@using System.Text.RegularExpressions @{ int XPCid = 0; string XPCParent = TemplateHelper.FieldValue(Model, "ItemXPCSysid").ToString(); if (!String.IsNullOrEmpty(XPCParent)) { try { XPCid = Convert.ToInt32(XPCParent); //Fetch the correct XPC sys id } catch(Exception ex) { } } else { foreach (var parentEntityId in TemplateHelper.GetParentEntityModels(Model, "Product")) { XPCid = parentEntityId.Id; break; } } int @TranslationEntityId = 26012; //Shall only be one in the system } @helper GetAvailableMeasurments(int XPCid) { string[] availableMeasures = {"ItemMeasurementA","ItemMeasurementB","ItemMeasurementC","ItemMeasurementD","ItemMeasurementF","ItemMeasurementG","ItemMeasurementH","ItemMeasurementI","ItemMeasurementJ","ItemMeasurementK","ItemMeasurementL","ItemMeasurementM","ItemMeasurementN","ItemMeasurementO","ItemMeasurementP","ItemMeasurementQ","ItemMeasurementR","ItemMeasurementS","ItemMeasurementT","ItemMeasurementU","ItemMeasurementV"}; int len = availableMeasures.Length; string preliminaryTolerance = TemplateHelper.FieldValue(XPCid, "ProductPreliminaryTolerances").ToString(); foreach (var measure in availableMeasures) { string measureValue = TemplateHelper.FieldValue(Model, measure).ToString(); if (measureValue!="") { string firstValue = measureValue.Split('/')[0]; double inchesValueFirst = Math.Round(Convert.ToDouble(firstValue) * 100.0 / 25.4) / 100; string roundUp = ""; string roundDown = ""; double roundUpInch = 0; double roundDownInch = 0; try { roundUp = measureValue.Split('/')[1]; roundDown = measureValue.Split('/')[2]; roundUpInch = Math.Round(Convert.ToDouble(roundUp) * 100.0 / 25.4) / 100; roundDownInch = Math.Round(Convert.ToDouble(roundDown) * 100.0 / 25.4) / 100; } catch { } string measureLabel = TemplateHelper.FieldTypeName(measure).ToString(); if(measure == "ItemMeasurementF") { measureLabel = "F"; try { firstValue = Math.Round(Convert.ToDouble(firstValue),1).ToString(); roundUp = "+" + Math.Round(Convert.ToDouble(roundUp),1).ToString(); roundDown = Math.Round(Convert.ToDouble(roundDown),1).ToString(); } catch (Exception e) { firstValue = "0"; roundUp = "0"; roundDown = "0"; } } if (preliminaryTolerance!="True") { if (@roundUp!="") { @measureLabel @firstValue.Replace(".00", "")@inchesValueFirst; } else { @measureLabel @firstValue.Replace(".00", "")@roundUp.Replace(".00", "")/@roundDown.Replace(".00", "")@inchesValueFirst (+@roundUpInch/@roundDownInch); } } else { @measureLabel @firstValue.Replace(".00", "")@inchesValueFirst; } } return; } @helper GetAvailableConnections() { string[] availableConnections = {"ItemConnectionsF1","ItemConnectionsF2","ItemConnectionsF3","ItemConnectionsF4","ItemConnectionsF5","ItemConnectionsF6","ItemConnectionsP1","ItemConnectionsP2","ItemConnectionsP3","ItemConnectionsP4","ItemConnectionsP5","ItemConnectionsP6"}; List usedConnections = new List(); List usedConnectionsPositions = new List(); List usedConnectionsStrPositions = new List(); foreach (var connection in availableConnections) { string connectionValue = TemplateHelper.FieldValue(Model, connection).ToString(); if (connectionValue != "" && connectionValue != "0") { string connectionLabel = TemplateHelper.FieldTypeName(connection).ToString(); if(!usedConnections.Contains(connectionValue)) { usedConnections.Add(connectionValue); usedConnectionsPositions.Add(connectionLabel); } else { for(int i = 0;i < usedConnections.Count; i++) { if(usedConnections[i] == connectionValue) { usedConnectionsPositions[i] = usedConnectionsPositions[i] + ", " + connectionLabel; } } } } for(int i = 0;i < usedConnections.Count; i++) { @usedConnectionsPositions[i]@usedConnections[i] @GetAccessoryName(@usedConnections[i]) } return; } @helper GetAccessoryName(String accessoryArtno) { foreach(var subModel in TemplateHelper.GetSubEntityModels(Model, "Connection")) { if(TemplateHelper.FieldValue(subModel, "ConnectionArtNo").ToString() == accessoryArtno) { @TemplateHelper.FieldValue(subModel, "ConnectionDenomination"); @TemplateHelper.FieldValue(subModel, "ConnectionHeight"); @TemplateHelper.FieldValue(subModel, "ConnectionMaterial"); } } return; } @helper GetAvailableAccessories() { foreach (var subModel in TemplateHelper.GetSubEntityModels(Model, "Accessory")) { @TemplateHelper.FieldValue(subModel, "AccessoryArtNo")@TemplateHelper.FieldValue(subModel, "AccessoryDenomination") } return; } @helper GetCorrectDenomination() { if(!TemplateHelper.FieldValue(Model, "ItemConnectionDenomination").ToString().Equals("")){@TemplateHelper.FieldValue(Model, "ItemConnectionDenomination"); } else { @TemplateHelper.FieldValue(Model, "ItemBPHEDenomination") @TemplateHelper.FieldValue(Model, "ItemDenomination"); } } @helper getCurrentDate() { @DateTime.Now.ToString("yyyy-MM-dd") return; } @helper fetchCorrectImage(string type, int XPCid) { if (XPCid > 0) { foreach (var subEntityId in TemplateHelper.GetSubEntityModels(XPCid, "Resource")) { if(TemplateHelper.FieldValue(subEntityId, "ResourceType").ToString() == type) {

return; } } } } @helper fetchCompleteWeight(int XPCid) { if (XPCid > 0) { string ProductWeightStart = TemplateHelper.FieldValue(XPCid, "ProductWeightStart").ToString(); string ProductWeightPlate = TemplateHelper.FieldValue(XPCid, "ProductWeightPlate").ToString(); string ProductWeightLessChpl = TemplateHelper.FieldValue(XPCid, "ProductWeightLessChpl").ToString(); string NoPItemNoP = TemplateHelper.FieldValue(Model, "ItemNoP").ToString(); try { double WeightStart = Convert.ToDouble(ProductWeightStart); double WeightPlate = Convert.ToDouble(ProductWeightPlate); double WeightLessChpl = Convert.ToDouble(ProductWeightLessChpl); double NumberOfPlates = Convert.ToDouble(NoPItemNoP); double totalWeight = WeightStart + WeightPlate * (NumberOfPlates - WeightLessChpl); double totalWeightlb = totalWeight * 2.2046; totalWeight = Math.Round(totalWeight,1); totalWeightlb = Math.Round(totalWeightlb, 1); @totalWeight kg (@totalWeightlb lbs) (Formula: @WeightStart + @WeightPlate * (@NumberOfPlates - @WeightLessChpl)) } catch(Exception ex) { } } } @helper fetchCompleteWeightNew() { try { string strCompleteWeight = TemplateHelper.FieldValue(Model, "ItemWeight").ToString(); double completeWeight = Convert.ToDouble(strCompleteWeight.Replace(",", ".")); completeWeight = Math.Round(completeWeight,1); @completeWeight kg } catch (Exception e) { } } @helper fetchMaterialFromXPC(int XPCid) { if (XPCid > 0) { string materialFromXPC = TemplateHelper.FieldValue(XPCid, "ProductMaterial").ToString(); materialFromXPC = materialFromXPC.Split('-')[1].Trim(); @materialFromXPC } } @helper getPreliminaryToleranceText(int XPCid) { if (XPCid > 0) { if (TemplateHelper.FieldValue(XPCid, "ProductPreliminaryTolerances").ToString()=="True") { int TranslationEntityId = 26012;

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationToleranceNotSet")

} } } @helper getHoldUpVolumes() { string holdUpVolume1 = TemplateHelper.FieldValue(Model, "ItemHoldUpVolume1").ToString().Replace(",", "."); string holdUpVolume2 = TemplateHelper.FieldValue(Model, "ItemHoldUpVolume2").ToString().Replace(",", "."); string holdUpVolume3 = TemplateHelper.FieldValue(Model, "ItemHoldUpVolume3").ToString().Replace(",", "."); if (holdUpVolume1!="") { double holdUpVolume1Dbl; Double.TryParse(holdUpVolume1,out holdUpVolume1Dbl); double holdUpVolume1FT3 = Math.Round(holdUpVolume1Dbl/28.3168466,3); holdUpVolume1 = Math.Round(holdUpVolume1Dbl,3).ToString(); string strHoldUpVolume1FT3 = Math.Round(holdUpVolume1FT3,3).ToString(); @TemplateHelper.FieldTypeName("ItemHoldUpVolume1") @holdUpVolume1 dm³ (@strHoldUpVolume1FT3 ft³) } if (holdUpVolume2!="") { double holdUpVolume2Dbl; Double.TryParse(holdUpVolume2,out holdUpVolume2Dbl); double holdUpVolume2FT3 = Math.Round(holdUpVolume2Dbl/28.3168466,3); holdUpVolume2 = Math.Round(holdUpVolume2Dbl,3).ToString(); string strHoldUpVolume2FT3 = Math.Round(holdUpVolume2FT3,3).ToString(); @TemplateHelper.FieldTypeName("ItemHoldUpVolume2") @holdUpVolume2 dm³ (@strHoldUpVolume2FT3 ft³) } if (holdUpVolume3!="") { double holdUpVolume3Dbl; Double.TryParse(holdUpVolume3,out holdUpVolume3Dbl); double holdUpVolume3FT3 = Math.Round(holdUpVolume3Dbl/28.3168466,3); holdUpVolume3 = Math.Round(holdUpVolume3Dbl,3).ToString(); string strHoldUpVolume3FT3 = Math.Round(holdUpVolume3FT3,3).ToString(); @TemplateHelper.FieldTypeName("ItemHoldUpVolume3") @holdUpVolume3 dm³ (@strHoldUpVolume3FT3 ft³) } return; }

@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONDOCUMENTTYPEPRODUCTSPECIFICATION")

@TemplateHelper.FieldValue(XPCid, "ProductCommercialName"), @TemplateHelper.FieldValue(Model, "ItemNumber")

@TemplateHelper.FieldTypeName("ItemNumber"): @TemplateHelper.FieldValue(Model, "ItemNumber")

@TemplateHelper.FieldTypeName("ItemConnectionDenomination"): @GetCorrectDenomination()

@TemplateHelper.FieldTypeName("ItemNoP"): @TemplateHelper.FieldValue(Model, "ItemNoP")

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationsStandardDimensionsHeader")

@fetchCorrectImage("Sketch", XPCid) @GetAvailableMeasurments(XPCid)

#	MM	IN

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationMeasurmentFootnote")

@getPreliminaryToleranceText(XPCid)

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationsConnectionsHeader")

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationsConnectionsExplanation")

@GetAvailableConnections()

@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONPOSITION")	@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONARTNO")	@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONDENOMINATION")	@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONHEIGHT")	@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONMATERIAL")

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationAccessoryHeader")

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationsAccessoryExplanation")

@GetAvailableAccessories()

@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONARTNO")	@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONDENOMINATION")

@fetchCorrectImage("Graph PED", XPCid)

@fetchCorrectImage("Graph UL", XPCid)

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationStandardDataHeader")

@getHoldUpVolumes()

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationPEDCategory")	@TemplateHelper.FieldValue(Model, "ItemPedCategory")
@TemplateHelper.FieldValue(TranslationEntityId, "TranslationTotalWeightIncConnections")	@fetchCompleteWeightNew()
@TemplateHelper.FieldValue(TranslationEntityId, "TranslationTotalHeatTransferArea")	

@TEMPLATEHELPER.FIELDVALUE(TRANSLATIONENTITYID, "TRANSLATIONMATERIALHEADER")
@fetchMaterialFromXPC(XPCid)

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationDisclaimerHeader")

@TemplateHelper.FieldValue(TranslationEntityId, "TranslationDisclaimer")

www.swep.net | Product specification: SWEP @GetCorrectDenomination()
Configuration No. @TemplateHelper.FieldValue(Model, "ItemNumber") | This PDF was created @getCurrentDate()