```
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using Autodesk.Revit.Attributes;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Autodesk.Revit.UI.Selection;
using PipeTagger.Windows;

namespace PipeTagger
{
    [Transaction(TransactionMode.Manual)]
    public class NotFinish : IExternalCommand
    {
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
        {
            UIDocument uidoc = commandData.Application.ActiveUIDocument;
            Document doc = uidoc.Document;

            try
            {
                NothingWindow window = new NothingWindow(uidoc);
                //window.Label_Name.Content = Properties.Settings.Default.Test_text;
                window.ShowDialog();

                return Result.Succeeded;
            }

            catch (Exception e)
            {
                MessageBox.Show(e.Message.ToString());
                message = e.Message;
                return Result.Failed;
            }
        }
    }

    [Transaction(TransactionMode.Manual)]
    public class Tag_setting : IExternalCommand
    {
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
        {
            UIDocument uidoc = commandData.Application.ActiveUIDocument;
            Document doc = uidoc.Document;

            try
            {
                setting_tag window = new setting_tag(uidoc);
                window.ShowDialog();
```

```
51
52                  return Result.Succeeded;
53              }
54
55              catch (Exception e)
56              {
57                  MessageBox.Show(e.Message.ToString());
58                  message = e.Message;
59                  return Result.Failed;
60              }
61          }
62      }
63
64
65      [Transaction(TransactionMode.Manual)]
66      public class Place_tag : IExternalCommand
67      {
68          public Result Execute(ExternalCommandData commandData, ref string
              message, ElementSet elements)
69          {
70              UIDocument uidoc = commandData.Application.ActiveUIDocument;
71              Document doc = uidoc.Document;
72
73              return Common.Placer.CreateSomething(uidoc, doc, ref message,
                  "Tag");
74
75          }
76      }
77
78      [Transaction(TransactionMode.Manual)]
79      public class Place_spot : IExternalCommand
80      {
81          public Result Execute(ExternalCommandData commandData, ref string
              message, ElementSet elements)
82          {
83              UIDocument uidoc = commandData.Application.ActiveUIDocument;
84              Document doc = uidoc.Document;
85
86              return Common.Placer.CreateSomething(uidoc, doc, ref message,
                  "Spot");
87
88          }
89      }
90
91
92  }
93
94
95  #region References
96
97  #region A. How to add tag
98  #region 1. "HorizontalContiTagged" -- > Mainv4_1 -->  Line 156 ()
99
```

```
100 //Go To  "HorizontalContiTagged" -- > Mainv4_1 -->  Line 156 ()
101
102
103 ///// <summary>
104 ///// 建立水管管徑標註
105 ///// </summary>
106 //[Transaction(TransactionMode.Manual)]
107 //public class CreatPipeDiameterTag : IExternalCommand
108 //{
109 //    #region IExternalCommand Members
110
111 //    public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
112 //    {
113 //        UIDocument uiDoc = commandData.Application.ActiveUIDocument;
114 //        Document doc = uiDoc.Document;//當前活動文件
115 //        Autodesk.Revit.DB.View view = uiDoc.ActiveView;//當前活動檢視
116 //        Selection sel = uiDoc.Selection;//選擇集
117 //        Transaction ts = new Transaction(doc, "水管管徑標記");
118 //        try
119 //        {
120 //            ts.Start();
121 //            PipeSelectionFilter psf = new PipeSelectionFilter(doc);
122 //            Reference refer = sel.PickObject(ObjectType.Element, psf, "請選擇要標註的水管:");
123 //            Pipe pipe = doc.GetElement(refer) as Pipe;
124 //            if (pipe == null)
125 //            {
126 //                ts.Dispose();
127 //                TaskDialog.Show("RevitMassge", "沒有選中水管");
128 //                return Result.Failed;
129 //            }
130 //            //Define tag mode and tag orientation for new tag
131 //            TagMode tageMode = TagMode.TM_ADDBY_CATEGORY;
132 //            TagOrientation tagOri = TagOrientation.Horizontal;
133 //            //Add the tag to the middle of duct
134 //            LocationCurve locCurve = pipe.Location as LocationCurve;
135 //            XYZ pipeMid = locCurve.Curve.Evaluate(0.275, true);
136
137
138 //            IndependentTag tag = IndependentTag.Create(doc, 399884, view.Id, ); //(https://www.revitapidocs.com/2019/b8e8eec2-8e3b-08f2-a9a5-89f24465c8b9.htm)
139 //                                                                                //view, pipe, false, tageMode, tagOri, pipeMid)';
140
141 //            //遍歷型別
142 //            FilteredElementCollector filterColl = GetElementsOfType(doc, typeof(FamilySymbol), BuiltInCategory.OST_PipeTags);
143 //            //WinFormTools.MsgBox(filterColl.ToElements().Count.ToString());
144 //            int elId = 0;
145 //            foreach (Element el in filterColl.ToElements())
146 //            {
```

```
147 //                if (el.Name == "管道尺寸標記")
148 //                    elId = el.Id.IntegerValue;
149 //            }
150 //            tag.ChangeTypeId(new ElementId(elId));
151 //            ElementId eId = null;
152 //            if (tag == null)
153 //            {
154 //                ts.Dispose();
155 //                TaskDialog.Show("RevitMassge", "建立標註失敗!");
156 //                return Result.Failed;
157 //            }
158 //            ICollection<ElementId> eSet = tag.GetValidTypes();
159 //            foreach (ElementId item in eSet)
160 //            {
161 //                if (item.IntegerValue == 532753)
162 //                {
163 //                    eId = item;
164 //                }
165 //            }
166 //            tag = doc.GetElement(eId) as IndependentTag;
167 //        }
168 //        catch (Exception)
169 //        {
170 //            ts.Dispose();
171 //            return Result.Cancelled;
172 //        }
173 //        ts.Commit();
174 
175 //        return Result.Succeeded;
176 //    }
177 //    FilteredElementCollector GetElementsOfType(Document doc, Type type,
      BuiltInCategory bic)
178 //    {
179 //        FilteredElementCollector collector = new FilteredElementCollector
      (doc);
180 
181 //        collector.OfCategory(bic);
182 //        collector.OfClass(type);
183 
184 //        return collector;
185 //    }
186 //    #endregion
187 //}
188 
189 ///// <summary>
190 /////水管選擇過濾器
191 ///// </summary>
192 //public class PipeSelectionFilter : ISelectionFilter
193 //{
194 //    #region ISelectionFilter Members
195 
196 //    Document doc = null;
197 //    public PipeSelectionFilter(Document document)
```

```
198 //    {
199 //        doc = document;
200 //    }
201 
202 //    public bool AllowElement(Element elem)
203 //    {
204 //        return elem is Pipe;
205 //    }
206 
207 //    public bool AllowReference(Reference reference, XYZ position)
208 //    {
209 //        return doc.GetElement(reference) is Pipe;
210 //    }
211 
212 //    #endregion
213 //}
214 
215 #endregion
216 
217 #region 2. Online example  https://forums.autodesk.com/t5/revit-api-forum/
      independenttag-how-do-i-call-this-in-revit/td-p/7733731
218 
219 //[Transaction(TransactionMode.Manual)]
220 //public class Tagtest : IExternalCommand
221 //{
222 //    #region Methods
223 
224 //    /// <summary>
225 //    ///       The CreateIndependentTag
226 //    /// </summary>
227 //    /// <param name="document">The <see cref="Document" /></param>
228 //    /// <param name="wall">The <see cref="Wall" /></param>
229 //    /// <returns>The <see cref="IndependentTag" /></returns>
230 //    public IndependentTag CreateIndependentTag(Document document, Wall wall)
231 //    {
232 //        TaskDialog.Show("Create Independent Tag Method", "Start Of Method
      Dialog");
233 //        // make sure active view is not a 3D view
234 //        var view = document.ActiveView;
235 
236 //        // define tag mode and tag orientation for new tag
237 //        var tagMode = TagMode.TM_ADDBY_CATEGORY;
238 //        var tagorn = TagOrientation.Horizontal;
239 
240 //        // Add the tag to the middle of the wall
241 //        var wallLoc = wall.Location as LocationCurve;
242 //        var wallStart = wallLoc.Curve.GetEndPoint(0);
243 //        var wallEnd = wallLoc.Curve.GetEndPoint(1);
244 //        var wallMid = wallLoc.Curve.Evaluate(0.5, true);
245 //        var wallRef = new Reference(wall);
246 
247 //        var newTag = IndependentTag.Create(document, view.Id, wallRef, true,
      tagMode, tagorn, wallMid);
```

```
248 //        if (null == newTag) throw new Exception("Create IndependentTag Failed.");
249
250 //        // newTag.TagText is read-only, so we change the Type Mark type parameter to
251 //        // set the tag text.  The label parameter for the tag family determines
252 //        // what type parameter is used for the tag text.
253
254 //        var type = wall.WallType;
255
256 //        var foundParameter = type.LookupParameter("Type Mark");
257 //        //var result = foundParameter.Set("Hello");
258
259 //        // set leader mode free
260 //        // otherwise leader end point move with elbow point
261
262 //        newTag.LeaderEndCondition = LeaderEndCondition.Free;
263 //        var elbowPnt = wallMid + new XYZ(5.0, 5.0, 0.0);
264 //        newTag.LeaderElbow = elbowPnt;
265 //        var headerPnt = wallMid + new XYZ(10.0, 10.0, 0.0);
266 //        newTag.TagHeadPosition = headerPnt;
267
268 //        TaskDialog.Show("Create Independent Tag Method", "End Of Method Dialog");
269
270 //        return newTag;
271 //    }
272
273 //    public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
274 //    {
275 //        UIDocument uidoc = commandData.Application.ActiveUIDocument;
276 //        Document doc = uidoc.Document;
277
278 //        try
279 //        {
280 //            Reference r = uidoc.Selection.PickObject (Autodesk.Revit.UI.Selection.ObjectType.Element);
281
282 //            Wall w = doc.GetElement(r.ElementId) as Wall;
283
284 //            using (Transaction t = new Transaction(doc, "create tag"))
285 //            {
286 //                t.Start();
287
288 //                CreateIndependentTag(doc, w);
289
290 //                t.Commit();
291 //            }
292 //        }
293 //        catch (Exception e)
294 //        {
```

```
295 //            message = e.Message;
296 //            return Result.Failed;
297 //        }
298 
299 //        return Result.Succeeded;
300 //    }
301 
302 //    #endregion
303 //}
304 
305 #endregion
306 
307 #region 3. Add tag for Linked reference https://forums.autodesk.com/t5/revit-api-forum/tagging-linked-elements-using-revit-api/m-p/8671548#M37534
308 /// <summary>
309 /// Tag all walls in all linked documents
310 /// </summary>
311 //void TagAllLinkedWalls(Document doc)
312 //{
313 //    // Point near my wall
314 //    XYZ xyz = new XYZ(-20, 20, 0);
315 
316 //    // At first need to find our links
317 //    FilteredElementCollector collector
318 //        = new FilteredElementCollector(doc)
319 //        .OfClass(typeof(RevitLinkInstance));
320 
321 //    foreach (Element elem in collector)
322 //    {
323 //        // Get linkInstance
324 //        RevitLinkInstance instance = elem
325 //            as RevitLinkInstance;
326 
327 //        // Get linkDocument
328 //        Document linkDoc = instance.GetLinkDocument();
329 
330 //        // Get linkType
331 //        RevitLinkType type = doc.GetElement(
332 //            instance.GetTypeId()) as RevitLinkType;
333 
334 //        // Check if link is loaded
335 //        if (RevitLinkType.IsLoaded(doc, type.Id))
336 //        {
337 //            // Find walls for tagging
338 //            FilteredElementCollector walls
339 //                = new FilteredElementCollector(linkDoc)
340 //                .OfCategory(BuiltInCategory.OST_Walls)
341 //                .OfClass(typeof(Wall));
342 
343 //            // Create reference
344 //            foreach (Wall wall in walls)
345 //            {
346 //                Reference newRef = new Reference(wall)
```

```
347 //                      .CreateLinkReference(instance);
348 
349 //                  // Create transaction
350 //                  using (Transaction tx = new Transaction(doc))
351 //                  {
352 //                      tx.Start("Create tags");
353 
354 //                      IndependentTag newTag = IndependentTag.Create(
355 //                          doc, doc.ActiveView.Id, newRef, true,
356 //                          TagMode.TM_ADDBY_MATERIAL,
357 //                          TagOrientation.Horizontal, xyz);
358 
359 //                      // Use TaggedElementId.LinkInstanceId and
360 //                      // TaggedElementId.LinkInstanceId to retrieve
361 //                      // the id of the tagged link and element:
362 
363 //                      LinkElementId linkId = newTag.TaggedElementId;
364 //                      ElementId linkInsancetId = linkId.LinkInstanceId;
365 //                      ElementId linkedElementId = linkId.LinkedElementId;
366 
367 //                      tx.Commit();
368 //                  }
369 //              }
370 //          }
371 //      }
372 //}
373 
374 
375 #endregion
376 
377 #endregion
378 
379 #region How to change tag style
380 
381 
382 #region 1. Online example https://adndevblog.typepad.com/aec/2013/01/setting-the-leader-arrowhead-for-a-structural-framing-tag-using-revit-api.html
383 
384 //namespace Revit.SDK.Samples.HelloRevit.CS
385 
386 //{
387 
388 //    [Transaction(TransactionMode.Manual)]
389 //    public class Command : IExternalCommand{
390 //        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
391 //        {
392 //            UIApplication uiApp = commandData.Application;
393 //            Document doc = uiApp.ActiveUIDocument.Document;
394 
395 //            // Access all elements in the model which represent Arrowheads
396 //            // This is being done by filtering all elements which
397 //            // are of ElementType and have the ALL_MODEL_FAMILY_NAME
```

```
398 //            // builtIn Parameter set to 'Arrowhead'
399 
400 //            ElementId id = new ElementId
      (BuiltInParameter.ALL_MODEL_FAMILY_NAME);
401 //            ParameterValueProvider provider = new ParameterValueProvider
      (id);
402 //            FilterStringRuleEvaluator evaluator = new FilterStringEquals();
403 //            FilterRule rule = new FilterStringRule(provider, evaluator,
      "Arrowhead", false);
404 //            ElementParameterFilter filter = new ElementParameterFilter
      (rule);
405 //            FilteredElementCollector collector = new
      FilteredElementCollector(doc).OfClass(typeof(ElementType)).WherePasses
      (filter);
406 
407 
408 //            using (Transaction trans = new Transaction(doc, "Arrowhead"))
409 //            {
410 //                trans.Start();
411 //                // For simplicity, assuming that the
412 //                // Structural Component Tag is selected
413 //                foreach (Element selectedElement in
414 //                  uiApp.ActiveUIDocument.Selection.Elements)
415 //                {
416 //                    IndependentTag tag = selectedElement as IndependentTag;
417 
418 //                    if (null != tag)
419 //                    {
420 //                        // Access the Symbol of the IndependentTag element
421 //                        FamilySymbol tagSymbol = doc.GetElement
      (tag.GetTypeId()) as FamilySymbol;
422 //                        // Set the LEADER_ARROWHEAD parameter of the
423 //                        // Symbol with one of the arrowheads that was
      filtered
424 //                        tagSymbol.get_Parameter
      (BuiltInParameter.LEADER_ARROWHEAD).Set(collector.ToElementIds
      ().ElementAt<ElementId>(0));
425 //                    }
426 //                    trans.Commit();
427 //                }
428 //            }
429 //            return Result.Succeeded;
430 //        }
431 //    }
432 //}
433 
434 #endregion
435 
436 #endregion
437 
438 #endregion
```