|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Anthro_Logo** | **Document Title:**  Anthro Add-in Functional Specification | | | **Document #:** | |
| **Approved by:** | | **Dept:**  Engineering | **Effective date:**  9/8/11 | | **Rev. #:**  A |

Synopsis:

The Anthro Add-in is a dynamically linked library written in VB.NET using the Inventor COM API and the Vault Web Services API. Its purpose is to improve productivity in a number of areas in the development process.

There are currently seven pieces of functionality planned for the Anthro Add-In. They are:

Move Fasteners

Print Drawings

Export DWFx files

Export DXF files

Update iProperties

Create Slot Feature

Draw Line Art

Description:

Move Fasteners

Create a folder in the model pane named fasteners and move all fasteners to the folder.

Print Drawings

Print all drawings selected by the user from a top level assembly.

Export DWFx

Export all drawings selected by the user from a top level assembly to \\Anthro3\dwf\standard.

Export DXF

Export all drawings selected by the user from a top level assembly to \\Svr12\trumpf\_pdm2\TRUMPF.NET\DXF

Update iProperties

Update the Authority: and Revision: iProperties for all components selected by the user and check them into the Vault.

Create Slot Feature

Create a slot feature in sheet metal part given the height and width of the slot at hole points on a sketch and create work axis at the center points.

Draw Line Art

Generate a new drawing document with ISO, top, side, and front views from an assembly document.

Purpose:

Move Fasteners

Provide a one click solution to move all the existing fasteners in an assembly into a folder to reduce clutter with in the model browser.

Print Drawings

Provide a solution to quickly access all drawings with in an assembly and print them.

Export DWFx

Provide a solution for the user to quickly access all drawings with in an assembly and export them for use on the shop floor.

Export DXF

Provide a solution to quickly access all flat patterns with in an assembly and export them for laser and press break programing.

Update iProperties

Provide a solution that will allow the user to open a top level assembly input iProperties for “Approved by”, “Revision”, and a check-in comment. The user will be able to select which drawing files to update.

Create Slot Feature

Provide an automated way to generate slots in a sheet metal part given a set of hole points

Draw Line Art

Provide an automated way to generate the basic views for line art

Functionality:

General:

Development will be done in Micorsoft Visual Studio 2010, Version 10.0.303019.1

Source code is located in [\\ANTHRO3\Design\AnthroInventorAddin\Source](file:///\\ANTHRO3\Design\AnthroInventorAddin\Source)

Source code should be put under revision contron but I am waiting on IT to provide a login.

An installation program will be provided. The installation program will be generated with Inno Setup Version 5.4.3(a).

Installation source code is located in [\\ANTHRO3\Design\AnthroInventorAddin\Source\AnthroAddIn\Setup](file:///\\ANTHRO3\Design\AnthroInventorAddin\Source\AnthroAddIn\Setup)

Final setup executables will be stored in [\\ANTHRO3\Design\AnthroInventorAddin\Setup](file:///\\ANTHRO3\Design\AnthroInventorAddin\Setup)

All code will be wrapped in Try/Catch blocks.

Move Fasteners to Folder

Module: MoveAllFasteners

Imports:

Inventor

System.Windows.Forms

Module Classes:

invApp Inventor.Application

invDoc Inventor.Document

invBrowserModelPane Inventor.BrowserPane

invFastenerFolder Inventor.BrowserFolder

invBrowserTopNode Inventor.BrowserNode

invBrowserNodeToMove Inventor.BrowserNode

Code Design:

1. Assign invDoc to the active document invApp.ActiveDocument.
2. Get the top level of the current “Model” pane and assign it to invBrowerTopNode
3. Check to see if a “Fastener” folder exists. If not, create one.
4. Iterate through the BrowerFolders to find the “Fasteners” folder.
5. Assign the “Fasteners” folder to the BrowserFolder invFastenersFolder.
6. Assign invBrowserNodes to invBrowerTopNode.BrowerNodes.
7. Iterate through the invBrowerNodes and if the first three chanector of the name start with “325”, a fastener, move that node into the “Fasteners” folder.

Print Drawings

Class: PrintDrawingsfromAssemblyDialog(Inventor.Application)

Imports:

Inventor

System.IO

System.Security

System.Windows.Forms

AnthroAddIn.Security

AnthroAddIn.DocumentSvc

System.Collections.Generic

System.Security.Principal.WindowsIdentity

Member Objects:

invApp Inventor.Application

drawingList DrawingDocuments

serverLogin ServerLogin

Member Variables:

bAllChecked Boolean

Functions:

DrawingDialog\_AddListBox(Control.ControlCollection) Boolean

Local Objects:

LablePosition System.Drawing.Point

ListPosition System.Drawing.Point

newList CheckedListBox

newLable Forms.Lable

Subroutens:

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

btnAccept\_Click(Object, System.EnentArgs) Handles btnAccept.Click

Local Objects:

files() DocumentSvc.File

valutService Vaultservices

aControl Control

anObject Object

invDocs Inventor.Documents

invDrawingDoc Inventor.DrawingDocument

Local Variables:

drawingFiles() String

downloadFiles() String

i Integer

j Integer

strDrawingFileNames String

intUserResponse Integer

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

SelectAll\_CheckedChanged(System.Object,System.EventArgs) Handles SelectAll.Checked

New(Inventor.Application)

Code Design:

DrawingDialog\_AddListBox(Control.ControlCollection) Boolean

1. Instantiate LablePosition and ListPosition
2. Set LablePosition.X and LablePosition.Y to 10
3. Set ListPosition.X to 10 and ListPosition.Y to 40
4. Instantiate newList and newLable
5. Set newList.ChackOnClick to True
6. Set newLable.Txt to “DrawingListDialogLable”
7. Set newLable.Text to “Select Drawings to Print”
8. Set newLable.Width to 200
9. Set newLable.Location to LablePosition
10. Set newList.Text to “Select Drawings”
11. Set newList.name to “DrawingListBox”
12. Set newList.Width to 215
13. Set newList.Heigth to 280
14. Set newList.Location to ListPosition
15. Iterate through the currents documents
16. If the document is a sheetmetal part check to see if it is already in newList. If it is not in the list add it.
17. Add newLable to the Control.ControlCollection
18. Sort newList
19. Add newList to the Control.ControlCollection

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

1. Close PrintDrawingsfromAssemblyDialog

btnAccept\_Click(Object, System.EnentArgs) Handles btnAccept.Click

1. Login to the vault server with a call to serverLogin.LoginToVault
2. Instantiate local objects and variables
3. Get the checklistbox control and then set the size of the drawingFiles and downloadFiles array to that size
4. Add the drawing documents that the user has selected to the drawingList array
5. If no drawing documents are selected display a MsgBox() to indicate that at leaset one drawing must be selected
6. Iterate through the list of documents selected by the user and build the drawingList and downloadFiles list. The drawingList has a path approprate for the call to FildLatestFilesByPath Call. The downloadFile has a path approprate for writing the files to disk.
7. Generate the list of drawings to present to the user that will be printed.
8. Display a MsgBox() with the a list of the drawings to be printed to the user with Yes/No buttons.
9. If the user selects Yes intUserResponse will be set to vbYes
10. If intUserResponse is set to vbYes:
11. Make the call to Vault to get the potential files to download with:
12. files = serverLogin.docSvc.FindLatestFilesByPaths(drawingFiles)
13. The list is potential because there is no garantee that there is a drawing file for every document selected.
14. Iterate through the list of files to down load. Error checking for the existence of files is handled in the Vaultservices class
15. Log out of the Vault with the call to serverLogin.LogoutOf Vault()
16. Iterate through all the files downloaded and open them as visable in Inventor This is needed because Inventor won't print a drawing file unless it is visable
17. Iterate through all the open files and print all drawing documents
18. Iterate through all the open files and close the drawing documents.
19. If intUserResponse is set to vbNo:
20. Set strDrawingFilenames to “”
21. Set drawingList.DrawingName.Clear()
22. Set drawingList.DrawingIndex.Clear()
23. Set aControl to Nothing
24. Set anObject to Nothing

SelectAll\_CheckedChanged(System.Object,System.EventArgs)

1. Iterate through all Controls
2. If the control name is “CheckListBox”
3. If bAllChecked is True
4. Iterate through the CheckedListBox and SetItemChecked to False and set bAllChecked to False
5. Else iterate through the CheckedListBox and SetItemChecked to True and set bAllChecked to True

New(Inventor.Application)

1. Set invApp to Inventor.Application

Export DWFx files

Class: ExportDWFxfromAssemblyDialog(Inventor.Application)

Imports:

Inventor

System.IO

System.Security

System.Windows.Forms

AnthroAddIn.Security

AnthroAddIn.DocumentSvc

System.Collections.Generic

System.Security.Principal.WindowsIdentity

Member Objects:

invApp Inventor.Application

drawingList DrawingDocuments

serverLogin ServerLogin

Member Variables:

bAllChecked Boolean

exportPath String

strFileName String

bAcceptClicked Boolean

bCancelClicked Boolean

Functions:

DrawingsDialog\_AddListBox(Control.ControlCollection) Boolean

Local Objects:

LablePosition System.Drawing.Point

ListPosition System.Drawing.Point

newList CheckedListBox

newLable Forms.Lable

Subroutens:

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

btnAccept\_Click(Object, System.EventArgs) Handles btnAccept.Click

Local Objects:

files() DocumentSvc.File

valutService Vaultservices

aControl Control

anObject Object

invDocs Inventor.Documents

invDrawingDoc Inventor.DrawingDocument

Local Variables:

drawingFiles() String

downloadFiles() String

i Integer

j Integer

strDrawingFileNames String

intUserResponse Integer

SelectAll\_CheckedChanged(System.Object,System.EventArgs) Handles SelectAll.Checked

New(Inventor.Application)

Code Design:

DrawingDialog\_AddListBox(Control.ControlCollection) Boolean

1. Instanciate LablePosition and ListPosition
2. Set LablePosition.X and LablePosition.Y to 10
3. Set ListPosition.X to 10 and ListPosition.Y to 40
4. Instantiate newList and newLable
5. Set newList.ChackOnClick to True
6. Set newLable.Txt to “DrawingListDialogLable”
7. Set newLable.Text to “Select Drawings to Print”
8. Set newLable.Width to 200
9. Set newLable.Location to LablePosition
10. Set newList.Text to “Select Drawings”
11. Set newList.name to “DrawingListBox”
12. Set newList.Width to 215
13. Set newList.Heigth to 280
14. Set newList.Location to ListPosition
15. Iterate through the currents documents
16. If the document is a sheetmetal part check to see if it is already in newList. If it is not in the list add it.
17. Add newLable to the Control.ControlCollection
18. Sort newList
19. Add newList to the Control.ControlCollection

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

1. Close ExportDWFxfromAssemblyDialog

btnAccept\_Click(Object, System.EnentArgs) Handles btnAccept.Click

1. Login to the vault server with a call to serverLogin.LoginToVault
2. Instantiate local objects and variables
3. Get the checklistbox control and then set the size of the drawingFiles and downloadFiles array to that size
4. Add the drawing documents that the user has selected to the drawingList array
5. If no drawing documents are selected display a MsgBox() to indicate that at leaset one drawing must be selected
6. Iterate through the list of documents selected by the user and build the drawingList and downloadFiles list. The drawingList has a path approprate for the call to FildLatestFilesByPath Call. The downloadFile has a path approprate for writing the files to disk.
7. Generate the list of drawings to present to the user that will be printed.
8. Display a MsgBox() with the a list of the drawings to be printed to the user with Yes/No buttons.
9. If the user selects Yes intUserResponse will be set to vbYes
10. If intUserResponse is set to vbYes:
11. Make the call to Vault to get the potential files to download with:
12. files = serverLogin.docSvc.FindLatestFilesByPaths(drawingFiles)
13. The list is potential because there is no garantee that there is a drawing file for every document selected.
14. Iterate through the list of files to down load. Error checking for the existence of files is handled in the Vaultservices class
15. Log out of the Vault with the call to serverLogin.LogoutOf Vault()
16. Iterate through all the files downloaded and open them as visable in Inventor This is needed because Inventor won't print a drawing file unless it is visable
17. Iterate through all the open files and and make the call to SaveAs using exportPath, strFileName, and “.dwfx”
18. Iterate through all the open files and close the drawing documents.
19. If intUserResponse is set to vbNo:
20. Set strDrawingFilenames to “”
21. Set drawingList.DrawingName.Clear()
22. Set drawingList.DrawingIndex.Clear()
23. Set aControl to Nothing
24. Set anObject to Nothing

SelectAll\_CheckedChanged(System.Object,System.EventArgs)

1. Iterate through all Controls
2. If the control name is “CheckListBox”
3. If bAllChecked is True
4. Iterate through the CheckedListBox and SetItemChecked to False and set bAllChecked to False
5. Else iterate through the CheckedListBox and SetItemChecked to True and set bAllChecked to True

New(Inventor.Application)

1. Set invApp to Inventor.Application

Export DXF files

Class: ExportDXFfromAssemblyDialog(Inventor.Application)

Imports:

Inventor

System.Windows.Forms

Member Object:

invApp Inventor.Application

DocList DXFDocuments

Member Variables:

strDXFLocation String

Functions:

IsSheetMetal(String) Boolean

Local Variables:

strFirstThreeChr String

strFirstTwoChr String

PartsDialog\_AddListBox(Control.ControlCollection,Documents)

Local Objects:

LablePosition System.Drawing.Point

ListPosition System.Drawing.Point

newList CheckedListBox

newLable Forms.Lable

Subroutens:

ExportToServer(Documents, DXFDocuments)

Local Objects:

InvDoc Inventor.Document

invDataIO DataIO

Local Variables:

strFullPathWithName String

i Integer

sOut String

btnAccept\_Click(Object, System.EventArgs) Handles btnAccept.Click

Local Objects:

aControl Control

anObject Object

Local Variables:

i Integer

j Integer

k Integer

strPartFileName String

intUserResponse Integer

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

New(Inventor.AApplication)

Code Design:

IsSheetMetal(String) Boolean

1. Instantiate strFirstFourChr
2. Set strFirstFourChr to the first four charactors of String
3. Instantiate strFirstTwoChr
4. Set strFirstTwoChr to the first two charactors of String
5. Check to see of strFirstFourChr is “225-“ Or strFirstTwoChr is “M-“
6. If so return True else return Flase

ExportToServer(Documents,DXFDocuments)

1. Instantiate strFullPathWithName
2. Instantiate i
3. Instantiate invDoc
4. Instantiate invDataIO
5. Instantiate sOut
6. Set sOut to DXF export options
7. Iterate through DXFDocuments
8. Set invDoc to current DXFDocuments
9. Set strFullPathWithName to strDXFLocation and current DXFDocuments name.
10. Call invDataIO.WriteDataToFile with sOut and strFullPathWithName

PartsDialog\_AddListBox(Control.ControlCollection) Boolean

1. Instantiate LablePosition and ListPosition
2. Set LablePosition.X and LablePosition.Y to 10
3. Set ListPosition.X to 10 and ListPosition.Y to 40
4. Instanciate newList and newLable
5. Set newList.ChackOnClick to True
6. Set newLable.Txt to “PartsListDialogLable”
7. Set newLable.Text to “Select Parts to Export”
8. Set newLable.Width to 200
9. Set newLable.Location to LablePosition
10. Set newList.Text to “Select Parts to Export”
11. Set newList.name to “PartsListBox”
12. Set newList.Width to 215
13. Set newList.Heigth to 280
14. Set newList.Location to ListPosition
15. Iterate through the currents documents
16. If the document is a PartDocument and a sheetmetal part add it to newList.
17. Add newLable to the Control.ControlCollection
18. Sort newList
19. Add newList to the Control.ControlCollection

btnAccept\_Click(Object, System.EventArgs) Handles btnAccept.Click

1. Instantiate aControl
2. Instantiate anObject
3. Instantiate i
4. Instantiate j
5. Instantiate strPartFileNames and set to “”
6. Instantiate intUserResponse and set to 0
7. Instantiate invDocs and set to invApp.Documents
8. Iterate through aControls
9. If aControl is “CheckedListBox” check that the CheckListBox.Count is not 0
10. If not 0 iterate through anObject
11. If anObject is a string add it to DocList
12. Else display a MsgBox with a message that at lease one part must be selected for export.
13. Iterate through DocList
14. Iterate InvDocs
15. If invDocs name equals DocName name and invDocs a part document set invDocs index to DocList index
16. Iterate through DocList
17. Set strPartFileNames to strPartFileNames and DocList name
18. Set intUserResponse to a msgbox with the list of parts to export with Yes/No buttons.
19. If intUserResponse equals vbYes Call ExportToServer(invDocs, DocList) and Close ExportDXFfromAssemblyDialog(Inventor.Application)
20. Else
21. Set strPartFilenames to “”
22. Clear DocList name
23. Clear DocList index
24. Set aControl to Nothing
25. Set anObject to Nothing

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

1. Clear DocList Name
2. Clear DocList Index
3. Close ExportDXFfromAssemblyDialog(Inventor.Application)

New(Inventor.Application)

1. Set invApp to Inventor.Application

Update iProperties

Class: UpdateiPropertiesDialog(Inventor.Application)

Imports:

Inventor

System.IO

System.Windows.Forms

AnthroAddIn.Security

AnthroAddIn.DocumentSvc

System.Collections.Generic

Member Object:

invApp Inventor.Application

serverLogin ServerLogin

drawingList DrawingDocuments

Member Variables:

bAllChecked Boolean

bLeavingiPropDlg Boolean

Functions:

UpdateiPropertiesDialog\_AddListBox(Control.ControlCollection) Boolean

Local Objects:

LablePosition System.Drawing.Point

ListPosition System.Drawing.Point

newList CheckedListBox

newLable Forms.Lable

Local Variables:

strListName String

Subroutens:

SetiProperty(Inventor.Document)

Local Objects:

oPropSets Inventor.PropertySets

oPropSet Inventor.PropertySet

oPartAuthorityProp Inventor.Property

oPartRevisionProp Inventor.Property

btnAccept\_Click(Object, System.EventArgs) Handles btnAccept.Click

Local Objects:

files() DocumentSvc.File

invSketchSymbol SketchedSymbol

invDrawingSheet Sheet

invDocs Documents

invDrawingDoc DrawingDocument

ecoPosition Point2d

strlstDocNames List(Of String)

strlstDocFile List(Of String)

strlstFolders List(Of String)

strlstLocalPath List(Of String)

vaultService VaultServices

Local Variables:

strDrawingFiles() String

strDownloadFiles() String

strDocFiles() String

strFolders() String

strFolder String

strLocalPath() String

strTmp String

folderLength Integer

intUserResponse Integer

SelectAll\_CheckedChanged(System.Object,System.EventArgs) Handles SelectAll.Checked

txtComment\_Validating(Object, System.ComponentModel.CancelEventArgs) Handles txtComment.Validating

txtBxRevision\_Validating(Object, SystemComponentModel.CancelEventArgs) Handles txtBxRevision.Validating

txtBxApprovedBy\_Validating(Object, SystemComponentModel.CancelEventArgs) Handles txtBxApprovedBy.Validating

btnCancel\_MouseClick(Object, System.EventArgs) Handles btnCancel.MouseHover

New(Inventor.Application)

Code Design:

UpdateiPropertiesDialog\_AddListBox(Control.ControlCollection) Boolean

1. Instanciate LablePosition and ListPosition
2. Set LablePosition.X and LablePosition.Y to 10
3. Set ListPosition.X to 10 and ListPosition.Y to 40
4. Instantiate newList and newLable
5. Set newList.ChackOnClick to True
6. Set newLable.Txt to “DrawingListDialogLable”
7. Set newLable.Text to “Select Drawings to Print”
8. Set newLable.Width to 200
9. Set newLable.Location to LablePosition
10. Set newList.Text to “Select Drawings”
11. Set newList.name to “DrawingListBox”
12. Set newList.Width to 215
13. Set newList.Heigth to 280
14. Set newList.Location to ListPosition
15. Iterate through the currents documents
16. If the document is a sheetmetal part check to see if it is already in newList. If it is not in the list add it.
17. Add newLable to the Control.ControlCollection
18. Sort newList
19. Add newList to the Control.ControlCollection

SetiProperty(Inventor.Document)

1. Instantiate oPropSets
2. Instantiate oPropSet
3. Instantiate oPartAuthorityProp
4. Instantiate oPartRevisionProp
5. Set oPropSets to Inventor.Document.PropertySets
6. Set oPropSet to oPropertySets “Design Tracking Properties”
7. Set oPartAuthorityProp to oPropSet “Authority”
8. Set oPartAuthorityProp Value to txtBxApprovedBy Text
9. Set oPropSet to oPropsets “Inventor summary information”
10. Set oPartRevisionProp to oPropSet “Revision”
11. Set oPartRevisionProp Value to txtBxRevision Text

btnAccept\_Click(Object, System.EventArgs) Handles btnAccept.Click

1. Instantiate invDocs and set to invApp.Documents
2. Instantiate invDrawingDoc
3. Instantiate invDrawingSheet
4. Instantiate invSketchSymbol
5. Instantiate ecoPosition
6. Instantiate files()
7. Instantiate strDrawingFiles()
8. Instantiate strDownloadFiles()
9. INstantiatestrlstdocNames
10. Instantiate strlstDocFiles
11. Instantiate strDocFiles()
12. Instantiate strlstFolders
13. Instantiate strFolders()
14. Instantiate strlstLocalPath
15. Instantiate strLocalPath()
16. Instantiate strTmp
17. Instantiate folderLength
18. Instantiate valutService
19. Instantiate intUserResponse
20. Set Curser to WaiteCursor
21. Call serverLogin LoginToVault
22. Iterate through aControls
23. If aControl is “CheckedListBox” check that the CheckListBox.Count is not 0
24. If not 0 iterate through anObject
25. If anObject is a string add it to drawingList
26. Else display a MsgBox with a message that at lease one document must be selected to updated the iProperties.
27. Set the size of strlstDrawingFiles() to the number of drawings selected by the user.
28. Set the size of strDownloadFiles() to the number of drawings selected by the user.
29. Iterate through the drawingList and prepend the local path to the name and add it to strDownloadFiles()
30. Get the latest version of the drawing files from the Vault server using strDrawingFiles() with the call to serverLogin.docSvc.FildLatestFilesByPaths()
31. Download the latest version of the drawing files and write them to the local disk. Using vaultService.Execute()
32. Open all the drawing files that have been downloaded to the local disk but don’t show them
33. Iterate through the strlstDocNames, convert the paths from local paths to Vault paths, and add them to strltFolders.
34. Iterate through strlstDocNames, append strlstDocNames to strlstFolderFiles, and add them to strlstDocFiles.
35. Get the latest version of the drawing files form the Vault server
36. Check out the drawing files from the Vault server
37. Iterate through strDocFiles and set the iProperties.
38. Iterate through the invDocs. If the document is a drawing document find all the ECO blocks and move them off the sheet.
39. Check in all the drawing documents that have been updated to the Vault server.
40. Log out of the Vault server
41. Set the cursor to default.

btnCancel\_Click(Object,System.EventArgs) Handles btnCancel.Click

1. Call Dispose

SelectAll\_CheckedChanged(System.Object,System.EventArgs) Handles SelectAll.Checked

1. Iterate through all Controls
2. If the control name is “CheckListBox”
3. If bAllChecked is True
4. Iterate through the CheckedListBox and SetItemChecked to False and set bAllChecked to False
5. Else iterate through the CheckedListBox and SetItemChecked to True and set bAllChecked to True

txtComment\_Validating(Object, System.ComponentModel.CancelEventArgs) Handles txtComment.Validating

1. If bLeaveiPropDlg is True set allow the user to leave txtBxComment and exit sub
2. If txtComment text is empty set the error provider for txtComment to “You must input a comment” and prevent the user from leaving txtBxComment
3. Else set the error provider for txtComment to “”

txtBxRevision\_Validating(Object, SystemComponentModel.CancelEventArgs) Handles txtBxRevision.Validating

1. If txtBxRevision is not a number, tash or letter set the error provider for txtBxRevision to “You must input a valid revision”
2. Else set the error provider for txtBxRevision to “”

txtBxApprovedBy\_Validating(Object, SystemComponentModel.CancelEventArgs) Handles txtBxApprovedBy.Validating

1. If txtBxApprovedBy is not all letters set the error provider for txtBxApprovedBy to “You must input a valid approved by initial”
2. Else set the terror provider for txtBxApprovedBy to “”

btnCancel\_MouseClick(Object, System.EventArgs) Handles btnCancel.MouseHover

1. Set bLeavIpropDlg to True

New(Inventor.Application)

1. Set invApp to ThisApp

Create Slot Feature

Class: SlotFeatureDialog(Inventor.Application)

Imports:

Inventor

System.Windows.Forms

Member Objects:

invApp Inventor.Application

oInteraction InteractionEvents

oSelectPoint SelectEvents

invPartDoc Inventor.PartDocument

invAssemDoc Inventor.AssemblyDocument

invPartCompDef Inventor.ComponentDefinition

invDataSets GraphicsDataSets

invCoordSet GraphicsCoordinateSet

oPoint Point

sPoint SketchPoint

sketchPoint Point2d

oSketch PlanarSketch

invSketch Sketch

oNormalVector UnitVector

oReferenceVector UnitVector

oSketchPlane Plane

invCurvesNode GraphicsNode

oCutDefinition CutDefinition

oCutFeature CutFeature

oSheetMetalFeatures SheetMetalFeatures

transientGeometry TransientGeometry

createSlotTransaction Transaction

invProfile Profile

oSlots ObjectCollection

invClientGraphics ClientGraphics

alSelectedPoints ArrayList

oWorkAxis WorkAxis

Member Variables:

bHorizontal Boolean

bVertical Boolean

bFoundVisableSketch Boolean

Global Variables:

iFeatureCount Integer

Subroutens:

btnCancel\_Click(Object, System.EventArgs) Handles btnCancel.Click

btnAccept\_Click(Object, System.EventArgs) Handles btnAccept.Click

Local Object:

oProfile Profile

Local Object:

strSlotName String

DrawSlot(Point2d, Integer)

Local Objects:

oUOM UnitsOfMeasure

lines(2) SketchLine

arcs(2) SketchArc

point SketchPoint

Local Variables:

dFirstPointX Double

dFirstPointY Double

dSecondPointX Double

dSecondPointY Double

dSlotHeight Double

dSlotWidth Double

transientGeometry TransientGeometry

DrawClientGraphics(Point2d, Point, Integer)

Local Objects:

oUOM UnitsOfMeasure

lines(2) SketchLine

arcs(2) SketchArc

oStartPoint2d Point2d

oEndPoint2d Point2d

oCenter2d Point2d

oStartPoint Point

oEndPoint Point

oCenter Point

Local Variables:

Coord2d(1) Double

dSlotHeigth Double

dSlotWidth Double

SlotFeatureDialog\_Disposed(Object,System.EventArgs)

ProcessKeyPreview(System.Wondows.Forms.Message) Boolean

Local Const:

WM\_KEYDOWN Integer

keycode System.Windows.Forms.Key

rbtnHorizontal\_Click(Object, System.EventArgs)

Local Variables:

strHLastError String

strWLastError String

iVisableSketchCount Integer

rbtnVertical\_Click(Object, System.EventArgs)

Local Variables:

strHLastError String

strWLastError String

iVisableSketchCount Integer

SlotFeatureDialog\_Show(Object, System.EventArgs) Handles Me.Show

Local Variables:

iVisableSketchCount Integer

bHoleCentersFound Boolean

New(Inventor.Application)

Finalize()

txtbHeight\_Leave(Object, System.EventArgs) Handles txtbHeigth.Leave

Local Variables:

strLastError String

iVisableSketchCount Integer

txtbWidth\_Leave(Object, System.EventArgs) Handles txtbHeigth.Leave

Local Variables:

strLastError String

iVisableSketchCount Integer

txtbWidth\_TextChanged(Object, System.EventArgs) Handles txtbWidth.TextChanged

Local Variables:

X Double

dSlotWidth Double

txtbWidth\_Validating(Object, System.ComponentModel.CancelEventArgs) Handles txtbWidth.Validating

Local Variables:

X Double

dSlotWidth Double

txtbHeigth\_TextChanged(Object, System.EventArgs) Handles txtbHeigth.TextChanged

Local Variables:

X Double

dSlotHeigth Double

txtbHeigth\_Validating(Object, System.ComponentModel.CancelEventArgs) Handles txtbHeigth.Validating

Local Variables:

X Double

dSlotHeigth Double

btnCancel\_MouseClick(Object, System.Wondows.Forms.MouseEventArgs) Handles btnCancel.MouseClick

btnCancel\_MouseHover(Object, System.EventArgs) Handles btnCancel.MouseHover

btnCancel\_MouseLeave(Object,System.EventArgs) Handles btnCancel.MouseLeave

Code Design:

btnCancel\_Click(Object, System.EventArgs) Handles btnCancel.Click

1. If invClientGraphics set delete invLlientGraphics
2. Call Dispose

btnAccept\_Click(Object, System.EventArgs) Handles btnAccept.Click

1. Instantiate oProfile and set to Nothing
2. Increment iFeatureCount by 1
3. Instantiate strSlotName and set to “Slot” + iFeature.Count
4. Set oSlot to invApp.TransientObjects.CreatObjectCollection
5. Iterate through invSketch.SketchPoints
6. If SketchPoint is a “HoleCenter” Call DrawSlot with SketchPoint
7. If cbWorkAxis checked add work axis with SketchPoint.
8. Set oProfile to oSketch.Profiles.AddForSolids with oSlot
9. Set oSheetMetalFeatures to invPartCompDef.Features
10. Set oCutDefinition to oSheetmMetalFeatures.CutFeatures.CreateCutDefinition with oProfile
11. Call oCutDefinition.SetToNextExtent()
12. Set oCutFeature to oSheetMetalFeatures.CutFeatures.Add with oCutDefinition
13. Set oCutFeature.Name to strSlotName
14. Set oSketch.Shared to False
15. Set oSketch.Visible to False
16. Call oSlots.Clear
17. Call invCurvesNode.Delete
18. Call invClientGraphics.Delete
19. Call invApp.ActiveView.Update
20. Call Dispose

DrawSlot(Point2d, Integer)

1. Instantiate oUOM
2. Instantiate lines(2)
3. Instantiate arcs(2)
4. Instantiate point
5. Instantiate dFirstPointX
6. Instantiate dFirstPointY
7. Instantiate dSecondPointX
8. Instantiate dSecontPointY
9. Instantiate dSlotHeigth and set to txtbHeight
10. Instantiate dSlotWidth and set to txtbWidth
11. Instantiate transientGeometry and set to invApp.TransientGeometry
12. If rbtnHorizontal is checked, Draw the lines and arks that make up the shap of the slot in the horizontal position and apply all constraints to fully constrain the sketch geometry
13. If rbtnVertical is checked, Draw the lines and arks that make up the shape of the slot in the vertical position and apply all constraints to fully constrain the sketch geometry.

DrawClientGraphics(Point2d, Point, Integer)

1. Instantiate oUOM
2. Instantiate lines(2)
3. Instantiate arcs(2)
4. Instantiate coord2d(1)
5. Instantiate oStartPoint2d
6. Instantiate oEndPoint2d
7. Instantiate oCenter2d
8. Instantiate oStartPoint
9. Instantiate oEndPoint
10. Instantiate oCenter
11. Instantiate dSlotHeigth and set to txtbHeight
12. Instantiate dSlotWidth and set to txtbWidth
13. Set invCurvesNode to invClientGraphics.AddNode(integer)
14. If rbtnHorizontal checked, drawing client graphics in a horizontal position.
15. If rbtnVertical, draw client graphics in a vertical position.
16. Call invApp.ActiveView.Update()

SlotFeatureDialog\_Disposed(Object,System.EventArgs)

1. If createSlotTransaction call createSlotTransaction.End()

ProcessKeyPreview(System.Wondows.Forms.Message) Boolean

1. Instantiate WM\_KEYDOWN and set to &H100
2. Instantiate keycode
3. If message is WM\_KEYDOWN and keycode is tab
4. Return TRUE
5. Else Return message

rbtnHorizontal\_Click(Object, System.EventArgs)

1. Instantiate strHLastError
2. Instantiate strWLastError
3. Set strHLastError to ErrorProvider1.GetError(txtbHeight)
4. Set strWLastError to ErrorProvider1.GetError(txtbWidth)
5. If strHLastError is not “” set focus to txtbHeigth and set rbtnHorizontal checked to False
6. If strWLastError is not “” set focus to txtbWidth and set rbtnHorizontal checked to False
7. Delete invClientGraphics
8. Call invApp.ActiveView.Update()
9. Instantiage new invClientGraphics
10. Instantiate iVisabelSketchCount and set to 0
11. Iterate through Sketches
12. If more than one Sketch is visable display a message to the user that only one sketch can be visable and exit the sub
13. Iterate through the Sketchs
14. If a visaable sketch is found set bFoundVisableSketch to True
15. If bFoundVisableSketch is False Display a message “No Visable Sketch Found” and exit the sub
16. Iterate through the SketchPoints in the current sketch
17. If SketchPoint. HoleCenter is true call DrawClientGraphics

rbtnVertical\_Click(Object, System.EventArgs)

1. Instantiate strHLastError
2. Instantiate strWLastError
3. Set strHLastError to ErrorProvider1.GetError(txtbHeight)
4. Set strWLastError to ErrorProvider1.GetError(txtbWidth)
5. If strHLastError is not “” set focus to txtbHeigth and set rbtnHorizontal checked to False
6. If strWLastError is not “” set focus to txtbWidth and set rbtnHorizontal checked to False
7. Delete invClientGraphics
8. Call invApp.ActiveView.Update()
9. Instantiage new invClientGraphics
10. Instantiate iVisabelSketchCount and set to 0
11. Iterate through Sketches
12. If more than one Sketch is visable display a message to the user that only one sketch can be visable and exit the sub
13. Iterate through the Sketchs
14. If a visaable sketch is found set bFoundVisableSketch to True
15. If bFoundVisableSketch is False Display a message “No Visable Sketch Found” and exit the sub
16. Iterate through the SketchPoints in the current sketch
17. If SketchPoint. HoleCenter is true call DrawClientGraphics

SlotFeatureDialog\_Show(Object, System.EventArgs) Handles Me.Show

1. Instantiate iVisableSketchCount and initialize to 0
2. Instantiate bHoleCentersFound and initialize to False
3. Iterate through Sketches
4. If more than one Sketch is visable display a message to the user that only one sketch can be visable and exit the sub
5. Iterate through the Sketches
6. If a visaable sketch is found set bFoundVisableSketch to True
7. If bFoundVisableSketch is False Display a message “No Visable Sketch Found” and exit the sub
8. Iterate through the SketchPoints in the current sketch
9. If SketchPoint. HoleCenter is true call DrawClientGraphics

New(Inventor.Application)

1. Set invApp to Inventor.Application
2. Start new Transaction and assign to createSlotTransaction
3. Set invPartDoc to invApp.ActiveDocument
4. Add new ClientGraphicsCollection and assign to invClientGraphics
5. Set transientGeometry to invApp.TransientGeometry

Finalize()

1. End createSlotTransaction
2. Call Finalize()

txtbHeight\_Leave(Object, System.EventArgs) Handles txtbHeigth.Leave

1. Instantiage strLastError
2. Set strHLastError to ErrorProvider1.GetError(txtbHeight)
3. If strLastError is not “” set focus to txtbHeight and exit sub
4. If bLeavingSlotFeatureDlg is False
5. Delete invClientGraphics
6. Call invApp.ActiveView.Update
7. Add new ClientGraphicsCollection and assign to invClientGraphics
8. Instantiate iVisableSketchCount
9. If a visaable sketch is found set bFoundVisableSketch to True
10. If bFoundVisableSketch is False Display a message “No Visable Sketch Found” and exit the sub
11. Iterate through the SketchPoints in the current sketch
12. If SketchPoint. HoleCenter is true call DrawClientGraphics

txtbWidth\_Leave(Object, System.EventArgs) Handles txtbHeigth.Leave

1. Instantiage strLastError
2. Set strHLastError to ErrorProvider1.GetError(txtbHeight)
3. If strLastError is not “” set focus to txtbWidth and exit sub
4. If bLeavingSlotFeatureDlg is False
5. Delete invClientGraphics
6. Call invApp.ActiveView.Update
7. Add new ClientGraphicsCollection and assign to invClientGraphics
8. Instantiate iVisableSketchCount
9. If a visaable sketch is found set bFoundVisableSketch to True
10. If bFoundVisableSketch is False Display a message “No Visable Sketch Found” and exit the sub
11. Iterate through the SketchPoints in the current sketch
12. If SketchPoint. HoleCenter is true call DrawClientGraphics

txtbWidth\_TextChanged(Object, System.EventArgs) Handles txtbWidth.TextChanged

1. If txtbWidth is null or empty
2. Set txtbWidth color to red
3. Set Error provider of txtbWidth to “Floating point value required”
4. Set invApp.StatusBarText to “Input must be a floating point number”
5. Instantiate X and set to Double.Parse(txtbWidth.Text)
6. Instantiate dSlotWidth and set to Convert.ToDouble(txtbWidth.Text)
7. If dSlotWidth is 0
8. Set txtbWidth color to red
9. Set Error provider of txtbWidth to “Width can not be0!”
10. Set invApp.StatusBarText to “Width can not be 0”
11. Set txtbWidth color to black
12. Clear Error provider
13. If Catch exception
14. Set Error provider of txtbWidth to “Floating point value required”
15. Set invApp.StatusBarText to “Input must be a floating point number
16. Set txtbWidth color to red

txtbWidth\_Validating(Object, System.ComponentModel.CancelEventArgs) Handles txtbWidth.Validating

1. If bLeavingSlotFeatureDlg is True
2. Set System.ComponentModel.CancelEventArgs to Flase
3. Set bLeavingSlotFeatureDlg to False
4. Return
5. If txtbWidth is null or empty
6. Set txtbWidth color to red
7. Set Error provider of txtbWidth to “Floating point value required”
8. Set invApp.StatusBarText to “Input must be a floating point number”
9. Instantiate X and set to Double.Parse(txtbWidth.Text)
10. Instantiate dSlotWidth and set to Convert.ToDouble(txtbWidth.Text)
11. If dSlotWidth is 0
12. Set txtbWidth color to red
13. Set Error provider of txtbWidth to “Width can not be0!”
14. Set invApp.StatusBarText to “Width can not be 0”
15. Set txtbWidth color to black
16. Clear Error provider
17. If Catch exception
18. Set Error provider of txtbWidth to “Floating point value required”
19. Set invApp.StatusBarText to “Input must be a floating point number
20. Set txtbWidth color to red

txtbHeigth\_TextChanged(Object, System.EventArgs) Handles txtbHeigth.TextChanged

1. If txtbHeigth is null or empty
2. Set txtbHeigth color to red
3. Set Error provider of txtbHeigth to “Floating point value required”
4. Set invApp.StatusBarText to “Input must be a floating point number”
5. Instantiate X and set to Double.Parse(txtbHeigth.Text)
6. Instantiate dSlotWidth and set to Convert.ToDouble(txtbHeigth.Text)
7. If dSlotWidth is 0
8. Set txtbWidth color to red
9. Set Error provider of txtbHeigth to “Heigth can not be0!”
10. Set invApp.StatusBarText to “Heigth can not be 0”
11. Set txtb Heigth color to black
12. Clear Error provider
13. If Catch exception
14. Set Error provider of txtbHeigth to “Floating point value required”
15. Set invApp.StatusBarText to “Input must be a floating point number
16. Set txtb Heigth color to red.

txtbHeigth\_Validating(Object, System.ComponentModel.CancelEventArgs) Handles txtbHeigth.Validating

1. If bLeavingSlotFeatureDlg is True
2. Set System.ComponentModel.CancelEventArgs to Flase
3. Set bLeavingSlotFeatureDlg to False
4. Return
5. If txtbHeigth is null or empty
6. Set txtbHeigth color to red
7. Set Error provider of txtbHeigth to “Floating point value required”
8. Set invApp.StatusBarText to “Input must be a floating point number”
9. Instantiate X and set to Double.Parse(txtbHeigth.Text)
10. Instantiate dSlotWidth and set to Convert.ToDouble(txtbHeigth.Text)
11. If dSlotWidth is 0
12. Set txtbWidth color to red
13. Set Error provider of txtbHeigth to “Heigth can not be0!”
14. Set invApp.StatusBarText to “Heigth can not be 0”
15. Set txtb Heigth color to black
16. Clear Error provider
17. If Catch exception
18. Set Error provider of txtbHeigth to “Floating point value required”
19. Set invApp.StatusBarText to “Input must be a floating point number
20. Set txtb Heigth color to red.

btnCancel\_MouseClick(Object, System.Wondows.Forms.MouseEventArgs) Handles btnCancel.MouseClick

1. Set bLeavingSlotFeatureDlg to True

btnCancel\_MouseHover(Object, System.EventArgs) Handles btnCancel.MouseHover

1. Set bLeavingSlotFeatureDlg to True

btnCancel\_MouseLeave(Object,System.EventArgs) Handles btnCancel.MouseLeave

1. SetbLeavingSlotFeatureDlg to False

Draw Line Art

Class: DrawLineArt(Inventor.Application)

Imports:

Inventor

System.Windows.Forms

Member Objects:

invApp Inventor.Application

Subroutens:

DrawViews()

Local Objects:

invDocs Inventor.Documents

invDoc Inventor.Document

invModel Inventor.Document

invDoawingDoc Inventor.DrawingDocument

invSheet Inventor.Sheet

invPoint Inventor.Point2d

invMovePoint Inventor.Point2d

invBaseView Inventor.DrawingView

invSideView Inventor.DrawingView

invTopView Inventor.DrawingView

invISOView Inventor.DrawingView

Local Variables:

strFullDocumentName String

New(Inventor.Application)

Code Design:

DrawViews()

1. Instantiate invDocs
2. Set invDocs to invApp.Documents
3. Instantiate invDoc
4. Set invDoc to invApp.ActiveDocument
5. Instantiate strFullDocumentName
6. Set strFullDocumentName to invDoc.FullDocumentName
7. Instantiate invModel
8. Set invModel to invApp.Documents.Open(strFullDocumentName)
9. Instantiage invDrawingDocument
10. Set invDrawingDocument to invDocs.Add(DocumentTypeEnum.kDrawingDocumentObject)
11. Instantiate invSheet
12. Set invSheet to invDrawingDoc.ActiveSheet
13. Set invSheet.Size toDrawingSheetSizeEnum. kBrawwingSheetSize
14. Instantiate invPoint
15. Set invPoint.X to 6, and invPoint.Y to 6
16. Instantiate invMovePoint
17. Set invMovePoint.X to 36 and invPoint.Y to 6
18. Instantiage invBaseView
19. Instantiage invSideView
20. Instantiage invTopView
21. Instantiage invISOView
22. Set invBaseView with AddBaseView
23. Set invBaseView.DisplayTangentEdges to True
24. Set invSideView with AddProjectedView
25. Move invBaseView to invMovePosition
26. Set invTopView with AddProjectedView
27. SetTopView.Aligned to False
28. Set invTopView.DisplayDefinitionInBase to False
29. Set invTopView.InheritBreak to False
30. Set invTopView.ShowLabel to Flase
31. Align invTopView with invSideView
32. Set invISOView with AddProjectedView
33. Move invISOView to X 6 Y 6
34. Move invSideView to X 26 Y 6
35. Move invTopView to X 16 Y 6
36. Set invISOView.DisplayTangentEdges to True