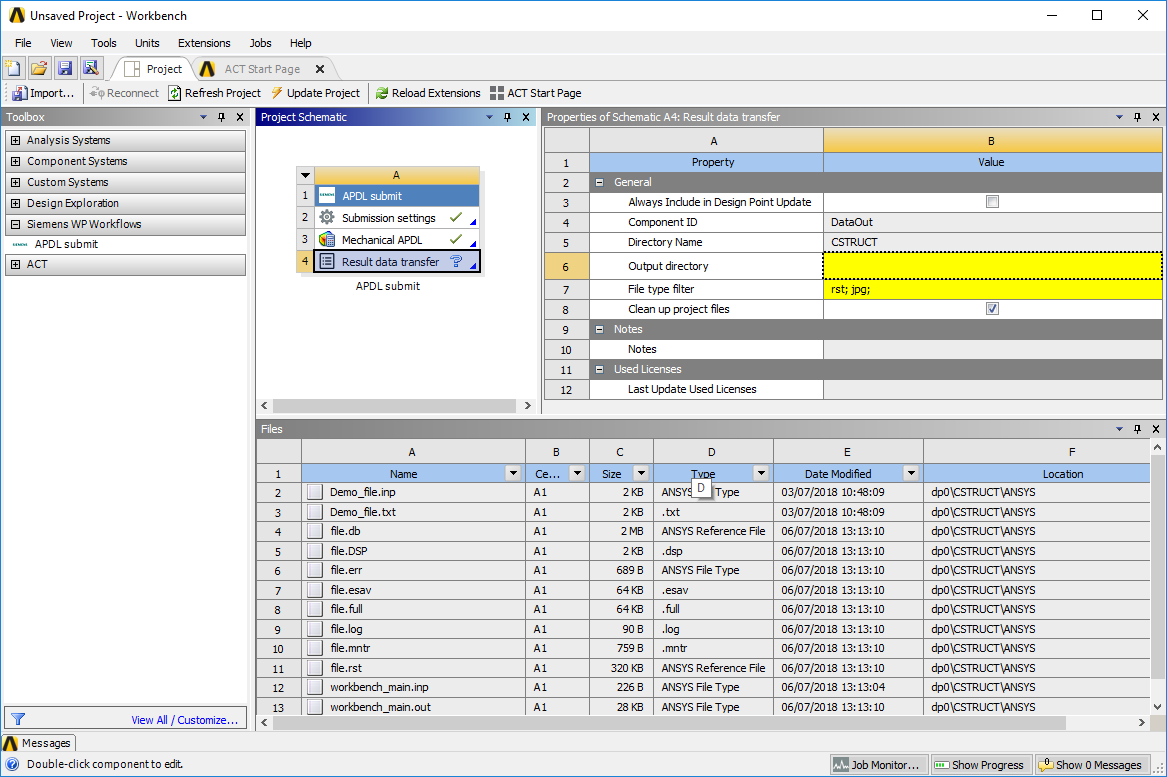
Selecting update task, by right clicking on the task in the included APDL task routine starts the file copying and addition to the current included APDL task group. Added file to the project can be seen in the Files viewer.

The green check mark and the Yellow update needed symbol on the Mechanical APDL indicates the readiness for the job to be submitted to solver.

Customer settings for the Mechanical APDL can be set in the property overview. Updating the Mechanical APDL starts the Solving process. Results are transferred back to the WB project and can by seen in the Files Viewer.



The final task in the APDL submit task routine is to specify the wanted output data storage folder, and the two optional output file filter and deleting of WB project stored result and model files.



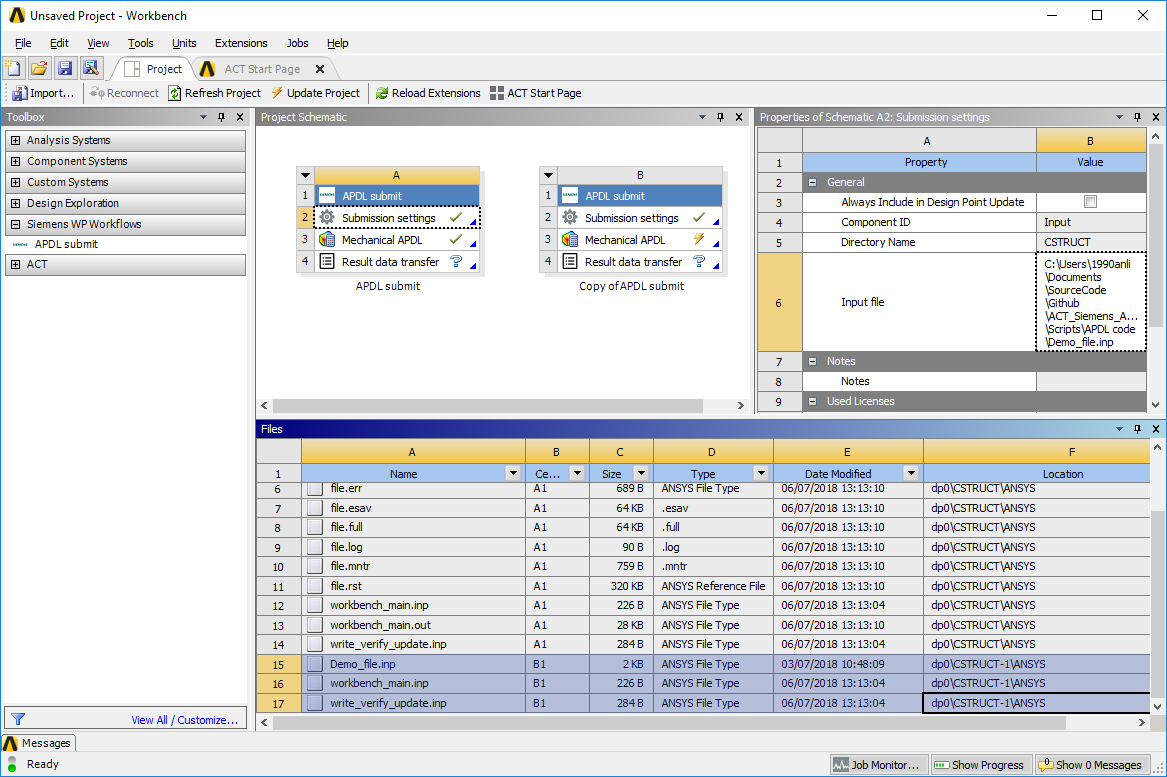
The Output File type filters options are displayed below in Table XXX. The filters can be combined with in lists with either comma (“,”) or semi comma (“;”) as delimiter.

|  |  |
| --- | --- |
| **Reference Filter option** | **Description** |
| 'txt’ | Text files |
| 'rst' | ANSYS Result Database file (Binary format) |
| 'inp' | Ansys input files (ASCII format) |
| 'db' | ANSYS Database file (Binary format) |
| ‘dsp’ | ANSYS Displacement vectors and Matrices (Binary format) |
| 'log' | ANSYS Log file (ASCII format) |
| 'out' | ANSYS Solver output file (ASCII format) |
| ‘err’ | ANSYS Solver error file (ASCII format) |
| ‘full’ | ANSYS matrice file (Binary format) |
| ‘mntr’ | ANSYS material file (Binary format) |
| 'all', '\*' or '' | All files |

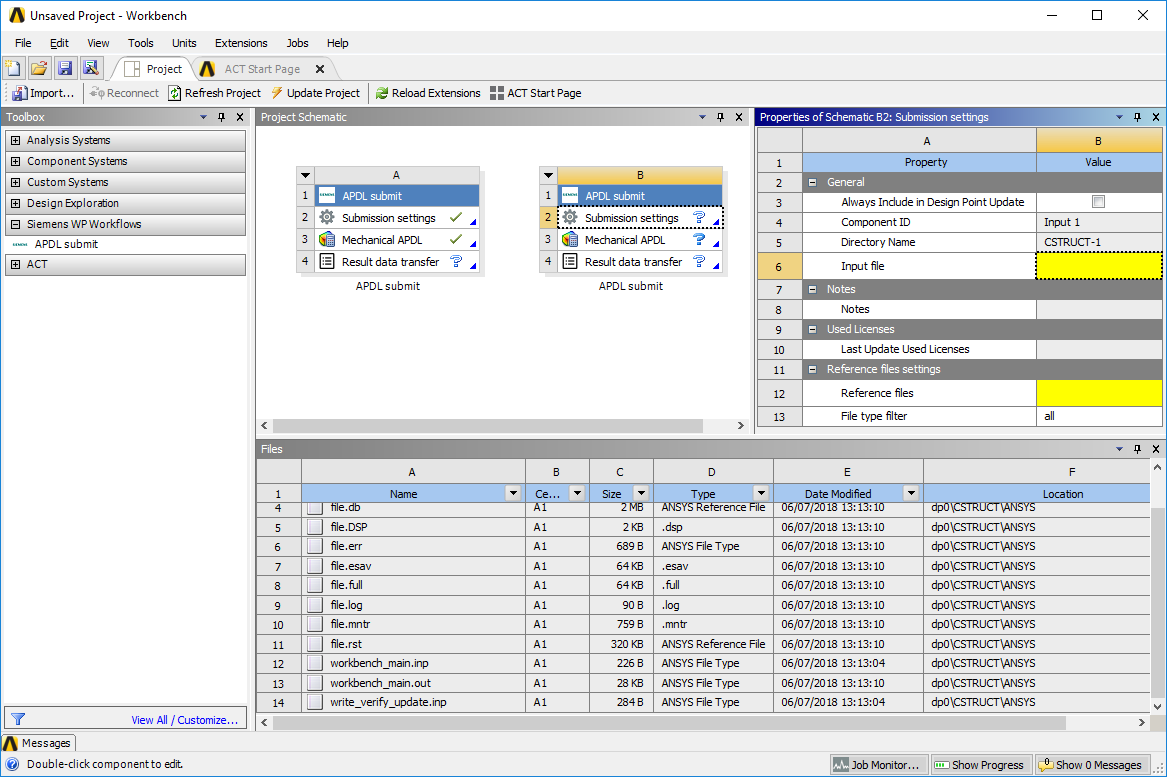
# Adding and duplication multiple APDL submit routine to project.

It is possible to add or make duplicates of the included APDL task routine within the project.

Using the Task Group duplicate option. The result files are not duplicated, only the given reference files and input files. The Mechanical APDL properties are also copied.



Using the add new APDL submit task routine the properties are all reset to default. No files are included, before the user full field the needed properties paths and filers.



# References

1. **ANSYS Inc.** Workbench User's Guide, Release R.18.2. [Online] August 2017. [Cited: 8 July 2018.] https://support.ansys.com/portal/site/AnsysCustomerPortal/template.fss?file=%2Fprod\_docu%2F18.2%2FWorkbench+Users+Guide.pdf.

2. **ANSYS Inc. .** ANSYS ACT Developer's Guide. [Online] August 2017. [Cited: 8 July 2018.] https://support.ansys.com/portal/site/AnsysCustomerPortal/template.fss?file=%2Fprod\_docu%2F18.2%2FANSYS+ACT+Developers+Guide.pdf.

1. Source code
   1. XML code

<extension version="1" name="CustomStructural" icon="images\custom\_structural\_extension.png">

<guid shortid="CustomStructural">69d0095b-e138-4841-a13a-de12238c85f9</guid>

<script src="main.py"/>

<interface context="Project">

<images>images</images>

</interface>

<workflow name="wf6" context="Project" caption="Siemens WP Workflows" version="1">

<callbacks>

<onbeforetaskupdate>onBeforeUpdate</onbeforetaskupdate>

<onaftertaskupdate>onAfterUpdate</onaftertaskupdate>

<onaftertasksourceschanged>onAfterSourcesChanged</onaftertasksourceschanged>

</callbacks>

<tasks>

<task caption="Input" icon="wheel" name="Input" version="1">

<callbacks>

<onupdate>updateInput</onupdate>

<oninitialize>taskinit</oninitialize>

</callbacks>

<property name="SelectInputAPDLFile" caption="Input file" control="fileopen" default="C:\script\input.txt" readonly="False" needupdate="true" visible="True" persistent="True" isparameter="False">

<callbacks>

<OnInit>OnInit</OnInit>

<IsValid>InputFileTypeCheck</IsValid>

<OnValidate>InputFilePathChanged</OnValidate>

</callbacks>

</property>

<propertygroup name="Reference files settings">

<property name="SelectReferenceAPDLdir" caption="Reference files" control="folderopen" default="C:\ script\input.txt" readonly="False" needupdate="true" visible="True" persistent="True" isparameter="False">

<callbacks>

<OnInit>OnInitRef</OnInit>

</callbacks>

</property>

<property name="RefFileFilter" caption="File type filter" control="string" default="all" readonly="False" needupdate="true" visible="True" persistent="True" isparameter="False">

<callbacks>

<IsValid>RefFileFilterCheck</IsValid>

</callbacks>

</property>

</propertygroup>

<!--inputs></inputs-->

<!--outputs></outputs-->

</task>

<task caption="Output" icon="data" name="DataOut" version="1">

<callbacks>

<onupdate>updateOutput</onupdate>

</callbacks>

<property name="OutputDirectory" caption="Output directory" control="folderopen" default="C:\Users\1990erol.CONNECT365\Desktop\Siemens\App\script\" readonly="False" needupdate="true" visible="True" persistent="True" isparameter="False">

<callbacks> <OnInit>OninitOutput</OnInit> <IsValid>OutputFilePathChanged</IsValid>

</callbacks>

</property>

<property name="FileFilter" caption="File type filter" control="string" default="rst; jpg;" readonly="False" needupdate="true" visible="True" persistent="True" isparameter="False">

<callbacks>

<IsValid>OutputFileFilterCheck</IsValid>

</callbacks>

</property>

<property name="DeleteDPs" caption="Clean up project files" control="boolean" default="True" readonly="False" needupdate="true" visible="True" persistent="True" isparameter="False">

<callbacks>

<OnApply>DeleteDP</OnApply>

</callbacks>

</property>

</task>

</tasks>

<taskgroups>

<taskgroup name="CustomStructural" caption="APDL submit" icon="custom\_structural" category="Siemens WP Workflows" abbreviation="CSTRUCT" version="1">

<includeTask caption="Submission settings" name="Input"/>

<includeTask external="True" name="ANSYSSetupCellTemplate" caption="Mechanical APDL"/>

<includeTask caption="Result data transfer" name="DataOut"/>

</taskgroup>

</taskgroups>

</workflow>

</extension>

* + 1. Python code

import os, shutil

import System

#from System.IO import Directory, Path

import clr

import string

#Adding the uniqe ANSYS API's

clr.AddReference("Ans.UI.Toolkit")

clr.AddReference("Ans.UI.Toolkit.Base")

from Ansys.UI.Toolkit import \*

global filepath

global activedirc

class Settings:

def \_\_init\_\_(self):

self.filepath = ""

self.activedirc = ""

def \_\_init\_\_(context):

setting = Settings()

ExtAPI.Log.WriteMessage("Init CustomStructural...")

return True

#-------------Input -----------------------------

def updateInput(task):

""" Running these lines of code when the update task in WB is activated.

The aggrument task is a UserTask

"""

ExtAPI.Log.WriteMessage('updating task ' + task.Name)

#Finding ObjectTest directory in WB project task when Updating.

if 'Input' in task.Name:

ExtAPI.Log.WriteMessage('Updating the Input task')

inputfilepath = task.Properties['SelectInputAPDLFile'].Value

inputReferencePath = task.Properties[1].Properties['SelectReferenceAPDLdir'].Value

inputRefFilters = task.Properties[1].Properties['RefFileFilter'].Value

#mapdlInputFile1 = setup1.AddInputFile(FilePath="") #removing peviously set input files

#Setting the filepath in the MAPDL setup, scirpt file is now move to the dp0 directory if the project.

#changing the directory path seperator - DAMMM Windows

#Becasue of the UserTask structure, it does not directly link back to the Automated ACT Taskgroup

#Containing the Extrenal Task Mechanical APDL

system1 = task.TaskGroup.InternalObject #this returns some thing like: /Schematic/System:CSTRUCT

system1Name = system1.Name #now we have the abbreviation "CSTRUCT" of the Automated TaskGroup

system1 = GetSystem(Name=system1Name)

setup1 = system1.GetContainer(ComponentName="Setup")

ExtAPI.Log.WriteMessage('Task setup1 name: ' + setup1.Name)

if os.path.isfile(inputfilepath):

#Adding input file

mapdlInputFile1 = setup1.AddInputFile(FilePath=inputfilepath)

#Adding reference files

APDLfilePath = task.ActiveDirectory

APDLfilePath = os.path.dirname(APDLfilePath)

#APDLfilePath = os.path.abspath(APDLfilePath.replace('\\','/'))

APDLfilePath = APDLfilePath + '\\ANSYS'

ExtAPI.Log.WriteMessage('Info: APDLfilePath dirc :' + APDLfilePath)

#Get filters

filtersList = inputRefFilters.split(';')

#Starts the copying of the Outputfiles

if not copyFiles(filtersList, inputReferencePath, APDLfilePath, False):

ExtAPI.Log.WriteMessage('Error: Error ourcurred in Copying of reference files')

return False

#Making sure that the input file is still the selected input file

#mapdlInputFile1 = setup1.AddInputFile(FilePath=

mapdlInputFile1 = setup1.AddInputFile(FilePath=inputfilepath)

Settings.filepath = inputfilepath

ExtAPI.Log.WriteMessage('Mapdl Input file set automatically')

return True

else:

ExtAPI.Log.WriteMessage('ERROR: Input file at:' + inputfilepath + 'not found')

ExtAPI.Log.WriteMessage('Warning: Mapdl Input file not set automatically:')

return False

def taskinit(task):

""" Initialing the input task in WB """

ExtAPI.Log.WriteMessage('Task initialized: ' + task.Name)

#Setting a node maybe

#Setting a default path (previous used

return True

def OnInit(entity, property):

#Setting default path in the Selection of Input file

property.Value = "" #Setting a blank input file

ExtAPI.Log.WriteMessage('Info: The input filepath has been reset')

return True

def InputFileTypeCheck(entity, property):

#Check the chosen file type is correct.

if property.Value != "":

filepath = property.Value

#print filepath

#ExtAPI.Log.WriteMessage('InputFileTypeCheck property.Value : ' + filepath)

#os.path.splitext()

fileext = System.IO.Path.GetExtension(filepath)

if fileext in [".cdb",".inp", ".db", ".txt"]:

#ExtAPI.Log.WriteMessage('Seleted ADPL File supported, file extension: ' + fileext)

if InputFilePathChanged(entity, property):

return True

else:

return False

else:

ExtAPI.Log.WriteMessage('Seleted ADPL input File type not supported, file extension: ' + fileext)

return False

else:

return False

def InputFilePathChanged(entity, property):

#Called when the user changes the filepath chacking the access to the file

#global filepath

if property.Value != "":

filepath = property.Value

full\_filepath = System.IO.Path.GetFullPath(filepath)

if System.IO.File.Exists(full\_filepath):

#ExtAPI.Log.WriteMessage('Info: Seleted ADPL File Found')

return True

else:

#ExtAPI.Log.WriteMessage('Warning: Seleted ADPL File not Found')

return False

def OnInitRef(entity, property):

#Setting the reference file path to input file directory

property.Value = ""

return True

def RefFileFilterCheck(entity, property):

#Check the selected filter setting for the reference filter

if filterCheck(property, filterString=['txt','inp','xml','db','rst','log','out','all','\*','','\*.\*']):

return True

else:

return False

#------------- MADPL -----------------------------

def updateMapdl(task):

#Updating the MAPDL task in the TaskGroup - NOT USED - Can not get at Custom callback from a external task

Settings.activedirc = task.ActiveDirectory

system1 = task.TaskGroup.InternalObject #this returns some thing like: /Schematic/System:CSTRUCT

system1Name = system1.Name #now we have the abbreviation "CSTRUCT" of the Automated TaskGroup

system1 = GetSystem(Name=system1Name)

input1 = system1.GetContainer(ComponentName="Input")

inputfilepath = input1.Properties['SelectInputAPDLFile'].Value

inputReferencePath = input1.Properties['SelectReferenceAPDLdirc'].Value

inputRefFilters = input1.Properties['RefFileFilter'].Value

ExtAPI.Log.WriteMessage('Info: inputReferencePath dirc :' + inputReferencePath)

#Adding reference files

APDLfilePath = setup1.ActiveDirectory

APDLfilePath = os.path.dirname(APDLfilePath)

#APDLfilePath = os.path.abspath(APDLfilePath.replace('\\','/'))

APDLfilePath = APDLfilePath + '\\ANSYS'

ExtAPI.Log.WriteMessage('Info: APDLfilePath dirc :' + APDLfilePath)

#if not System.IO.Directory.Exists(APDLfilePath):

# ExtAPI.Log.WriteMessage('Info: Directory not found :' + str(APDLfilePath))

# return False

#ExtAPI.Log.WriteMessage('Info: files in ActiveMAPDL dirc :' + str(files))

#Get filters

filtersList = inputRefFilters.split(';')

#Starts the copying of the Outputfiles

if not copyFiles(filtersList, inputReferencePath, APDLfilePath, DeleteFilesInDP=False):

ExtAPI.Log.WriteMessage('Error: Error ourcurred in Copying of reference files')

return False

"""

Reffiles=[]

for (path, dirc, files) in os.walk(inputReferencePath):

temp = System.IO.Path.Combine(path, file) #full path til the file

if not temp == inputfilepath:

Reffiles.append(temp) #sourcefile

ExtAPI.Log.WriteMessage('Info: Adding file :' + temp)

setup1.AddFile(FilePath=temp)

else:

ExtAPI.Log.WriteMessage('Info: Input file allready added:' + temp)

"""

#Adding input file

setup1 = system1.GetContainer(ComponentName="Setup")

if System.IO.File.Exists(inputfilepath):

mapdlInputFile1 = setup1.AddInputFile(FilePath=inputfilepath)

return True

#------------- Output -----------------------------

def updateOutput(task):

"""updating the Data out task in the TaskGroup"""

#Getting the Mapdl for this TaskGroup Active directory

#property.Value = ""

ExtAPI.Log.WriteMessage('Info: Updating the Output task')

systemName = task.TaskGroup.InternalObject.Name

ExtAPI.Log.WriteMessage('SystemName : ' + systemName)

setup1 = ExtAPI.DataModel.GetTaskGroupById(systemName).Tasks[1]

activeDir = setup1.ActiveDirectory

ExtAPI.Log.WriteMessage('Info: ActiveMAPDL dirc :' + activeDir)

APDLfilePath = setup1.ActiveDirectory

APDLfilePath = os.path.dirname(APDLfilePath)

#APDLfilePath = os.path.abspath(APDLfilePath.replace('\\','/'))

APDLfilePath = APDLfilePath + '\\ANSYS'

ExtAPI.Log.WriteMessage('Info: APDLfilePath dirc :' + APDLfilePath)

if not System.IO.Directory.Exists(APDLfilePath):

ExtAPI.Log.WriteMessage('Info: Directory not found :' + str(APDLfilePath))

return False

ExtAPI.Log.WriteMessage('Info: files in ActiveMAPDL dirc :' + str(files))

#get filters

filters = task.Properties['FileFilter'].Value

filtersList = filters.split(';')

# starts the copying of the Outputfiles

DataOut = ExtAPI.DataModel.GetTaskGroupById(systemName).Tasks[2]

DataOutDirc = DataOut.Properties['OutputDirectory'].Value

DeleteFilesInDP = DataOut.Properties['DeleteDPs'].Value

if copyFiles(filtersList, APDLfilePath, DataOutDirc, DeleteFilesInDP):

return True

else:

return False

def OninitOutput(entity, property):

#empty the output directory path

ExtAPI.Log.WriteMessage('Info: initializing Output: ')

property.Value = ""

return True

def OutputFileFilterCheck(entity, property):

#Checking the selected filter types - Receiving a String like jpg;rst;...,all

if filterCheck(property, filterString=['rst','inp','dsp','err','esav','full','log','mntr','out','all','\*','']):

return True

else:

return False

def OutputFilePathChanged(entity, property):

#checking the path of the selected output file directory

if property.Value != "":

if not System.IO.Directory.Exists(property.Value):

ExtAPI.Log.WriteMessage('The Output directory :' + property.Value + 'was not found, files are not copied')

#ExtAPI.Log.WriteMessage('INFO: Output file located at :' + Settings.activedirc)

return False

else:

#ExtAPI.Log.WriteMessage('INFO: Output file located at :' + Settings.activedirc)

return True

else:

return False

def DeleteDP(entity, property):

pass

return True

#------------- Global WB -----------------------------

def onBeforeUpdate(task):

#Call before a global project update is done

msg = getPrintMessage('pre-update', task)

print msg

return True

def onAfterUpdate(task):

#called after a successfull completion of all task in a TaskGroup, continues though all TaskGroups in project

msg = getPrintMessage('post-update', task)

print msg

return True

def onAfterSourcesChanged(task):

#called after a change in one of the tasks in a TaskGroup, continues though all TaskGroups in project

msg = getPrintMessage('Task changed', task)

print msg

return True

def getPrintMessage(msg, task):

taskName = 'none'

if task != None:

taskName = task.Name

return 'in ' + msg + ' callback for task ' + task.Name

#global filepath

#------------ General --------------------------

def filterCheck(property, filterString=['all']):

out = []

#ExtAPI.Log.WriteMessage('Info: Seleted ...: ' + str(property.Value))

filters = str(property.Value)

filters = filters.replace(',',';') #changing , to ;

filters = filters.replace(' ','') # removing white spaces

filtersList = filters.split(';')

for filter in filtersList:

filter = filter.lower()

#ExtAPI.Log.WriteMessage('File Filter : ' + filter)

if filter == '\*.\*':

out.append('True')

else:

filter = filter.replace('.','')

if filter in filterString:

out.append('True')

else:

out.append('False')

#property.Value = filters

#ExtAPI.Log.WriteMessage('filter list Out : ' + str(out))

if 'False' in out:

#ExtAPI.Log.WriteMessage('False found in list Out, returning False')

return False

else:

return True

def copyFiles(filtersList, InDirc, OutDirc, DeleteFiles=False):

#Copying of files

for (path, dirc, files) in os.walk(InDirc):

pass

#filter files list

FilesToCopy = []

if '\*' or '\*.\*' or 'all' in filtersList: #all files

FilesToCopy = files

else:

for file in files: # some of the files

if System.IO.Path.GetExtension(file)[1:] in filtersList:

FilesToCopy.append(file)

for file in FilesToCopy:

ExtAPI.Log.WriteMessage('Info: file found ActiveMAPDL dirc :' + str(file))

fullPath\_source = System.IO.Path.Combine(path, file) #sourcefile

fullPath\_target = System.IO.Path.Combine(OutDirc, file) #targetfile

ExtAPI.Log.WriteMessage('Info Source :' + fullPath\_source + ' Target :' + fullPath\_target)

#copying files in stored list

if System.IO.File.Exists(fullPath\_source):

System.IO.File.Copy(fullPath\_source, fullPath\_target)

ExtAPI.Log.WriteMessage('Info: file copied:')

#Deleting Source file in Dp0

if DeleteFiles:

ExtAPI.Log.WriteMessage('Info: file delete at Source as Requested:')

System.IO.File.Delete(fullPath\_source)

#ExtAPI.Log.WriteMessage('Updating the DataOut routine')

return True