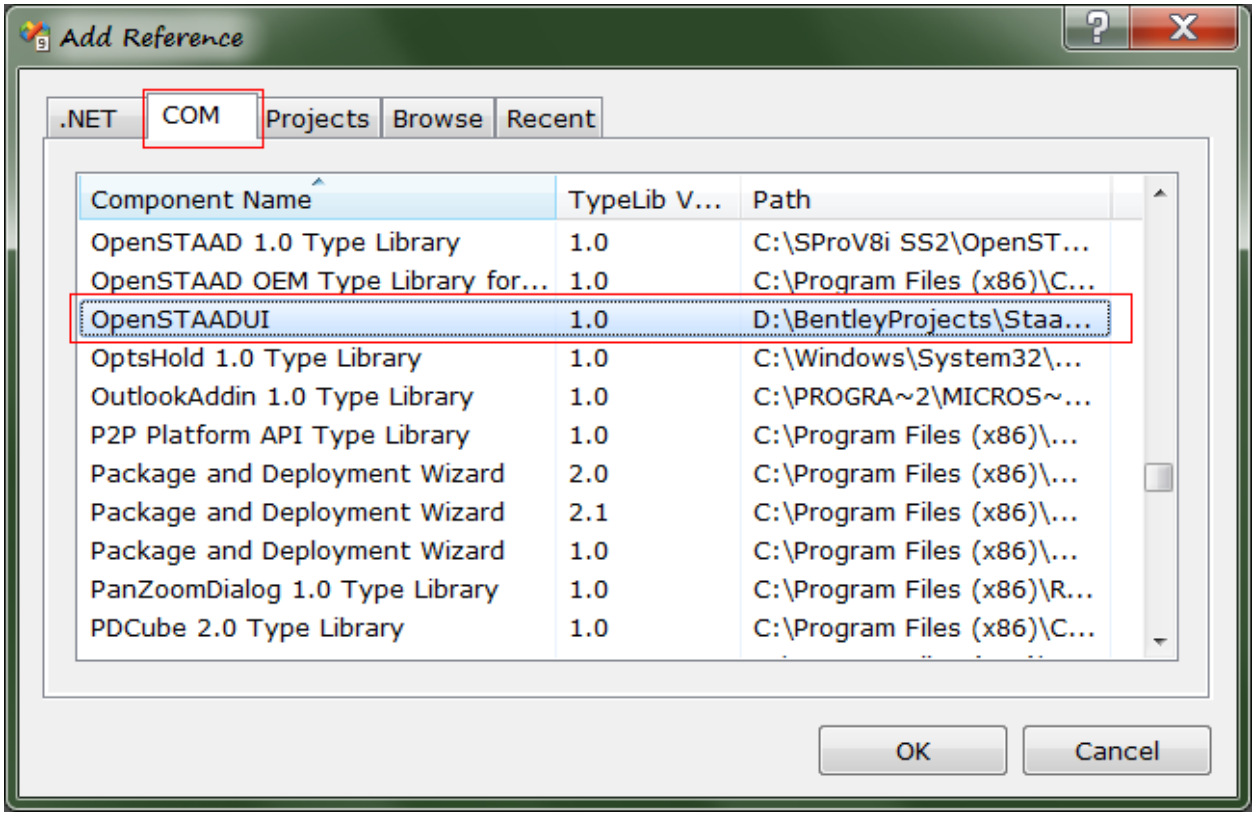
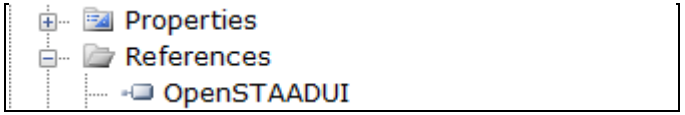
Using OpenSTAAD application object (C#)

Include OpenSTAADUI Library

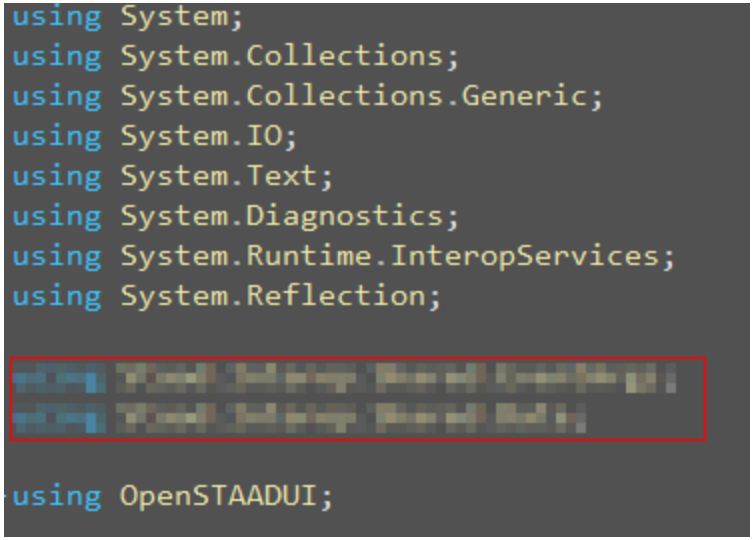
1. Right click on ‘Reference’ node and click on ‘Add References’
2. This will bring up the ‘Add Reference’ dialog box. Select ‘COM’ tab and scroll down to find ‘OpenSTAADUI’ object. (Refer diagram below).



3. Click ‘OK’ to add ‘OpenSTAADUI’ to project.



4. Now, at the top of code file add the following line,



Connection

The following section will describe how to connect to the STAAD.Pro application using OpenSTAADUI object.

Get list of opened STAAD files,

```
public static List<string> GetOpenedStaadFiles()
{
      try
      {
            List<string> fileList = null;
            Hashtable addedFileNames = new Hashtable();

            Type type = Type.GetTypeFromProgID("StaadPro.OpenSTAAD");
            string clsid = type.GUID.ToString();
            string lookUpCandidateName = String.Format("!{0}{1}{2}", "{", clsid, "}").ToUpper();

            List<RunningObjectInfo> runningObjects = GetRunningObjectList();
            foreach (RunningObjectInfo item in runningObjects)
            {
                  string candidateName = item.Name.ToUpper();
                  // check if its a valid STAAD file
                  string extStr = Path.GetExtension(candidateName);
                  if (extStr != null)
                  {
                        if (!String.IsNullOrEmpty(extStr))
```

```
                    {
                        if (extStr == ".STD")
                        {
                            // we have a std file opened, but this can be opened by other applications
also,
                            // we need to get the staadobject to see if it is opened by STAAD
                            OpenSTAAD osObject = item.Value as OpenSTAAD;
                            if (osObject != null)
                            {
                                if (fileList == null)
                                    fileList = new List<string>();

                                if (!addedFileNames.Contains(candidateName.ToUpper()))
                                {
                                    fileList.Add(candidateName.ToUpper());
                                    addedFileNames.Add(candidateName.ToUpper(), "");
                                }
                            }
                        }
                    }

                    if (candidateName.StartsWith(lookUpCandidateName))
                    {
                        // we got the staad instance
                        OpenSTAAD osObject = item.Value as OpenSTAAD;
                        if (osObject != null)
                        {
                            // get staad filename
                            string staadFilename = "";
                            bool includeFullPath = true;
                            Object oArg1 = staadFilename as object;
                            Object oArg2 = includeFullPath as object;
                            osObject.GetSTAADFile(ref oArg1, oArg2);
                            staadFilename = oArg1 as string;
                            //

                            if (!String.IsNullOrEmpty(staadFilename))
                            {
                                if (fileList == null)
                                    fileList = new List<string>();

                                if (!addedFileNames.Contains(staadFilename.ToUpper()))
                                {
```

```
                                        fileList.Add(staadFilename.ToUpper());
                                        addedFileNames.Add(staadFilename.ToUpper(), "");
                                    }
                                }
                            }
                        }
                    }
                    return fileList;
            }
            catch (COMException)
            {
                    throw;
            }
}
```

Get the STAAD.Pro application associated with opened STAAD file,

```
public static OpenSTAAD GetSTAADUIInterface(string staadFilename)
{
      OpenSTAAD osObject = null;

      try
      {
            Type type = Type.GetTypeFromProgID("StaadPro.OpenSTAAD");
            string clsid = type.GUID.ToString();
            string lookUpCandidateName = String.Format("!{0}{1}{2}", "{", clsid, "}").ToUpper();

            List<RunningObjectInfo> runningObjects = GetRunningObjectList();
            foreach (RunningObjectInfo item in runningObjects)
            {
                  string candidateName = item.Name.ToUpper();
                  // check if its a valid STAAD file
                  string extStr = Path.GetExtension(candidateName);
                  if (extStr != null)
                  {
                        if (!String.IsNullOrEmpty(extStr))
                        {
                              if (extStr == ".STD")
                              {
                                    if (String.Compare(staadFilename.ToUpper(), candidateName.ToUpper()) == 0)
                                    {
                                          // we have a std file opened, but this can be opened by other
applications also,
```

```
                                    // we need to get the staadobject to see if it is opened by STAAD
                                    osObject = item.Value as OpenSTAAD;
                                    if (osObject != null)
                                    {
                                          return osObject;
                                    }
                              }
                        }
                  }

                  if (candidateName.StartsWith(lookUpCandidateName))
                  {
                        // we got the staad instance
                        osObject = item.Value as OpenSTAAD;

                        if (osObject != null)
                        {
                              // get staad filename
                              string lookupStaadFilename = "";
                              bool includeFullPath = true;
                              Object oArg1 = lookupStaadFilename as object;
                              Object oArg2 = includeFullPath as object;
                              osObject.GetSTAADFile(ref oArg1, oArg2);
                              lookupStaadFilename = oArg1 as string;

                              if (String.Compare(staadFilename.ToUpper(), lookupStaadFilename.ToUpper())
== 0)
                              {
                                    return osObject;
                              }
                        }
                  }
            }
      }
      catch (COMException)
      {
            throw;
      }

      return null;
}
```

Get STAAD.Pro instance count

```
public static int GetSTAADInstanceCount()
{
      int count = 0;
      try
      {
            Type type = Type.GetTypeFromProgID("StaadPro.OpenSTAAD");
            string clsid = type.GUID.ToString();
            string lookUpCandidateName = String.Format("!{0}{1}{2}", "{", clsid, "}").ToUpper();

            List<RunningObjectInfo> runningObjects = GetRunningObjectList();
            foreach (RunningObjectInfo item in runningObjects)
            {
                  string candidateName = item.Name.ToUpper();
                  if (candidateName.StartsWith(lookUpCandidateName))
                  {
                        count++;
                  }
            }
      }
      catch (COMException)
      {
            throw;
      }
      return count;
}
```

Any STAAD.Pro instance running?

```
public static bool AnySTAADInstanceRunning()
{
      try
      {
            Type type = Type.GetTypeFromProgID("StaadPro.OpenSTAAD");
            string clsid = type.GUID.ToString();
            string lookUpCandidateName = String.Format("!{0}{1}{2}", "{", clsid, "}").ToUpper();

            List<RunningObjectInfo> runningObjects = GetRunningObjectList();
            foreach (RunningObjectInfo item in runningObjects)
            {
                  string candidateName = item.Name.ToUpper();
                  if (candidateName.StartsWith(lookUpCandidateName))
                  {
                        return true;
```

```
                }
            }
        }
        catch (COMException)
        {
            throw;
        }
        return false;
}
```

Are multiple STAAD.Pro instances running?

```
public static bool AreMultipleSTAADInstances()
{
        int count = 0;
        try
        {
            Type type = Type.GetTypeFromProgID("StaadPro.OpenSTAAD");
            string clsid = type.GUID.ToString();
            string lookUpCandidateName = String.Format("!{0}{1}{2}", "{", clsid, "}").ToUpper();

            List<RunningObjectInfo> runningObjects = GetRunningObjectList();
            foreach (RunningObjectInfo item in runningObjects)
            {
                string candidateName = item.Name.ToUpper();
                if (candidateName.StartsWith(lookUpCandidateName))
                {
                    count++;
                }
            }
        }
        catch (COMException)
        {
            throw;
        }
        return (count > 1);
}
```

Helper Methods

```
[DllImport("ole32.dll")]
private static extern int GetRunningObjectTable(int reserved, out
System.Runtime.InteropServices.ComTypes.IRunningObjectTable prot);

[DllImport("ole32.dll")]
```

```
private static extern int CreateBindCtx(int reserved, out System.Runtime.InteropServices.ComTypes.IBindCtx
ppbc);

private class RunningObjectInfo
{
      public string Name
      {
            get;
            set;
      }

      public object Value
      {
            get;
            set;
      }
}

private static List<RunningObjectInfo> GetRunningObjectList()
{
      try
      {
            List<RunningObjectInfo> result = new List<RunningObjectInfo>();

            IntPtr numFetched = new IntPtr();
            System.Runtime.InteropServices.ComTypes.IRunningObjectTable runningObjectTable;
            System.Runtime.InteropServices.ComTypes.IEnumMoniker monikerEnumerator;
            System.Runtime.InteropServices.ComTypes.IMoniker[] monikers = new
System.Runtime.InteropServices.ComTypes.IMoniker[1];

            GetRunningObjectTable(0, out runningObjectTable);

            runningObjectTable.EnumRunning(out monikerEnumerator);
            monikerEnumerator.Reset();

            while (monikerEnumerator.Next(1, monikers, numFetched) == 0)
            {
                  System.Runtime.InteropServices.ComTypes.IBindCtx ctx;
                  CreateBindCtx(0, out ctx);

                  string runningObjectName;
                  monikers[0].GetDisplayName(ctx, null, out runningObjectName);
```

```
                object runningObjectVal;
                runningObjectTable.GetObject(monikers[0], out runningObjectVal);

                RunningObjectInfo objInfo = new RunningObjectInfo
                {
                      Name = runningObjectName,
                      Value = runningObjectVal
                };

                result.Add(objInfo);
          }

          return result;
     }
     catch (Exception)
     {
          throw;
     }
}
```

Accessing Individual Objects

Example: Accessing Geometry Object

```
OSGeometryUI _osGeom = _os.Geometry as OSGeometryUI;
```

Accessing Methods of the Objects

Example: Accesing 'GetNodeList' method of Geometry object

```
public int GetNodeList(ref int[] nodeList)
{
     nodeList = null;

     try
     {
          if (_osGeom != null)
          {
                int count = GetNodeCount();
                if (count > 0)
                {
                      nodeList = new int[count];
                      Object o1 = nodeList as object;
                      _osGeom.GetNodeList(ref o1);
```

```
                    nodeList = o1 as int[];
                }
                return count;
            }
        }
        catch (COMException ex)
        {
            return 0;
        }
        return 0;
}
```