Listings Referenced in

Getting Started with Geoprocessing and ArcObjects in .NET

By Chad D. Cooper, GISP, Southwestern Energy Company

Listings are provided for reference when reading the article. Use the .cs files also contained in this archive (gpandnet.zip) when working the exercise.

```
1  using System;
2     public class Hello
3     {
4        public static void Main()
5        {
6           Console.WriteLine("Hello world!");
7        }
8     }
```

Listing 1. Your first C# program, Hello world

```
1  C:\WINDOWS\system32>csc /out:c:\testing\csharp\HelloWorld.exe
   c:\testing\csharp\HelloWorld.cs
```

Listing 2. Compiling HelloWorld.cs at the command line

```
1  C:\WINDOWS>cd system32
2  C:\WINDOWS\system32>cd c:\testing\csharp
3  C:\Testing\csharp>HelloWorld
4  Hello world!
5  C:\Testing\csharp>
```

Listing 3. Running your first .NET program at the command prompt

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using ESRI.ArcGIS.Geoprocessor;
5  using ESRI.ArcGIS.Geoprocessing;
6
7  namespace GeoprocessingExample1
8  {
9     class Program
10    {
11       static void Main(string[] args)
12       {
13          // Create the geoprocessor object
14          Geoprocessor gp = new Geoprocessor();
15          // Set the workspace
16          gp.SetEnvironmentValue("workspace", @"C:\Data");
17          // Get enumeration of workspace objects
```

```
18            //   in C:\(geodatabases)
19            IGpEnumList gdbs = gp.ListWorkspaces("", "");
20            string gdb = gdbs.Next();
21            // Enumerate through the geodatabase, print out
22            //   featureclass names to standard output
23            while (gdb != "")
24            {
25               Console.WriteLine(gdb.ToString());
26               gp.SetEnvironmentValue("workspace", gdb);
27               IGpEnumList fcs = gp.ListFeatureClasses("", "", "");
28               string fc = fcs.Next();
29               while (fc != "")
30               {
31                  Console.WriteLine(fc.ToString());
32                  fc = fcs.Next();
33               }
34               gdb = gdbs.Next();
35            }
36         }
37      }
38   }
```

Listing 4. Enumerating through a geodatabase and printing the featureclass names to standard output

```
1  C:\VS\ArcUserArticle\Geoprocessing\GpExample\GpExample\bin\Debug>GpExample.exe
2  C:\Data\ArkansasData.gdb
3  ArkansasCityLimits
4  ArkansasCounties
5  NhdMediumResLinesBeaver
6  NhdMediumResLinesIllinois
7
8  C:\VS\ArcUserArticle\Geoprocessing\GpExample\GpExample\bin\Debug>
```

Listing 5. Results of running Listing 4 on our sample dataset

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using ESRI.ArcGIS.AnalysisTools;
5  using ESRI.ArcGIS.DataManagementTools;
6  using ESRI.ArcGIS.Geoprocessor;
7  using ESRI.ArcGIS.Geoprocessing;
8  using ESRI.ArcGIS.esriSystem;
9  using ESRI.ArcGIS.Geodatabase;
10  using ESRI.ArcGIS.DataSourcesFile;
11  using ESRI.ArcGIS.DataSourcesGDB;
12  using System.IO;
13
14  namespace GpExampleToolboxTools
15  {
```

```
class Program
{
   static void Main(string[] args)
   {
      Geoprocessor gp = new Geoprocessor();
      // Set the workspace
      gp.SetEnvironmentValue("workspace", @"C:\Data\ArkansasData.gdb");
      // Set up StreamWriter to log to
      StreamWriter lw = File.AppendText(@"C:\Data\Logfile_" +
            DateTime.Now.ToString("yyyy_MM_dd_hh_mm_ss_tt") + ".log");
      // Setup array with featureclasses we need to convert to layers
      string[] fcs = new string[2] { "NhdMediumResLinesBeaver",
                                     "NhdMediumResLinesIllinois" };
      // Iterate through featureclasses, process each one
      foreach (string fc in fcs)
      {
         // Make feature layers for input into SelectByAttribute tool
         MakeFeatureLayer mfl = new MakeFeatureLayer();
         // Set input and output parameters
         mfl.in_features = fc;
         mfl.out_layer = fc + "_Lyr";
         // Run the tool
         RunTool(gp, mfl, null);

         // Run SelectByAttribute to find perennial streams
         SelectLayerByAttribute sba = new SelectLayerByAttribute();
         // Set input and output parameters
         sba.in_layer_or_view = fc + "_Lyr";
         sba.selection_type = "NEW_SELECTION";
         // Note that in where_clause we must escape double quotes around
         //  field name with whacks (backward slashes)
         sba.where_clause = "\"FCODE_DESC\" = 'Stream/River: " +
                  "Hydrographic Category = Perennial'";
         RunTool(gp, sba, null);

         // Run SelectLayerByLocation tool to find perennial stream
         //  segments that intersect city limits
         SelectLayerByLocation sbl = new SelectLayerByLocation();
         sbl.in_layer = fc + "_Lyr";
         // Do a selection on the subset that we already selected above
         //  by attribute (select on the
         //  already existing selection set)
         sbl.selection_type = "SUBSET_SELECTION";
         sbl.select_features = "ArkansasCityLimits";
         sbl.overlap_type = "INTERSECT";
         RunTool(gp, sbl, null);

         // Run Clip tool - note that we must fully qualify the
         //  toolbox reference as there is also a
         //  Clip tool in the Data Management toolbox
         ESRI.ArcGIS.AnalysisTools.Clip clip =
                  new ESRI.ArcGIS.AnalysisTools.Clip();
         clip.in_features = fc + "_Lyr";
         // Clip the the city limits layer
         clip.clip_features = "ArkansasCityLimits";
```

```
71              // Push out a new featureclass to our file geodatabase
72              clip.out_feature_class = fc + "_WithinCityLimits";
73              RunTool(gp, clip, null);
74
75              // Get a sum of the stream lengths
76              double sum = SumStreamLengths(fc + "_WithinCityLimits");
77              string msg = "Stream length sum for featureclass \"" + fc +
78                      "_WithinCityLimits\": " + sum.ToString() + " KM";
79              Console.WriteLine(msg);
80              Log(msg, lw);
81          }
82      }
83
84      // Method to iterate through a featureclass and sum the stream lengths
85      private static double SumStreamLengths(string inputFc)
86      {
87          // Get a featureclass object
88          IFeatureClass fc = OpenFc(@"C:\Data\ArkansasData.gdb", inputFc);
89          // Set up a queryfilter on the LENGTHKM field
90          IQueryFilter2 qf = new QueryFilterClass();
91          qf.SubFields = "LENGTHKM";
92          int fieldPositionLenghtKm = fc.FindField("LENGTHKM");
93          // Create a featurecursor to loop through
94          IFeatureCursor featCur = fc.Search(qf, false);
95          IDataset ds = (IDataset)fc;
96          IFeature feature = null;
97          double k = 0;
98          // Start looping through the features
99          while ((feature = featCur.NextFeature()) != null)
100         {
101             // Get the value of the LENGTHKM field for each feature
102             double lenKm = Convert.ToDouble(
103                     feature.get_Value(fieldPositionLenghtKm));
104             // Sum the values with every iteration
105             k = k + lenKm;
106         }
107         // Return the final summation of the stream lengths
108         return k;
109     }
110
111     // Method to open up a featureclass and return it as an object
112     private static IFeatureClass OpenFc(string inputWs, string inputFc)
113     {
114         // Set up a workspace factory for a file geodatabase
115         IWorkspaceFactory2 wsf2 = (IWorkspaceFactory2)new
116             ESRI.ArcGIS.DataSourcesGDB.FileGDBWorkspaceFactoryClass();
117         // Create a workspace in our file geodatabase and open it
118         IWorkspace ws = wsf2.OpenFromFile(inputWs, 0);
119         IFeatureWorkspace fws = (IFeatureWorkspace)ws;
120         // Open up the featureclass, stuff into a IFeatureClass object
121         IFeatureClass fc = fws.OpenFeatureClass(inputFc);
122         // Return the featureclass object
123         return fc;
124     }
125
```

```
126         // Method to run our geoprocessing tools and handle
127         //  any errors that occur
128         private static void RunTool(Geoprocessor geoprocessor,
129                                     IGPProcess process, ITrackCancel TC)
130         {
131            // Set the overwrite output option to true; avoid issues
132            //  with output datasets that may already exist
133            geoprocessor.OverwriteOutput = true;
134            // Execute the tool
135            try
136            {
137               geoprocessor.Execute(process, null);
138               // Print out processing messages to stdout
139               ReturnMessages(geoprocessor);
140            }
141            catch (Exception err)
142            {
143               Console.WriteLine(err.Message);
144               ReturnMessages(geoprocessor);
145            }
146         }
147
148         // Method for returning the tool messages.
149         private static void ReturnMessages(Geoprocessor gp)
150         {
151            if (gp.MessageCount > 0)
152            {
153               for (int Count = 0; Count <= gp.MessageCount - 1; Count++)
154               {
155                  Console.WriteLine(gp.GetMessage(Count));
156               }
157            }
158         }
159
160         private static void Log(String logMessage, TextWriter w)
161         // Write messages to log file
162         {
163            w.WriteLine(logMessage);
164            w.Flush();
165         }
166      }
167   }
```

Listing 6. Using geoprocessing tools on the ArkansasData file geodatabase

```
1  Stream length sum for featureclass "NhdMediumResLinesBeaver_WithinCityLimits": 102.923 KM
2  Stream length sum for featureclass "NhdMediumResLinesIllinois_WithinCityLimits": 143.515 KM
```

Listing 7. Log file results from Listing 6

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using ESRI.ArcGIS.Geodatabase;
5  using ESRI.ArcGIS.esriSystem;
6  using ESRI.ArcGIS.DataSourcesGDB;
7
8  namespace ConnectToSde
9  {
10     class Program
11     {
12        public static List<string> FetchSdeFcList(string inWs, string dbType)
13        {
14           List<string> fcsList = new List<string>();
15           IWorkspaceFactory2 wsf = new SdeWorkspaceFactoryClass();
16           IPropertySet ps = new PropertySetClass();
17           if (dbType == "enterprise")
18           {
19              // Enterprise SDE SQL Server 2005
20              // Server name
21              ps.SetProperty("SERVER", "server");
22              // Server instance
23              ps.SetProperty("INSTANCE", "sde:sqlserver:server");
24              // Authentication mode: database
25              ps.SetProperty("AUTHENTICATION_MODE", "DBMS");
26              // Since using database authentication, we must pass in
27              //   credentials
28              ps.SetProperty("USER", "user");
29              ps.SetProperty("PASSWORD", "pass");
30              // Database name
31              ps.SetProperty("DATABASE", "sde");
32              // Database version
33              ps.SetProperty("VERSION", "sde.DEFAULT");
34           }
35           else
36           {
37              // Workgroup SDE SQL Server Express 2008
38              ps.SetProperty("SERVER", "server_sqlexpress");
39              ps.SetProperty("INSTANCE",
40                       "sde:sqlserver:server\\sqlexpress");
41              // Here we use operating system authentication (OSA), so we
42              //   don't need to supply credentials (username and password)
43              ps.SetProperty("AUTHENTICATION_MODE", "OSA");
44              ps.SetProperty("DATABASE", "db");
45              ps.SetProperty("VERSION", "dbo.DEFAULT");
46           }
47           // Open up the workspace connection
48           IWorkspace ws = wsf.Open(ps, 0);
49           IFeatureWorkspace fws = (IFeatureWorkspace)wsf.Open(ps, 0);
50           // Open the featuredataset we specify as an input argument
51           IFeatureClassContainer fcc =
52                    (IFeatureClassContainer)fws.OpenFeatureDataset(inWs);
53           // Get an enumeration of the featureclasses in the featuredataset
54           IEnumFeatureClass efc = fcc.Classes;
55           IFeatureClass fc = efc.Next();
```

```
56            // Run through the enumeration
57            while (fc != null)
58            {
59               Console.WriteLine(fc.AliasName);
60               fc = efc.Next();
61            }
62            return fcsList;
63         }
64
65
66         static void Main(string[] args)
67         {
68            // Fetch a license and initialize it
69            IAoInitialize aoInit = new AoInitializeClass();
70            aoInit.Initialize(
71                  esriLicenseProductCode.esriLicenseProductCodeArcInfo);
72            // Call our method, passing in arguments: 1) featuredataset name
73            //   2) SDE type - workgroup or enterprise
74            FetchSdeFcList("db.DBO.feature dataset", "workgroup");
75         }
76      }
77   }
```

Listing 8. Connecting to SDE and listing featureclasses in a feature dataset

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using ESRI.ArcGIS.Server;
5
6  namespace AgsGetServices
7  {
8     class Program
9     {
10       static void Main(string[] args)
11       {
12          GISServerConnection conn = null;
13          IServerObjectAdmin4 soa = null;
14
15          conn = new GISServerConnection();
16          conn.Connect("server");
17
18          soa = (IServerObjectAdmin4)conn.ServerObjectAdmin;
19          IEnumServerObjectConfiguration enumConfig =
20                               soa.GetConfigurations();
21          IServerObjectConfigurationStatus configStatus = null;
22          IServerObjectConfiguration config = enumConfig.Next();
23          config.WaitTimeout = 600;
24          while (config != null)
25          {
26             configStatus = soa.GetConfigurationStatus(
27                            config.Name, config.TypeName);
```

```
28                 Console.WriteLine(String.Concat("Service ",
29                                         config.Name,
30                                         " is " ,
31                                         configStatus.Status));
32                 config = enumConfig.Next();
33               }
34             }
35           }
36         }
```

Listing 9. Interacting with ArcGIS Server services.

```
1   C:\VS\ArcUserArticle\AgsGetServices\AgsGetServices\bin\Debug>AgsGetServices.exe
2   Service Charlotte is esriCSStarted
3   Service CharlotteLocator is esriCSStarted
4   Service Texas_Aerials_All_UTMS is esriCSStarted
5   Service TexasGeneral/TexasGeneral is esriCSStopped
6   Service TexasGeneral/TexasWells is esriCSStarted
7   Service TexasGeneral/TexasGeneralRaster is esriCSStopped
8   Service TexasHSE/TexasHSEBasemap is esriCSStarted
9   Service TexasHSE/HSE is esriCSStarted
10  Service TexasGeology/TexasGeology is esriCSStarted
11
12  C:\VS\ArcUserArticle\AgsGetServices\AgsGetServices\bin\Debug>
```

Listing 10. Results of running Listing 9 at the command line