Public ExecDrive

Dim stepsCountFromFunctionsSheet, stepsCountFromFnStepCountSheet, fnInTheSheet

Dim strObjectHirearchy, strObject

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: Intializing Libraries, Object Repositories

' Relative Paths, Global Dictionary, Execution Logs

' Inputs: NONE

'

' Returns: True - If all the variables were intialized successfully

' False - If any of the initiaization is failed

'

' Author: Ritesh Kawadkar

' Cretesd At: 4/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Public Sub MirrorIntialize()

'Fetch Relative PATH

'Get the Root folder of the Test so that we can make use of relative paths.

Dim x : x = Instr(Environment.Value("TestDir"),"Mirror - MainDriver")-2

ExecDrive = mid(Environment.Value("TestDir"),1,x)

' Get the path of Libraries using relative to the current Drive

GlobalDictionary("LibPath") = ExecDrive+"\Libraries\"

GlobalDictionary("ORPath") = ExecDrive+"\OR\"

GlobalDictionary("TestDataPath") = ExecDrive+"\TestData\TestSuite.xlsm"

GlobalDictionary("ExecutionLogsPath") = ExecDrive+"\Reporting\"

' Dynamically Load the All the Libraries

Set oFSO = CreateObject("Scripting.FileSystemObject")

'Create execution logs

Dim SnapPath, cTime, currTime, currDate, Timestr, ilen, folderexist

cTime = Split(Replace(Time,":","\_")," ", -1, 1)

currDate = Replace(Date,"/","\_")

ilen = (Len(cTime(0))-3)

Timestr = Left(cTime(0),ilen)

GlobalDictionary("currentDate") = currDate + " " + Timestr

GlobalDictionary("SnapPath") = GlobalDictionary("ExecutionLogsPath") + "\ExecutionReports\TestResult\_" + GlobalDictionary("currentDate")

If oFso.FolderExists(CStr(GlobalDictionary("SnapPath"))) Then

'dont create one

Else

oFso.CreateFolder(CStr(GlobalDictionary("SnapPath")))

End If

Set oFSO = Nothing

WriteLog "INFO","MirrorIntialize - Executed"

End Sub

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: It takes one agrument as function name and finds

' where it is located

' Inputs: fnName: Name of the functions which needs to be searched

'

' Returns: Sheet Name where the function is located

'

'

' Author: Ritesh Kawadkar

' Cretesd At: 5/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Public Function getTheSheetNameFromFnStepCountSheet(fnName)

Set objSheetName = new Connections

objSheetName.fnConnectToXL(GlobalDictionary("TestDataPath"))

strQuery = "Select SheetReferrence from [Function Referrences$] where Functions=""" & fnName & """"

objSheetName.fnExecuteQuery(strQuery)

On Error Resume Next

getTheSheetNameFromFnStepCountSheet = objSheetName.objRecordSet.Fields(0).Value

If err.description = "Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." and err.number = 3021 Then

WriteLog "ERROR",fnName & " - Function is not present in Function Referrences sheet...Exititng"

Set objSheetName = Nothing

ExitTest

End If

On Error GoTo 0

Set objSheetName = Nothing

End Function

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: Validates if the function is present in the

' Function Sheet or not

' Inputs: fnName: Name of the functions which needs to be searched

' fnInTheSheet: Sheet name where function has to be searched

' Returns: True - If the function is found the the sheet

' False - If the function is not found the the sheet

'

' Author: Ritesh Kawadkar

' Cretesd At: 5/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Public Function ValidateFunctionPresenceInFnSheet(fnName, fnInTheSheet)

Set objFunction = new Connections

objFunction.fnConnectToXL(GlobalDictionary("TestDataPath"))

strQuery = "Select Count(\*) from [" & fnInTheSheet & "$] where Functions=""" & fnName & """"

objFunction.fnExecuteQuery(strQuery)

stepsCountFromFunctionsSheet = objFunction.objRecordSet.Fields(0).Value

If objFunction.objRecordSet.Fields.Count = 1 Then

ValidateFunctionPresenceInFnSheet = True

Else

WriteLog "ERROR",fnName & " - Function is not present in Function\_XXX sheet...Exititng"

ValidateFunctionPresenceInFnSheet = False

ExitTest

End If

Set objFunction = Nothing

End Function

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: Validates the no of steps a function have listed

' in "Functions Referrence" Sheet

' Inputs: fnName: Name of the functions which needs to be searched

'

' Returns: True - If the function has more than 0 steps

' False - If the function has no steps

'

' Author: Ritesh Kawadkar

' Cretesd At: 5/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Public Function ValidateStepsCountInFnStepCountSheet(fnName)

Set objFnStepsCount = new Connections

objFnStepsCount.fnConnectToXL(GlobalDictionary("TestDataPath"))

strQuery = "Select StepsCount from [Function Referrences$] where Functions=""" & fnName & """"

objFnStepsCount.fnExecuteQuery(strQuery)

stepsCountFromFnStepCountSheet = objFnStepsCount.objRecordSet.Fields(0).Value

If stepsCountFromFnStepCountSheet >0 Then

ValidateStepsCountInFnStepCountSheet = True

Else

WriteLog "ERROR",fnName & " - Function does not have any steps...Exititng"

ValidateStepsCountInFnStepCountSheet = False

ExitTest

End If

Set objFnStepsCount = Nothing

End Function

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: Execute the function's steps listed in Functions sheets

'

' Inputs: fnName: Name of the functions which needs to be searched

' fnInTheSheet: Sheet name where function has to be searched

' Returns: True - If all the sub functions executed successfully

' False - If any the sub functions execution failed

'

' Author: Ritesh Kawadkar

' Created At: 6/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Public Function ExecuteAllStepsInFunction(fnName, fnInTheSheet, TS\_Name, stepname, TC\_Name)

iRetVal=True

flag=True

Set objFnExecute = new Connections

objFnExecute.fnConnectToXL(GlobalDictionary("TestDataPath"))

strQuery = "Select \* from [" & fnInTheSheet & "$] where Functions=""" & fnName & """"

objFnExecute.fnExecuteQuery(strQuery)

countOfTS = 1

'Read all the Key/value pair and store in dictionary

Do Until objFnExecute.objRecordSet.EOF

stepDescription = stepname & "\_" & countOfTS & "\_DESC\_"

For each oField in objFnExecute.objRecordSet.Fields

If NOT GlobalDictionary.Exists(oField.Name) Then

GlobalDictionary.Add oField.Name, oField.Value

Else

GlobalDictionary(oField.Name) = oField.Value

End If

Next

'Check for the Execution Flag for Each Step

If GlobalDictionary("Execute Step")="Y" Then

strObjectHirearchy = GlobalDictionary("Object")

strKeyword = GlobalDictionary("Action")

strParam1 = fetchParamValue(GlobalDictionary("Param\_1"), TC\_Name, TS\_Name)

strParam2 = fetchParamValue(GlobalDictionary("Param\_2"), TC\_Name, TS\_Name)

strParam3 = fetchParamValue(GlobalDictionary("Param\_3"), TC\_Name, TS\_Name)

strParam4 = fetchParamValue(GlobalDictionary("Param\_4"), TC\_Name, TS\_Name)

strScreenShot = GlobalDictionary("ScreenShot")

strFunctionName = GlobalDictionary("Functions")

strTCStepID = GlobalDictionary("Step\_ID")

strControls = GlobalDictionary("Controls")

WriteLog "STEP", GlobalDictionary("Step\_ID")

If strObjectHirearchy = "User Defined Functions" Then

Select Case strKeyword

Case "VerifyText" iRetVal = Keyword\_verifyText(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "InStr" iRetVal = Keyword\_inStr(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "AssertText" iRetVal = Keyword\_assertText(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "AssertNull" iRetVal = Keyword\_assertNull(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "VerifyNull" iRetVal = Keyword\_verifyNull(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "ExecuteQTPCommand" iRetVal = Keyword\_executeQTPCommand(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Run" iRetVal = Keyword\_run(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Wait" iRetVal = Keyword\_wait(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "DateDiff" iRetVal = Keyword\_dateDiff(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "RemoveTimeFromApprovals" iRetVal = Keyword\_removeTimeFromApprovals(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "BuildApprovalFormat" iRetVal = Keyword\_buildApprovalFormat(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Maths" iRetVal = Keyword\_maths(strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case Else msgbox strKeyword & " not found"

iRetVal = False

End Select

Else

i = InStr(StrReverse(Replace(strObjectHirearchy, " > ",".")), ".")

If InStr(1, strObjectHirearchy, ">") > 1 Then

strObject = StrReverse(Mid(StrReverse(Replace(strObjectHirearchy, " > ",".")), 1,i-1))

End If

If InStr(1, strObjectHirearchy, ">") > 1 Then

ii = InStr(StrReverse(strObjectHirearchy), " > ")

strParent = Replace(Replace(strObjectHirearchy, Right(strObjectHirearchy, ii-1), ""), " > ", ".")

Else

strParent=""

strObject=strObjectHirearchy

End If

Select Case strKeyword

'Check for the Keyword and execute the function accordingly

Case "Click" iRetVal = Keyword\_click(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "DblClick" iRetVal = Keyword\_dblClick(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Drag" iRetVal = Keyword\_drag(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Drop" iRetVal = Keyword\_drop(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Highlight" iRetVal = Keyword\_highlight(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Set" iRetVal = Keyword\_set(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "SetFocus" iRetVal = Keyword\_setFocus(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Select" iRetVal = Keyword\_select(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "SelectListItem" iRetVal = Keyword\_selectListItem(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Type" iRetVal = Keyword\_type(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "SetSecure" iRetVal = Keyword\_setSecure(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "GetROProperty" iRetVal = Keyword\_getROProperty(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "SetTOProperty" iRetVal = Keyword\_setTOProperty(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Exists" iRetVal = Keyword\_exists(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "IfExistsThenClick" iRetVal = Keyword\_ifExistsThenClick(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "SendKeys" iRetVal = Keyword\_sendKeys(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "Activate" iRetVal = Keyword\_activate(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "IsChecked" iRetVal = Keyword\_isChecked(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "IsReadOnly" iRetVal = Keyword\_isReadOnly(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "ValidateModifiedByAt" iRetVal = Keyword\_validateModifiedByAt(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case "ListCount" iRetVal = Keyword\_listCount(strObject, strParent, strParam1, strParam2, strParam3, strParam4, strScreenShot, TS\_Name, strFunctionName, strTCStepID, stepDescription, TC\_Name)

Case Else WriteLog "KEYWORD FAILED", strKeyword & " - Not found"

iRetVal = False

End Select

End If

GlobalReporting(stepname & "\_" & countOfTS)= iRetVal

Else

GlobalReporting(stepDescription)="Step skipped"

GlobalReporting(stepname & "\_" & countOfTS)= "Skipped"

iRetVal = True

End If

countOfTS = countOfTS +1

objFnExecute.objRecordSet.MoveNext

If iRetVal=False Then

ExecuteAllStepsInFunction= iRetVal

If strKeyword="AssertText" Then

'dont Exit function

Else

ExecuteAllStepsInFunction= False

Exit Function

End If

End If

Loop

GlobalDictionary("countOfTS") = countOfTS

Set objFnExecute = Nothing

ExecuteAllStepsInFunction= True

End Function

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: Execute the function's steps listed in Functions sheets

'

' Inputs: fnName: Name of the functions which needs to be searched

' fnInTheSheet: Sheet name where function has to be searched

' Returns: True - If all the sub functions executed successfully

' False - If any the sub functions execution failed

'

' Author: Ritesh Kawadkar

' Created At: 6/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function fetchParamValue(strParam, TC\_Name, TS\_Name)

'for concatenated

If InStr(1, strParam, ";;")>0 Then

fetchParamValue = Replace(strParam, ";;", ";")

ElseIf InStr(1, strParam, ";")>0 Then

arrParam = split(strParam, ";")

arrCount = UBound(arrParam)

For i = 0 To arrCount

strParam = arrParam(i)

If Right(strParam, 5) = "<<OUT" Then

strParam = Replace(strParam, "<<OUT", "")

strParam = Trim(strParam)

Set objParam1 = new Connections

objParam1.fnConnectToIOXL GlobalDictionary("TestDataPath"), TS\_Name

strQuery = "Select [Values] from [Output$] where [Variables] = '" & strParam & "' and [TestCases] = '" & TC\_Name & "'"

On Error Resume Next

err.clear

objParam1.fnExecuteQuery(strQuery)

strParam = objParam1.objRecordSet.Fields(0).Value

If err.number=3021 and err.description="Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." Then

WriteLog "ERROR",strParam & " not found in the sheet- " & TS\_Name & " under TC- " & TC\_Name & ". Exiting..."

ExitTest

err.clear

End If

On Error Goto 0

Set objParam1 = Nothing

End If

If Right(strParam, 4) = "<<GD" Then

strParam = Replace(strParam, "<<GD", "")

strParam = Trim(strParam)

strParam = GlobalDictionary(strParam)

End If

If Right(strParam, 4) = "<<IN" Then

strParam = Replace(strParam, "<<IN", "")

strParam = Trim(strParam)

Set objParam2 = new Connections

objParam2.fnConnectToIOXL GlobalDictionary("TestDataPath"), TS\_Name

strQuery = "Select [Values] from [Input$] where [Variables] = '" & strParam & "' and [TestCases] = '" & TC\_Name & "'"

On Error Resume Next

err.clear

objParam2.fnExecuteQuery(strQuery)

strParam = objParam2.objRecordSet.Fields(0).Value

If err.number=3021 and err.description="Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." Then

WriteLog "ERROR",strParam & " not found in the sheet- " & TS\_Name & " under TC- " & TC\_Name & ". Exiting..."

ExitTest

err.clear

End If

On Error Goto 0

Set objParam2 = Nothing

End If

If Right(strParam, 5) = "<<IN$" Then

strParam = Replace(strParam, "<<IN$", "")

strParam = Trim(strParam)

Set objParam3 = new Connections

objParam3.fnConnectToIOXL GlobalDictionary("TestDataPath"), TS\_Name

strQuery = "Select [Values] from [Input$] where [Variables] = '" & strParam & "' and [TestCases] = '" & TC\_Name & "'"

On Error Resume Next

err.clear

objParam3.fnExecuteQuery(strQuery)

strParam = objParam3.objRecordSet.Fields(0).Value & "\_" & fn\_randomString(5)

If err.number=3021 and err.description="Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." Then

WriteLog "ERROR",strParam & " not found in the sheet- " & TS\_Name & " under TC- " & TC\_Name & ". Exiting..."

ExitTest

err.clear

End If

On Error Goto 0

Set objParam3 = Nothing

End If

If strParam = "DATE" Then

strParam = date

End If

If strParam = "TIME" Then

strParam = time

End If

strConcatenatedParam = strConcatenatedParam & strParam & " "

Next

If InStr(1, strParam, ";;")>0 Then

fetchParamValue = Replace(strParam, ";;", ";")

ElseIf InStr(1, strParam, ";")>0 Then

fetchParamValue = readTestDataFile(strParam, TS\_Name)

Else

fetchParamValue = Trim(strConcatenatedParam )

End If

Else

'for non concatenated

If Right(strParam, 5) = "<<OUT" Then

strParam = Replace(strParam, "<<OUT", "")

strParam = Trim(strParam)

Set objParam1 = new Connections

objParam1.fnConnectToIOXL GlobalDictionary("TestDataPath"), TS\_Name

strQuery = "Select [Values] from [Output$] where [Variables] = '" & strParam & "' and [TestCases] = '" & TC\_Name & "'"

On Error Resume Next

err.clear

objParam1.fnExecuteQuery(strQuery)

strParam = objParam1.objRecordSet.Fields(0).Value

If err.number=3021 and err.description="Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." Then

WriteLog "ERROR",strParam & " not found in the sheet- " & TS\_Name & " under TC- " & TC\_Name & ". Exiting..."

ExitTest

err.clear

End If

On Error Goto 0

Set objParam1 = Nothing

End If

If Right(strParam, 4) = "<<GD" Then

strParam = Replace(strParam, "<<GD", "")

strParam = Trim(strParam)

strParam = GlobalDictionary(strParam)

End If

If Right(strParam, 4) = "<<IN" Then

strParam = Replace(strParam, "<<IN", "")

strParam = Trim(strParam)

Set objParam2 = new Connections

objParam2.fnConnectToIOXL GlobalDictionary("TestDataPath"), TS\_Name

strQuery = "Select [Values] from [Input$] where [Variables] = '" & strParam & "' and [TestCases] = '" & TC\_Name & "'"

On Error Resume Next

err.clear

objParam2.fnExecuteQuery(strQuery)

strParam = objParam2.objRecordSet.Fields(0).Value

If err.number=3021 and err.description="Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." Then

WriteLog "ERROR",strParam & " not found in the sheet- " & TS\_Name & " under TC- " & TC\_Name & ". Exiting..."

ExitTest

err.clear

End If

On Error Goto 0

Set objParam2 = Nothing

End If

If Right(strParam, 5) = "<<IN$" Then

strParam = Replace(strParam, "<<IN$", "")

strParam = Trim(strParam)

Set objParam3 = new Connections

objParam3.fnConnectToIOXL GlobalDictionary("TestDataPath"), TS\_Name

strQuery = "Select [Values] from [Input$] where [Variables] = '" & strParam & "' and [TestCases] = '" & TC\_Name & "'"

On Error Resume Next

err.clear

objParam3.fnExecuteQuery(strQuery)

strParam = objParam3.objRecordSet.Fields(0).Value & "\_" & fn\_randomString(5)

If err.number=3021 and err.description="Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." Then

WriteLog "ERROR",strParam & " not found in the sheet- " & TS\_Name & " under TC- " & TC\_Name & ". Exiting..."

ExitTest

err.clear

End If

On Error Goto 0

Set objParam3 = Nothing

End If

If strParam = "DATE" Then

strParam = date

End If

If strParam = "TIME" Then

strParam = time

End If

If InStr(1, strParam, ";;")>0 Then

fetchParamValue = Replace(strParam, ";;", ";")

ElseIf InStr(1, strParam, ";")>0 Then

fetchParamValue = readTestDataFile(strParam, TS\_Name)

Else

fetchParamValue = Trim(strParam)

End If

End If

End Function

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: Execute the function's steps listed in Functions sheets

'

' Inputs: fnName: Name of the functions which needs to be searched

' fnInTheSheet: Sheet name where function has to be searched

' Returns: True - If all the sub functions executed successfully

' False - If any the sub functions execution failed

'

' Author: Ritesh Kawadkar

' Created At: 6/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function readTestDataFile(strParam, TS\_Name)

'for concatenated

arrParam = split(strParam, ";")

arrCount = UBound(arrParam)

For i = 0 To arrCount

strParam = arrParam(i)

If InStr(1, strParam, "<<")>0 Then

variableName = split(strParam, "<<")(0)

TC\_Name = split(strParam, "<<")(1)

strParam = Replace(strParam, "<<" & TC\_Name, "")

strParam = Trim(strParam)

Set objParam1 = new Connections

objParam1.fnConnectToIOXL GlobalDictionary("TestDataPath"), TS\_Name

strQuery = "Select [Values] from [Output$] where [Variables] = '" & strParam & "' and [TestCases] = '" & TC\_Name & "'"

On Error Resume Next

err.clear

objParam1.fnExecuteQuery(strQuery)

strParam = objParam1.objRecordSet.Fields(0).Value

If err.number=3021 and err.description="Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record." Then

WriteLog "ERROR",strParam & " not found in the sheet- " & TS\_Name & " under TC- " & TC\_Name & ". Exiting..."

ExitTest

err.clear

End If

On Error Goto 0

Set objParam1 = Nothing

End If

If strParam = "DATE" Then

strParam = date

End If

If strParam = "TIME" Then

strParam = time

End If

strConcatParam = strConcatParam & strParam & " "

Next

readTestDataFile = Trim(strConcatParam)

End Function

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Purpose: Reads the TestSuite Sheet and based on the Execution Flag

' status stores all the Function in a dictionary and Executes

' Inputs: NONE

'

' Returns: True - If all the sub functions executed successfully

' False - If any the sub functions execution failed

'

' Author: Ritesh Kawadkar

' Cretesd At: 4/4/2018

' Modified By:

' Reason For Change:

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Public Function ExecuteTestSuites()

'Connect to TestSuite Excel File

Set objTestSuite = new Connections

objTestSuite.fnConnectToXL(GlobalDictionary("TestDataPath"))

strQuery = "Select \* from [TestSuite$] where [Execute]=""Y"""

objTestSuite.fnExecuteQuery(strQuery)

' If Not (objTestSuite.objRecordSet.Fields.Count>0) Then

' 'Print "No records found in Excel, TestSuite sheet has no executable tests"

' WriteLog "ERROR","No records found in Excel, TestSuite sheet has no executable tests"

' ExitTest

' End If

Dim newTotalCountOfTCExecuted

newTotalCountOfTCExecuted = 0

newTotalCountOfTCBlocked = 0

iTestSuiteExecuted = 0

iTSCount = 0

GlobalDictionary("TS\_NAME\_" & iTSCount) = "TS\_Name"

bTCPassed = True

GlobalReporting(stepname) = "TS\_1"

'Read all the Key/value pair and store in dictionary

'for all the TS

Do Until objTestSuite.objRecordSet.EOF

For each oField in objTestSuite.objRecordSet.Fields

If NOT GlobalDictionary.Exists(oField.Name) Then

GlobalDictionary.Add oField.Name, oField.Value

Else

GlobalDictionary(oField.Name) = oField.Value

End If

Next

TC\_ID = GlobalDictionary("TC ID")

TC\_D\_ID = GlobalDictionary("Depends On TC")

GlobalDictionary("TC\_STATUS\_0") = bTCPassed

'Check for the Execution Flag

If GlobalDictionary("Execute")="Y" and (TC\_D\_ID=0 or GlobalDictionary("TC\_STATUS\_" & TC\_D\_ID) = True) Then

TS\_Name = GlobalDictionary("TS Name")

TC\_Name = GlobalDictionary("TC Name")

If TS\_Name = GlobalDictionary("TS\_NAME\_" & iTSCount) Then

Else

WriteLog "INFO","\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Executing Test Suite - " & TS\_Name & "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

iTSCount = iTSCount + 1

End If

'Looping to execute all the functions

iCount = 1

iiCount = 1

If Split(GlobalReporting(stepname),"\_")(0) = TS\_Name Then

iTCCount = iTCCount + 1

Else

iTCCount = 1

End If

WriteLog "INFO","Executing Test Case - " & TC\_Name

While (GlobalDictionary.Exists("Functions\_" & iCount)) and (not GlobalDictionary("Functions\_" & iCount)="") and (not IsNull(GlobalDictionary("Functions\_" & iCount))) and (not IsEmpty(GlobalDictionary("Functions\_" & iCount)))

fnName = GlobalDictionary("Functions\_" & iCount) ' Get the fn Name

WriteLog "INFO","Executing FUNCTION = " & fnName

stepname = "TS\_" & iTSCount & "\_TC\_" & iTCCount & "\_FN\_" & iiCount

If Left(fnName, 4) = "UDF\_" and b\_skipFunction = False Then

udfFnName = fnName

Execute("result = " & udfFnName)

If NOT (result = True) Then

udfFnName = Replace(udfFnName, "\_", "")

WriteLog "FAILED", udfFnName

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & udfFnName & "\_F"

ExecuteTestSuites = False

b\_skipFunction = True

Else 'UDF function passed

udfFnName = Replace(udfFnName, "\_", "")

stepDescription = stepname & "\_" & iTCCount & "\_DESC\_"

GlobalReporting(stepDescription)="UDF Function passed " & udfFnName

WriteLog "PASSED",udfFnName

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & udfFnName & "\_P"

End If

ElseIf Left(fnName, 4) = "UDF\_" and b\_skipFunction = True Then 'UDF function blocked

udfFnName = fnName

udfFnName = Replace(udfFnName, "\_", "")

WriteLog "BLOCKED", udfFnName

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & udfFnName & "\_B"

b\_skipFunction = True

Else

fnInTheSheet = getTheSheetNameFromFnStepCountSheet(fnName) ' Get the sheet referrence

If ValidateFunctionPresenceInFnSheet(fnName, fnInTheSheet) AND ValidateStepsCountInFnStepCountSheet(fnName) Then

If stepsCountFromFunctionsSheet = stepsCountFromFnStepCountSheet and b\_skipFunction=False Then

GlobalDictionary(TS\_Name & "\_" & TC\_Name & "\_TCExecutionStartedAt") = now()

If ExecuteAllStepsInFunction(fnName, fnInTheSheet, TS\_Name, stepname, TC\_Name) Then

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & fnName & "\_P"

WriteLog "PASSED",fnName

Else

WriteLog "FAILED",fnName

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & fnName & "\_F"

b\_skipFunction = True

End If

GlobalDictionary(TS\_Name & "\_" & TC\_Name & "\_TCExecutionEndedAt") = now()

Else 'Block execution of this function

b\_skipFunction = True

WriteLog "BLOCKED", fnName

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & fnName & "\_B"

If NOT GlobalDictionary.Exists("TotalTCBlocked") Then

GlobalDictionary.Add "TotalTCBlocked", 1

Else

newTotalCountOfTCBlocked = newTotalCountOfTCBlocked + 1

GlobalDictionary("TotalTCBlocked") = newTotalCountOfTCExecuted

End If

End If

Else

WriteLog "ERROR","Function is not present in Functions Sheet...Exititng"

ExecuteTestSuites = False

ExitTest

End If

End If

iCount= iCount +1

iiCount=iiCount+1

Wend

GlobalDictionary("TS\_NAME\_" & iTSCount) = GlobalDictionary("TS Name")

If b\_skipFunction = False Then

GlobalDictionary("TC\_STATUS\_" & TC\_ID) = True

GlobalReporting("TS\_" & iTSCount & "\_TC\_" & iTCCount) = "P"

Else

GlobalDictionary("TC\_STATUS\_" & TC\_ID) = False

GlobalReporting("TS\_" & iTSCount & "\_TC\_" & iTCCount) = "F"

b\_skipFunction = False

End If

iiCount=0

ElseIf GlobalDictionary("Execute")="Y" and (TC\_D\_ID<>0 or GlobalDictionary("TC\_STATUS\_" & TC\_D\_ID) = False) Then

TS\_Name = GlobalDictionary("TS Name")

TC\_Name = GlobalDictionary("TC Name")

If TS\_Name = GlobalDictionary("TS\_NAME\_" & iTSCount) Then

Else

WriteLog "INFO","\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Executing Test Suite - " & TS\_Name & "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

iTSCount = iTSCount + 1

End If

'Looping to execute all the functions

iCount = 1

iiCount = 1

If Split(GlobalReporting(stepname),"\_")(0) = TS\_Name Then

iTCCount = iTCCount + 1

Else

iTCCount = 1

End If

WriteLog "INFO","Executing Test Case - " & TC\_Name

While (GlobalDictionary.Exists("Functions\_" & iCount)) and (not GlobalDictionary("Functions\_" & iCount)="") and (not IsNull(GlobalDictionary("Functions\_" & iCount))) and (not IsEmpty(GlobalDictionary("Functions\_" & iCount)))

fnName = GlobalDictionary("Functions\_" & iCount) ' Get the fn Name

WriteLog "INFO","Executing FUNCTION = " & fnName

stepname = "TS\_" & iTSCount & "\_TC\_" & iTCCount & "\_FN\_" & iiCount

If Left(fnName, 4) = "UDF\_" Then

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & udfFnName & "\_F"

Else

GlobalDictionary(TS\_Name & "\_" & TC\_Name & "\_TCExecutionStartedAt") = now()

GlobalReporting(stepname) = TS\_Name & "\_" & TC\_Name & "\_" & fnName & "\_B"

GlobalDictionary(TS\_Name & "\_" & TC\_Name & "\_TCExecutionEndedAt") = now()

End If

iCount= iCount +1

iiCount=iiCount+1

Wend

If NOT GlobalDictionary.Exists("TotalTCBlocked") Then

GlobalDictionary.Add "TotalTCBlocked", 1

Else

newTotalCountOfTCBlocked = newTotalCountOfTCBlocked + 1

GlobalDictionary("TotalTCBlocked") = newTotalCountOfTCExecuted

End If

GlobalReporting("TS\_" & iTSCount & "\_TC\_" & iTCCount) = "B"

iiCount=0

End If

objTestSuite.objRecordSet.MoveNext

GlobalDictionary("iTestSuiteExecuted") = GlobalDictionary("iTestSuiteExecuted") +1

Loop

Set objTestSuite = Nothing

Set objTestSuite1 = Nothing

ExecuteTestSuites = True

End Function