```
1  using Autodesk.Revit.DB;
2  using Autodesk.Revit.UI;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Windows.Forms;
7  using Autodesk.Revit.UI.Selection;
8  using System.Windows.Media.Imaging;
9  using System.IO;
10 using System.Drawing.Imaging;
11 using System.Drawing;
12
13
14 namespace PipeTagger.Common
15 {
16     [Autodesk.Revit.Attributes.Transaction
         (Autodesk.Revit.Attributes.TransactionMode.Manual)]
17     public class Picker
18     {
19
20
21         //public class All_Pipe_Selection_Filter : ISelectionFilter
22         //{
23         //    Document doc = null;
24         //    public All_Pipe_Selection_Filter(Document document)
25         //    {
26         //        doc = document;
27         //    }
28         //    public bool AllowElement(Element elem)
29         //    {
30         //        if (elem.Category.Name == "管" || elem.Category.Name == "風
             管" || elem.Category.Name == "電管" || elem.Category.Name == "電纜
             架")
31         //        {
32         //            return true;
33         //        }
34         //        return true;
35         //    }
36
37         //    public bool AllowReference(Reference refe, XYZ position)
38         //    {
39         //        Element elem = doc.GetElement(refe);
40
41         //        if (elem.GetType().Name == "RevitLinkInstance")
42         //        {
43         //            RevitLinkInstance revitlinkinstance = doc.GetElement
             (refe) as RevitLinkInstance;
44         //            Document docLink = revitlinkinstance.GetLinkDocument();
45         //            Element element = docLink.GetElement
             (refe.LinkedElementId);
46         //            if (element.Category.Name == "管" ||
             element.Category.Name == "風管" || element.Category.Name == "電管"
             || element.Category.Name == "電纜架")
```

```
47          //            {
48          //                return true;
49          //            }
50          //        }
51
52          //        return false;
53          //    }
54          //}
55
56          public class PipeSelectionFilter : ISelectionFilter
57          {
58              public bool AllowElement(Element element)
59              {
60                  if (element.Category.Name == "管" || element.Category.Name ==
                      "風管" || element.Category.Name == "電管" ||
                      element.Category.Name == "電纜架")
61                  {
62                      return true;
63                  }
64                  return false;
65              }
66              public bool AllowReference(Reference refer, XYZ point)
67              {
68                  return false;
69              }
70          }
71
72          public class LinkPipeSelectionFilter : ISelectionFilter
73          {
74              Document doc = null;
75              public LinkPipeSelectionFilter(Document document)
76              {
77                  doc = document;
78              }
79              public bool AllowElement(Element element)
80              {
81                  return true;
82              }
83              public bool AllowReference(Reference refer, XYZ point)
84              {
85
86                  RevitLinkInstance revitlinkinstance = doc.GetElement(refer) as
                       RevitLinkInstance;
87                  Document docLink = revitlinkinstance.GetLinkDocument();
88                  Element element = docLink.GetElement(refer.LinkedElementId);
89                  if (element.Category.Name == "管" || element.Category.Name ==
                      "風管" || element.Category.Name == "電管" ||
                      element.Category.Name == "電纜架")
90                  {
91                      return true;
92                  }
93                  return false;
94              }
```

```
95          }
96
97
98          public static (bool isUp, bool isHorizontal, bool isLeft,
              IList<Reference> sorted_list) is_UD_HV_LR(UIDocument uidoc,
              List<Reference> pipes, XYZ pickPoint1, XYZ pickPoint2)
99          {
100             bool isUp = true;
101             bool isHorizontal = true;
102             bool isLeft = true;
103
104             Document doc = uidoc.Document;
105
106             #region 2.1.1 Get x, y location of pipes' project point by
                  Pickpoint 1
107
108             List<Double> xi = new List<Double>();
109             List<Double> yi = new List<Double>();
110
111             foreach (Reference refe in pipes)
112             {
113                 Element elem = doc.GetElement(refe);
114
115                 //1. Get pipe element
116                 // Case 1.1 Host element
117                 if (elem.GetType().Name != "RevitLinkInstance")
118                 {
119                     elem = doc.GetElement(refe);
120
121                     LocationCurve locCurve = elem.Location as LocationCurve;
122                     Line line = locCurve.Curve as Line;
123
124                     xi.Add(line.Project(pickPoint1).XYZPoint.X);
125                     yi.Add(line.Project(pickPoint1).XYZPoint.Y);
126
127                     //MessageBox.Show((line.Project
                        (pickPoint1).XYZPoint.X).ToString(), (line.Project
                        (pickPoint1).XYZPoint.Y).ToString());
128                 }
129
130
131                 // Case 1.2 Linked instance
132                 // 此時選擇的外參reference仍是外參檔的reference，須轉型成外參
                      檔裡的管
133                 else
134                 {
135                     RevitLinkInstance linkInstance = elem as
                        RevitLinkInstance; // 將選取的物件轉型為外參檔
136
137                     //XYZ trans = linkInstance.GetTotalTransform().Origin;
138                     Transform trans = linkInstance.GetTotalTransform();
139
140                     Document docLink = linkInstance.GetLinkDocument(); // 指名
```

```
                    新的document存放該外參檔資料
141                 elem = docLink.GetElement(refe.LinkedElementId); // 從外參
                    的document裡尋找欲選擇的管，轉型成element
142
143                 LocationCurve locCurve = elem.Location as LocationCurve;
144
145                 Curve line = locCurve.Curve as Curve;
146
147                 Curve trans_line = line.CreateTransformed(trans);
148
149                 xi.Add(trans_line.Project(pickPoint1).XYZPoint.X);
150                 yi.Add(trans_line.Project(pickPoint1).XYZPoint.Y);
151
152                 //MessageBox.Show((trans_line.Project
                    (pickPoint1).XYZPoint.X).ToString(),
153                 //                (trans_line.Project
                    (pickPoint1).XYZPoint.Y).ToString());
154
155             }
156         }
157
158         //if (xi.Count != yi.Count) { MessageBox.Show("Picked points'x, y
              coord are not the same, something may happen"); }
159
160         #endregion
161
162         #region 2.1.1.A. Slope of pipes' midpoints-> tag text:
              isHorizontal ?
163
164         // Case 1. no difference in x_i => pipes stack vertically =>
              Horizontal text
165         if (xi.Max() - xi.Min() < 0.001) { isHorizontal = true; }
166
167         // Case 2. no difference in y_i => pipes stack horizontally =>
              Vertical text
168         else if (yi.Max() - yi.Min() < 0.001) { isHorizontal = false; }
169
170         // Case 3. slope of midpoints of first two pipes (slope of every
              two pipes should be the same) -> Absolute value
171         else
172         {
173             List<double> slopes = new List<double>();
174             foreach (int coor1 in Enumerable.Range(0, xi.Count))
175             {
176                 foreach (int coor2 in Enumerable.Range(0, xi.Count))
177                 {
178                     if (coor2 != coor1)
179                     {
180                         slopes.Add(Math.Abs((yi[coor1] - yi[coor2]) / (xi
                    [coor1] - xi[coor2])));
181                     }
182                 }
183             }
```

```
184
185                 if ((slopes.Max() - slopes.Min()) > 0.001)
186                 {
187                     MessageBox.Show("生成的標籤可能無法對齊顯示,請注意第一個點擊點.");
188                     //MessageBox.Show("The first picked point cannot project to all pipes, the tages may not be aligned!!!");
189                     //MessageBox.Show($"Max slope: {slopes.Max()}, Min slope: {slopes.Min()}");
190                 }
191
192
193                 double slope = slopes.Average();
194                 if (slope >= 1) { isHorizontal = true; }
195                 else { isHorizontal = false; }
196             }
197             #endregion
198
199             #region 2.1.1.B. pickPoint1 -> isUp ?
200
201             if (isHorizontal) { isUp = (pickPoint1.Y > yi.Average()) ? true : false; }  //Case Horizontal text:  1. Pick.Y > Y_avg => UP  2. Pick.Y < Y_avg => Down
202             else { isUp = (pickPoint1.X > xi.Average()) ? true : false; }               //Case vertical text:    1. Pick.X > X_avg => UP  2. Pick.X < X_avg => Down
203
204             #endregion
205
206             #region 2.1.1.C pickPoint2 -> isLeft?
207
208             if (isHorizontal) { isLeft = (pickPoint1.X > pickPoint2.X) ? true : false; } //Case Horizontal text:  1. Pick1.X > Pick2.X => Left  2. Pick1.X < Pick2.X => Right
209             else { isLeft = (pickPoint1.Y > pickPoint2.Y) ? true : false; }             //Case vertical text:    1. Pick1.Y > Pick2.Y => Left  2. Pick1.Y > Pick2.Y => Right
210             #endregion
211
212             #region 2.1.2 Sorted Tag's order
213
214             // 4 Cases in total:
215             // using y coor. : U-H / D-H
216             // using x coor. : U-V / D-V
217
218             pipes.Sort((ref1, ref2) =>
219             {
220
221
222
223                 Element elem1 = Common_Tool.host_link_element_All_in_one(ref1, doc, "Element");
224                 Element elem2 = Common_Tool.host_link_element_All_in_one(ref2,
```

```
                    doc, "Element");
225
226                 #region Replace by common tool.Get_host_or_link_element
227
228
229                 //Element elem1 = doc.GetElement(ref1);
230                 //Element elem2 = doc.GetElement(ref2);
231
232                 //if (elem1.GetType().Name != "RevitLinkInstance"){
233                 //    elem1 = doc.GetElement(ref1);}
234                 //else{
235                 //    RevitLinkInstance linkInstance = elem1 as RevitLinkInstance; // 將選取的物件轉型為外參檔
236                 //    Document docLink = linkInstance.GetLinkDocument(); // 指名新的document存放該外參檔資料
237                 //    elem1 = docLink.GetElement(ref1.LinkedElementId); // 從外參的document裡尋找欲選擇的管，轉型成element
238                 //}
239
240                 //if (elem2.GetType().Name != "RevitLinkInstance")
241                 //{
242                 //    elem2 = doc.GetElement(ref2);
243                 //}
244                 //else
245                 //{
246                 //    RevitLinkInstance linkInstance = elem2 as RevitLinkInstance; // 將選取的物件轉型為外參檔
247                 //    Document docLink = linkInstance.GetLinkDocument(); // 指名新的document存放該外參檔資料
248                 //    elem2 = docLink.GetElement(ref2.LinkedElementId); // 從外參的document裡尋找欲選擇的管，轉型成element
249                 //}
250                 #endregion
251
252                 Line line1 = (elem1.Location as LocationCurve).Curve as Line;
253                 Line line2 = (elem2.Location as LocationCurve).Curve as Line;
254
255                 //XYZ Porj_pt1 = line1.Project(pickPoint1).XYZPoint;
256                 //XYZ Porj_pt2 = line2.Project(pickPoint1).XYZPoint;
257
258                 XYZ Porj_pt1 = Common_Tool.host_link_element_All_in_one(ref1, doc, "Projection", pickPoint1);
259                 XYZ Porj_pt2 = Common_Tool.host_link_element_All_in_one(ref2, doc, "Projection", pickPoint1);
260
261                 int comparer = 0;
262
263                 if (isUp & isHorizontal) { comparer = (Porj_pt1.Y).CompareTo(Porj_pt2.Y); }
264                 if (!isUp & isHorizontal) { comparer = (Porj_pt2.Y).CompareTo(Porj_pt1.Y); }
265
266                 if (isUp & !isHorizontal) { comparer = (Porj_pt1.X).CompareTo
```

```
                    (Porj_pt2.X); }
267                 if (!isUp & !isHorizontal) { comparer = (Porj_pt2.X).CompareTo
                    (Porj_pt1.X); }
268
269
270                 //MessageBox.Show(comparer.ToString()); => -1, 1 , 0
271
272                 return comparer;
273
274
275             });
276
277
278
279             #endregion
280
281             return (isUp, isHorizontal, isLeft, pipes);
282         }
283
284
285     }
286     public class Placer
287     {
288         public static Result CreateSomething(UIDocument uidoc, Document doc,
              ref string message, string place_type)
289         {
290             try
291             {
292
293                 List<Reference> pipes = new List<Reference>();
294
295                 using (Transaction t = new Transaction(doc, "Select & Create
                    preview line"))
296                 {
297                     t.Start();
298
299                     #region 1. Pick pipe and put into a list "pipes"
300
301                  //Filter for internal and linked "Pipe"
302                     ISelectionFilter selFilter = new
                    Picker.PipeSelectionFilter();
303                     ISelectionFilter selFilter_link = new
                    Picker.LinkPipeSelectionFilter(doc);
304
305                     #region [To do] Pick host and link at the same time
306
307                     //ISelectionFilter selFilter_All = new
                    Picker.All_Pipe_Selection_Filter(doc);
308                     //IList<Reference> refElemLinked_all =
                    uidoc.Selection.PickObjects(ObjectType.PointOnElement,
                    selFilter_All, "3.請點選要連續標註的管，結束選取按完成");
309                     //IList<Reference> refElemLinked_all =
                    uidoc.Selection.PickObjects(ObjectType.Element, selFilter,
```

```
                    "1.請點選要連續標註的管(專案內)，結束選取按完成",
                    uidoc.Selection.PickObjects(ObjectType.LinkedElement,
                    selFilter_link, "2.請點選要連續標註的管(外參)，結束選取按完
                    成"));
310
311                 //MessageBox.Show(refElemLinked_all.Count.ToString());
312                 //IList<Element> refElem_PickBox =
                    uidoc.Selection.PickElementsByRectangle();
313
314                 //string outtt = "";
315                 //foreach (Element elem in refElem_PickBox)
316                 //{
317                 //    outtt += elem.Name + "\n";
318                 //}
319
320                 //MessageBox.Show(outtt);
321                 //MessageBox.Show(refElem_PickBox.Count.ToString());
322
323                 // 依序轉型成Element，存入pipes
324                 //foreach (Element element in refElem_PickBox) //Concat  =
                     refElem + refElemLinked
325                 //{
326
327                 //    pipes.Add(refe);
328                 //}
329                 #endregion
330
331              //Allow user to pick pipe only
332                 IList<Reference> refElem = uidoc.Selection.PickObjects
                    (ObjectType.Element, selFilter, "1.請點選要連續標註的管(專
                    案內)，結束選取按完成");
333
334
335                 IList<Reference> refElemLinked = null;
336
337                 if (place_type != "Spot")
338                 {
339                     refElemLinked = uidoc.Selection.PickObjects
                    (ObjectType.LinkedElement, selFilter_link, "2.請點選要連續
                    標註的管(外參)，結束選取按完成");
340                 }
341
342             //Pick point to place tag
343                 XYZ pickPoint1 = uidoc.Selection.PickPoint("3.請點選標籤起
                    始位置");
344                 XYZ temp_proj_on_pipe =
                    Common_Tool.host_link_element_All_in_one(refElem[0], doc,
                    "Projection", pickPoint1);
345
346             // Create preview line
347
348                 XYZ P1 = new XYZ(pickPoint1.X, pickPoint1.Y, 0);
349                 XYZ P2 = new XYZ(temp_proj_on_pipe.X, temp_proj_on_pipe.Y,
```

```
                    0);
350                 Line L1 = Line.CreateBound(P1, P2);
351                 var temp_line = doc.Create.NewDetailCurve(doc.ActiveView,
                    L1);
352
353
354
355                 XYZ pickPoint2 = uidoc.Selection.PickPoint("4.請選擇標籤向
                    左長或向右長(往左或右任意點選)");
356                 doc.Delete(temp_line.Id);
357
358             // 依序轉型成Element，存入pipes
359
360                 foreach (Reference refe in refElem)
361                 {
362                     pipes.Add(refe);
363                 }
364
365                 if (place_type != "Spot")
366                 {
367                     foreach (Reference refe in refElemLinked)
368                     {
369                         pipes.Add(refe);
370                     }
371                 }
372
373
374                 #endregion
375
376                 (bool isUp, bool isHorizontal, bool isLeft,
                    IList<Reference> sorted_list) = Picker.is_UD_HV_LR(uidoc,
                    pipes, pickPoint1, pickPoint2);
377
378                 #region [DEBUG only] Check UP/Left/Hoeizontal
379                 //string UD = isUp ? "U" : "D";
380                 //string LR = isLeft ? "L" : "R";
381                 //string HV = isHorizontal ? "H" : "V";
382                 //MessageBox.Show($"{UD}{LR}{HV}");
383                 #endregion
384
385             // Initial tag location for recursive location
386
387                 XYZ first_proj_on_pipe =
                    Common_Tool.host_link_element_All_in_one(sorted_list[0],
                    doc, "Projection", pickPoint1);
388                 Line normLine = Line.CreateBound(first_proj_on_pipe,
                    pickPoint1);
389                 XYZ normal_dir = normLine.Direction.Normalize();
390                 int num_placement = 0;
391
392
393
394                 //For pip in pipes -> Get location with Host element or
```

```
                    linked instance
395                 foreach (Reference refe in sorted_list)
396                 {
397
398                     //1. Get Element from host / linked
399                     Element elem =
                    Common_Tool.host_link_element_All_in_one(refe, doc,
                    "Element");
400
401                     //2. Get pipe's location
402                     XYZ proj_on_pipe =
                    Common_Tool.host_link_element_All_in_one(refe, doc,
                    "Projection", pickPoint1);
403
404                     XYZ extented_pt = new XYZ(num_placement *
                    Properties.Settings.Default.label_dist_mm / 304.8 *
                    normal_dir.X,
405                                               num_placement *
                    Properties.Settings.Default.label_dist_mm / 304.8 *
                    normal_dir.Y,
406                                               0.0);
407
408                     //3. Place Independent Tag
409                     Placer.Create_Tag_or_Spot(doc, elem, proj_on_pipe,
                    extented_pt + pickPoint1, refe, isUp, isHorizontal, isLeft,
                     place_type);
410                     num_placement += 1;
411                 }
412                 t.Commit();
413             }
414
415             #region Test
416             //string pipes_xyzs = "";
417             //int count = 1;
418             //foreach (XYZ xyz in xyzs)
419             //{
420             //    pipes_xyzs += "####\n" + (count / 2).ToString() + "\n";
421             //    pipes_xyzs += xyz.ToString() + "\n";
422             //    count += 1;
423             //}
424             //MessageBox.Show("pipes.Count" + pipes.Count.ToString());
425             //MessageBox.Show("xyzss.Count" + xyzs.Count.ToString());
426             //MessageBox.Show(pipes_xyzs);
427             //string output = "";
428             //foreach (Reference elem in refElem) { output +=
                  elem.ElementId.ToString() + "\n"; }
429             //foreach (Reference elem in refElemLinked) { output +=
                  elem.ElementId.ToString() + "\n"; }
430             //MessageBox.Show(output);
431
432             #endregion
433
434             return Result.Succeeded;
```

```
435             }
436
437             catch (Exception e)
438             {
439                 MessageBox.Show(e.Message.ToString());
440                 message = e.Message;
441                 return Result.Failed;
442             }
443         }
444
445         public static IndependentTag Create_Tag_or_Spot(Document document,
              Element elem, XYZ on_pipe, XYZ tag_start, Reference refe, bool isUp,
               bool isHorizontal, bool isLeft, string place_type)
446         {
447             // Make sure active view is not a 3D view
448             var view = document.ActiveView;
449
450             // Define tag mode and tag orientation for new tag
451             TagMode tagMode = TagMode.TM_ADDBY_CATEGORY;
452
453             // Add the tag to the projected point from Click_Pt to picked pipe
                  of the wall
454             Reference elemRef = Common_Tool.host_link_element_All_in_one(refe,
                  document, "Reference");
455
456             #region Create tag
457
458             IndependentTag newTag = IndependentTag.Create(document, view.Id,
                elemRef, true, tagMode, TagOrientation.Horizontal, on_pipe);
459             if (null == newTag) throw new Exception("Create IndependentTag
                Failed.");
460
461             newTag.ChangeTypeId(getPipeTagId(document, refe));
462
463             newTag.LeaderEndCondition = LeaderEndCondition.Free;
464             newTag.LeaderElbow = tag_start;
465             newTag.LeaderEnd = on_pipe;
466
467             XYZ header_pos = tag_start;
468
469             // All diff are measure in foot
470             // 1 ft = 12 * 25.4 = 304.8 mm
471
472             double Tag_text_length = newTag.get_BoundingBox(view).Max.X -
                newTag.TagHeadPosition.X;
473
474             double label_length =
                Properties.Settings.Default.label_length_mm / 304.8;
475
476             if (place_type != "Spot")
477             {
478                 if (isLeft & isHorizontal) { header_pos += new XYZ(-
                    label_length - Tag_text_length, 0.0, 0.0); }
```

```
479                 if (isLeft & !isHorizontal) { header_pos += new XYZ(0.0, -
                      label_length - Tag_text_length, 0.0); }
480             }
481             else
482             {
483                 if (isLeft & isHorizontal) { header_pos += new XYZ(-
                      label_length, 0.0, 0.0); }
484                 if (isLeft & !isHorizontal) { header_pos += new XYZ(0.0, -
                      label_length, 0.0); }
485             }
486
487             if (!isLeft & isHorizontal) { header_pos += new XYZ(label_length,
                  0.0, 0.0); }
488             if (!isLeft & !isHorizontal) { header_pos += new XYZ(0.0,
                  label_length, 0.0); }
489
490             newTag.TagHeadPosition = header_pos;
491
492             // Default orientation = Horizontal
493             if (!isHorizontal) { newTag.TagOrientation =
                  TagOrientation.Vertical; }
494
495             newTag.LeaderEndCondition = LeaderEndCondition.Attached;
496
497             #endregion
498
499             #region Create spot dimension & del tag
500
501
502             if (place_type == "Spot")
503             {
504                 SpotDimension new_spot = document.Create.NewSpotElevation
                      (view, elemRef, on_pipe, tag_start, header_pos, on_pipe,
                      true);
505
506                 document.Delete(newTag.Id);
507             }
508
509             #endregion
510
511             return newTag;
512         }
513         public static ElementId getPipeTagId(Document doc, Reference refe)
514         {
515             //Element pipe = doc.GetElement(refe);
516
517             Element pipe = Common_Tool.host_link_element_All_in_one(refe, doc,
                  "Element");
518             ElementId pipeTagId = null;
519
520             // Create a list to store the tags
521             IList<Element> pipeTags = new FilteredElementCollector
                  (doc).OfClass(typeof(Family)).Where(x =>
```

```
522                                 x.Name ==
    Properties.Settings.Default.PI_1st_Selected.ToString() ||
523                                 x.Name ==
    Properties.Settings.Default.DT_C_1st_Selected.ToString() ||
524                                 x.Name ==
    Properties.Settings.Default.DT_R_1st_Selected.ToString() ||
525                                 x.Name ==
    Properties.Settings.Default.CN_1st_Selected.ToString() ||
526                                 x.Name ==
    Properties.Settings.Default.CT_1st_Selected.ToString()).ToList();
527
528             // Match the [ Pipe.Category.Name ] & [ Pipe Tag ]
529             foreach (Element elem in pipeTags)
530             {
531                 Family family = elem as Family;
532                 ISet<ElementId> familySymbolIds = family.GetFamilySymbolIds();
533
534
535
536                 if (pipe.Category.Name is "管")
537                 {
538                     if (family.Name ==
                    Properties.Settings.Default.PI_1st_Selected.ToString())  //
                     e.g. "L_圓形管標籤_標註符號"
539                     {
540                         foreach (ElementId elemId in familySymbolIds)
541                         {
542                             FamilySymbol familySymbol = doc.GetElement(elemId)
                         as FamilySymbol;
543
544
545                             if (familySymbol.Name ==
                        Properties.Settings.Default.PI_2nd_Selected) // e.g. "標稱
                        +系統+BOP/COP"
546                             {
547                                 pipeTagId = familySymbol.Id;
548                                 break;
549                             }
550                         }
551                         break;
552                     }
553                 }
554                 else if (pipe.Category.Name is "風管")
555                 {
556                     Autodesk.Revit.DB.Mechanical.Duct duct = pipe as
                    Autodesk.Revit.DB.Mechanical.Duct;
557                     Autodesk.Revit.DB.Mechanical.DuctType ductType =
                    duct.DuctType;
```

```
558                     if (ductType.FamilyName is "圓形風管")
559                     {
560                         if (family.Name ==                                  ↪
                    Properties.Settings.Default.DT_C_1st_Selected.ToString())
561                         {
562                             foreach (ElementId elemId in familySymbolIds)
563                             {
564                                 FamilySymbol familySymbol = doc.GetElement  ↪
                    (elemId) as FamilySymbol;
565                                 if (familySymbol.Name ==                    ↪
                    Properties.Settings.Default.DT_C_2nd_Selected)
566                                 {
567                                     pipeTagId = familySymbol.Id;
568                                     break;
569                                 }
570                             }
571                             break;
572                         }
573                     }
574                     else
575                     {
576                         if (family.Name ==                                  ↪
                    Properties.Settings.Default.DT_R_1st_Selected.ToString())
577                         {
578                             foreach (ElementId elemId in familySymbolIds)
579                             {
580                                 FamilySymbol familySymbol = doc.GetElement  ↪
                    (elemId) as FamilySymbol;
581                                 if (familySymbol.Name ==                    ↪
                    Properties.Settings.Default.DT_R_2nd_Selected) //is "WxH+系↪
                    統+BOD")
582                                 {
583                                     pipeTagId = familySymbol.Id;
584                                     break;
585                                 }
586                             }
587                             break;
588                         }
589                     }
590                 }
591                 else if (pipe.Category.Name is "電管")
592                 {
593                     if (family.Name ==                                      ↪
                    Properties.Settings.Default.CN_1st_Selected.ToString())
594                     {
595                         foreach (ElementId elemId in familySymbolIds)
596                         {
597                             FamilySymbol familySymbol = doc.GetElement(elemId) ↪
                     as FamilySymbol;
598                             if (familySymbol.Name ==                        ↪
                    Properties.Settings.Default.CN_2nd_Selected) //is "材質+標 ↪
                    稱+BOP")
599                             {
```

```
600                                 pipeTagId = familySymbol.Id;
601                                 break;
602                             }
603                         }
604                         break;
605                     }
606                 }
607                 else if (pipe.Category.Name is "電纜架")
608                 {
609                     if (family.Name == Properties.Settings.Default.CT_1st_Selected.ToString())
610                     {
611                         foreach (ElementId elemId in familySymbolIds)
612                         {
613                             FamilySymbol familySymbol = doc.GetElement(elemId) as FamilySymbol;
614                             if (familySymbol.Name == Properties.Settings.Default.CT_2nd_Selected) // is "服務類型+標稱+BOD")
615                             {
616                                 pipeTagId = familySymbol.Id;
617                                 break;
618                             }
619                         }
620                         break;
621                     }
622                 }
623                 else
624                     MessageBox.Show("此物件不是任一種管");
625             }
626             return pipeTagId;
627         }
628     }
629     public class Common_Tool
630     {
631         public static dynamic host_link_element_All_in_one(Reference refe, Document doc, string return_type, XYZ pickPoint1 = null)
632         {
633 
634             ///<example>
635             ///
636             /// return [Element]    ==> Common_Tool.host_link_element_All_in_one(refe, document, "Element");
637             /// return [Reference]  ==> Common_Tool.host_link_element_All_in_one(refe, document, "Reference");
638             /// return [XYZ]        ==> Common_Tool.host_link_element_All_in_one(ref1, doc, "Projection", pickPoint1);
639             ///
640             /// </example>
641 
```

```
642             string[] return_types = { "Reference", "Element", "Projection" };
643
644             if (return_types.Contains(return_type) == false)
645             {
646                 MessageBox.Show("From func [host_link_element_All_in_one],
                      please input return_type either 'Reference', 'Element', or
                      'Projection'");
647                 return 0;
648             }
649
650             Element elem = doc.GetElement(refe);
651
652             LocationCurve locCurve;
653             Line line;
654
655             if (elem.GetType().Name != "RevitLinkInstance")
656             {
657                 elem = doc.GetElement(refe);
658                 locCurve = elem.Location as LocationCurve;
659                 line = locCurve.Curve as Line;
660
661                 if (return_type == "Reference")
662                 {
663                     return new Reference(elem);
664                 }
665             }
666             else
667             {
668                 RevitLinkInstance linkInstance = elem as RevitLinkInstance; //
                      將選取的物件轉型為外參檔
669                 Document docLink = linkInstance.GetLinkDocument(); // 指名新的
                      document存放該外參檔資料
670                 elem = docLink.GetElement(refe.LinkedElementId); // 從外參的
                      document裡尋找欲選擇的管，轉型成element
671
672                 Transform Link_trans = linkInstance.GetTotalTransform();
673                 locCurve = elem.Location as LocationCurve;
674                 line = locCurve.Curve as Line;
675                 line = (Line)line.CreateTransformed(Link_trans);
676
677                 if (return_type == "Reference")
678                 {
679                     return new Reference(docLink.GetElement
                          (refe.LinkedElementId)).CreateLinkReference(linkInstance);
680                 }
681
682             }
683
684             if (return_type == "Element") { return elem; }
685
686             if (return_type == "Projection")
687             {
688                 if (pickPoint1 == null) { MessageBox.Show("From func
```

```
                    [host_link_element_All_in_one], please input the pickpoint
                    if you want to get the project point"); }
689                 try { return line.Project(pickPoint1).XYZPoint; }
690                 catch (Exception e) { MessageBox.Show(e.Message.ToString()); }
691             }
692
693             return "haha";
694
695         }
696
697         #region 3 func combined as host_link_element_All_in_one
698
699         //public static Element Get_host_or_link_element(Reference refe,
              Document doc)
700         //{
701         //    Element elem = doc.GetElement(refe);
702
703         //    if (elem.GetType().Name != "RevitLinkInstance")
704         //    {
705         //        elem = doc.GetElement(refe);
706         //    }
707         //    else
708         //    {
709         //        RevitLinkInstance linkInstance = elem as
              RevitLinkInstance; // 將選取的物件轉型為外參檔
710         //        Document docLink = linkInstance.GetLinkDocument(); // 指名新
              的document存放該外參檔資料
711         //        elem = docLink.GetElement(refe.LinkedElementId); // 從外參的
              document裡尋找欲選擇的管，轉型成element
712         //        //MessageBox.Show(elem.Category.Name);
713         //    }
714
715         //    return elem;
716         //}
717
718         //public static Reference Create_new_reference(Reference refe,
              Document doc)
719         //{
720         //    Element elem = doc.GetElement(refe);
721
722         //    // Case 1: Host element
723         //    if (elem.GetType().Name != "RevitLinkInstance")
724         //    {
725         //        return new Reference(elem);
726         //    }
727         //    // Case 2: Linked element
728         //    else
729         //    {
730
731         //        RevitLinkInstance linkInstance = doc.GetElement(refe) as
              RevitLinkInstance;
732         //        Document docLink = linkInstance.GetLinkDocument();
733
```

```
734         //        Reference elemRef = new Reference(docLink.GetElement
              (refe.LinkedElementId)).CreateLinkReference(linkInstance);
735
736         //        return elemRef;
737
738         //    }
739
740         //}
741
742         //public static XYZ Get_transformed_project_pt(Reference refe,
              Document doc, XYZ pickPoint1)
743         //{
744         //    Element elem = doc.GetElement(refe);
745
746         //    XYZ proj_on_pipe;
747         //    LocationCurve locCurve;
748         //    Line line;
749
750         //    if (elem.GetType().Name != "RevitLinkInstance")
751         //    {
752         //        elem = doc.GetElement(refe);
753         //        locCurve = elem.Location as LocationCurve;
754         //        line = locCurve.Curve as Line;
755         //        proj_on_pipe = line.Project(pickPoint1).XYZPoint;
756         //    }
757         //    else
758         //    {
759         //        RevitLinkInstance linkInstance = elem as
              RevitLinkInstance; // 將選取的物件轉型為外參檔
760         //        XYZ trans = linkInstance.GetTotalTransform().Origin;
761         //        Transform Link_trans = linkInstance.GetTotalTransform();
762         //        Document docLink = linkInstance.GetLinkDocument(); // 指名新
              的document存放該外參檔資料
763         //        elem = docLink.GetElement(refe.LinkedElementId); // 從外參的
              document裡尋找欲選擇的管，轉型成element
764         //        locCurve = elem.Location as LocationCurve;
765         //        line = locCurve.Curve as Line;
766         //        Curve trans_line = line.CreateTransformed(Link_trans);
767
768         //        proj_on_pipe = trans_line.Project(pickPoint1).XYZPoint;
769         //    }
770
771         //    return proj_on_pipe;
772
773         //}
774
775
776         #endregion
777
778
779     }
780
781     public class UI_Tool {
```

```
782         public static List<String> Get_Tag_2nd_options(String family_Name,
              Document doc)
783         {
784             List<string> TypeOptions = new List<string>();
785
786
787             //IList<Element> pipeTags = new FilteredElementCollector
                  (doc).OfClass(typeof(Family)).Where(x => x.Name is "L_圓形管標籤
                  _標註符號" ||
788             //                                      x.Name is "L_圓形風管標籤_標註符
                  號" ||
789             //                                      x.Name is "L_方形風管標籤_標註符
                  號" ||
790             //                                      x.Name is "L_電管標籤_標註符號"
                  ||
791             //                                      x.Name is "L_電纜架標籤_標註符
                  號").ToList();
792
793             IList<Element> pipeTags = new FilteredElementCollector
                  (doc).OfClass(typeof(Family)).Where(x => x.Name.Contains
                  (family_Name)).ToList();
794
795             if (pipeTags.Count == 0) { return TypeOptions; }
796             //MessageBox.Show("Get_Tag_2nd_options!!!");
797
798                 foreach (Element elem in pipeTags)
799             {
800                 Family family = elem as Family;
801                 ISet<ElementId> familySymbolIds = family.GetFamilySymbolIds();
802
803                 if (family.Name == family_Name)
804                 {
805                     foreach (ElementId elemId in familySymbolIds)
806                     {
807                         FamilySymbol familySymbol = doc.GetElement(elemId) as
                        FamilySymbol;
808                         TypeOptions.Add(familySymbol.Name);
809                     }
810                 }
811
812                 #region Reaplce by above code
813                 //if (type is "管")
814                 //{
815                 //    if (family.Name is "L_圓形管標籤_標註符號")
816                 //    {
817                 //        foreach (ElementId elemId in familySymbolIds)
818                 //        {
819                 //            FamilySymbol familySymbol = doc.GetElement
                      (elemId) as FamilySymbol;
```

```
820                 //            TypeOptions.Add(familySymbol.Name);
821                 //            //MessageBox.Show("L_圓形管標籤_標註符號" +
                      familySymbol.Name);
822                 //        }
823                 //    }
824                 //}
825
826                 //else if (type is "圓風管")
827                 //{
828                 //    if (family.Name is "L_圓形風管標籤_標註符號")
829                 //    {
830                 //        foreach (ElementId elemId in familySymbolIds)
831                 //        {
832                 //            FamilySymbol familySymbol = doc.GetElement
                      (elemId) as FamilySymbol;
833                 //            TypeOptions.Add(familySymbol.Name);
834                 //            //MessageBox.Show("L_圓形風管標籤_標註符號" +
                      familySymbol.Name);
835                 //        }
836                 //    }
837                 //}
838                 //else if (type is "方風管")
839                 //{
840                 //    if (family.Name is "L_方形風管標籤_標註符號")
841                 //    {
842                 //        foreach (ElementId elemId in familySymbolIds)
843                 //        {
844                 //            FamilySymbol familySymbol = doc.GetElement
                      (elemId) as FamilySymbol;
845                 //            TypeOptions.Add(familySymbol.Name);
846                 //            //MessageBox.Show("L_方形風管標籤_標註符號" +
                      familySymbol.Name);
847                 //        }
848                 //    }
849                 //}
850                 //else if (type is "電管")
851                 //{
852                 //    if (family.Name is "L_電管標籤_標註符號")
853                 //    {
854                 //        foreach (ElementId elemId in familySymbolIds)
855                 //        {
856                 //            FamilySymbol familySymbol = doc.GetElement
                      (elemId) as FamilySymbol;
857                 //            TypeOptions.Add(familySymbol.Name);
858                 //            //MessageBox.Show("L_電管標籤_標註符號" +
                      familySymbol.Name);
859                 //        }
860                 //    }
861                 //}
862                 //else if (type is "電纜架")
863                 //{
864                 //    if (family.Name is "L_電纜架標籤_標註符號")
865                 //    {
```

```
866                 //        foreach (ElementId elemId in familySymbolIds)
867                 //        {
868                 //            FamilySymbol familySymbol = doc.GetElement
                      (elemId) as FamilySymbol;
869                 //            TypeOptions.Add(familySymbol.Name);
870                 //            //MessageBox.Show("L_電纜架標籤_標註符號" +
                      familySymbol.Name);
871                 //        }
872                 //    }
873                 //}
874
875                 #endregion
876
877                 //else MessageBox.Show("此物件不是任一種管");
878             }
879             return TypeOptions;
880         }
881
882         public static List<String> Get_Tag_1st_options(String type, Document
              doc)
883         {
884             List<string> TypeOptions = new List<string>();
885
886             if (type is "管")
887             {
888                 IList<Element> pipeTags = new FilteredElementCollector
                      (doc).OfCategory(BuiltInCategory.OST_PipeTags).OfClass
                      (typeof(FamilySymbol)).WhereElementIsElementType
                      ().ToElements();
889                 List<String> option = new List<String>();
890
891                 foreach (Element elem in pipeTags)
892                 {
893                     option.Add(((ElementType)elem).FamilyName.ToString());
894                 }
895                 option = option.Distinct().ToList();
896                 //foreach (String opt in option) { MessageBox.Show(opt); }
897                 return option;
898             }
899
900             else if (type is "圓風管")
901             {
902                 IList<Element> pipeTags = new FilteredElementCollector
                      (doc).OfCategory(BuiltInCategory.OST_DuctTags).OfClass
                      (typeof(FamilySymbol)).WhereElementIsElementType
                      ().ToElements();
903                 List<String> option = new List<String>();
904
905                 foreach (Element elem in pipeTags)
906                 {
907                     option.Add(((ElementType)elem).FamilyName.ToString());
908                 }
909                 option = option.Distinct().ToList();
```

```
910                 //foreach (String opt in option) { MessageBox.Show(opt); }
911                 return option;
912             }
913             else if (type is "方風管")
914             {
915                 IList<Element> pipeTags = new FilteredElementCollector
                      (doc).OfCategory(BuiltInCategory.OST_DuctTags).OfClass
                      (typeof(FamilySymbol)).WhereElementIsElementType
                      ().ToElements();
916                 List<String> option = new List<String>();
917 
918                 foreach (Element elem in pipeTags)
919                 {
920                     option.Add(((ElementType)elem).FamilyName.ToString());
921                 }
922                 option = option.Distinct().ToList();
923                 //foreach (String opt in option) { MessageBox.Show(opt); }
924                 return option;
925             }
926             else if (type is "電管")
927             {
928                 IList<Element> pipeTags = new FilteredElementCollector
                      (doc).OfCategory(BuiltInCategory.OST_ConduitTags).OfClass
                      (typeof(FamilySymbol)).WhereElementIsElementType
                      ().ToElements();
929                 List<String> option = new List<String>();
930 
931                 foreach (Element elem in pipeTags)
932                 {
933                     option.Add(((ElementType)elem).FamilyName.ToString());
934                 }
935                 option = option.Distinct().ToList();
936                 //foreach (String opt in option) { MessageBox.Show(opt); }
937                 return option;
938             }
939             else if (type is "電纜架")
940             {
941                 IList<Element> pipeTags = new FilteredElementCollector
                      (doc).OfCategory(BuiltInCategory.OST_CableTrayTags).OfClass
                      (typeof(FamilySymbol)).WhereElementIsElementType
                      ().ToElements();
942                 List<String> option = new List<String>();
943 
944                 foreach (Element elem in pipeTags)
945                 {
946                     option.Add(((ElementType)elem).FamilyName.ToString());
947                 }
948                 option = option.Distinct().ToList();
949                 //foreach (String opt in option) { MessageBox.Show(opt); }
950                 return option;
951             }
952             //else MessageBox.Show("此物件不是任一種管");
953             return TypeOptions;
```

```
954         }
955
956         public static BitmapSource GetImageSource(Image img)
957         {
958             //製作一個function專門來處理圖片
959             BitmapImage bmp = new BitmapImage();
960
961             using (MemoryStream ms = new MemoryStream())
962             {
963                 img.Save(ms, ImageFormat.Png);
964                 ms.Position = 0;
965
966                 bmp.BeginInit();
967
968                 bmp.CacheOption = BitmapCacheOption.OnLoad;
969                 bmp.UriSource = null;
970                 bmp.StreamSource = ms;
971
972                 bmp.EndInit();
973             }
974
975             return bmp;
976         }
977
978     }
979 }
```