

Steven D. Straka
Sanborn

Building Geodatabases for City and County Clients

In Sanborn's current contracts with many of the Cities and Counties in which they perform data conversion work for, coverage or shapefile deliveries are required. These Cities and Counties also, many times desire an enterprise or personal Geodatabase delivery at the end of the data conversion cycle. This paper will address how Sanborn uses ArcObjects together with their conversion software to produce these Geodatabases. Topics that will be addressed are vector translations along with annotation translations and the issues encountered with both. How Sanborn addressed topology issues and how TrueType fonts were handled will be two of the detailed topics addressed in this paper. Also addressed will be how the raster data was handled in ArcSDE and the problems encountered with the loading of TIFF images.

BACKGROUND

Sanborn clients are requiring data deliveries in many different formats ranging from ASCII files, coverages, and shapefiles up to personal and enterprise Geodatabases. To address this need Sanborn has produced different translators from our conversion platform into the desired target format. The latest in this family of translators is the Geodatabase translator.

Sanborn has been able to delivery Geodatabase by either going into a coverage or shapefile and then perform an additional translation into the Geodatabase. It was the desire of Sanborn management to eliminate the second translation and just be able to translate directly into the Geodatabase.

Also Sanborn came across issues that the shapefile and coverages could not support. One thing that was important to many Sanborn clients was the delivery of true circular arcs. Sanborn's software supported circular arcs, but Sanborn had no way to translate them to a format that ESRI products would support before the Geodatabase.

Annotation was the other feature that Sanborn felt must be supported from the translator. Shapefiles of course do not support annotation and translating to a coverage file format just to get annotation into a Geodatabase was highly inefficient.

Sanborn made the decision early in 2004 that a translator directly into a Geodatabase must be added to its family of software translators for the future.

DESIGN CONSIDERATION

Sanborn wanted this translator to be more than just a shapefile or coverage translator replacement. It was important that the new translator support features not supported in the other translators. The two most important features were the support for circular arcs and the ability to handle annotation during translation.

Sanborn needed to pick a language to develop the translator in. The shapefile translator that had been developed in ArcObjects a couple of years earlier was written in VB. The majority of the examples for developers were also in VB. The major factor in the final decision was that Sanborn felt that they could gain more speed with the final product if they developed it in C++. This proved to be a good decision even though the COM/C++ skills were not as strong as the COM/VB skills at the start of the development process.

Sanborn wanted personal and multi-user Geodatabase translations supported. Sanborn had various clients that currently preferred the different Geodatabases and wanted to be able to translate directly to either type of Geodatabase.

Sanborn would start with just a one way translator from its software into a Geodatabase. It was assumed that a translator back into Sanborn's software could be written fairly quickly once the experience was gained from coding the 1st translator. The other factor was that the early documentation pointed to a XML exchange format in ArcGIS 9.0 that might be a better solution for the 2nd translator, if not of both.

Domains and Subtypes needed to be able to transfer from Sanborn's software into the Geodatabase as appropriate.

Attribute support was must. Sanborn uses attributes for doing the majority of its processing and being able to transfer these attributes as well as associate Sanborn data types to Geodatabase data types was very important to the success of the translator.

The desire of management was for the translator to be able to work in both an interactive graphical mode and a batch mode, as Sanborn executes much of its work in both ways.

The ESRI geometry types that were being supported by the Sanborn's shapefile translator would be supported with the Geodatabase translator. These included point, multipoint, line, polyline, polygon. Polygons with and without "holes" or what ESRI calls inner and outer rings would need to be supported. Also multiple rings and multiple paths needed to be supported.

DEVELOPMENT / CODE EXAMPLES

After the initial decision to use C++ and COM to code the Geodatabase translator, trying to find relevant examples and coding small prototypes functions was the next step in making sure the Sanborn had a good grasp of coding ArcObjects with C++. This ended up to be easier than 1st estimated but still quite an up hill challenge. The ESRI Object Browser was a must for coding in C++ and it gave the IDL format that was needed to correctly code the calls to the COM methods. Note, ArcGIS 9.0 documentation has C++ Help and seems to have done a much better job of ArcObjects documentation for the C++/COM developer.

The handling of the ESRI geometry types that Sanborn had already coded in VB for the shapefile translator was very straight forward once the COM/C++ hand shaking has all figured out. It was mainly just switching workspace types and changing the code from VB to C++.

Sanborn wished to emulate the ArcGIS connection environment as close as possible and came up with the following two connection menus as part of its graphical Geodatabase conversion wizard. The 1st menu below shows connecting to a personal Geodatabase and the 2nd menu demonstrates connecting to a multi-user ArcSDE Geodatabase. It is assumed in both cases that the Geodatabase has already been created with ArcCatalog, Case Tool, or some other method.

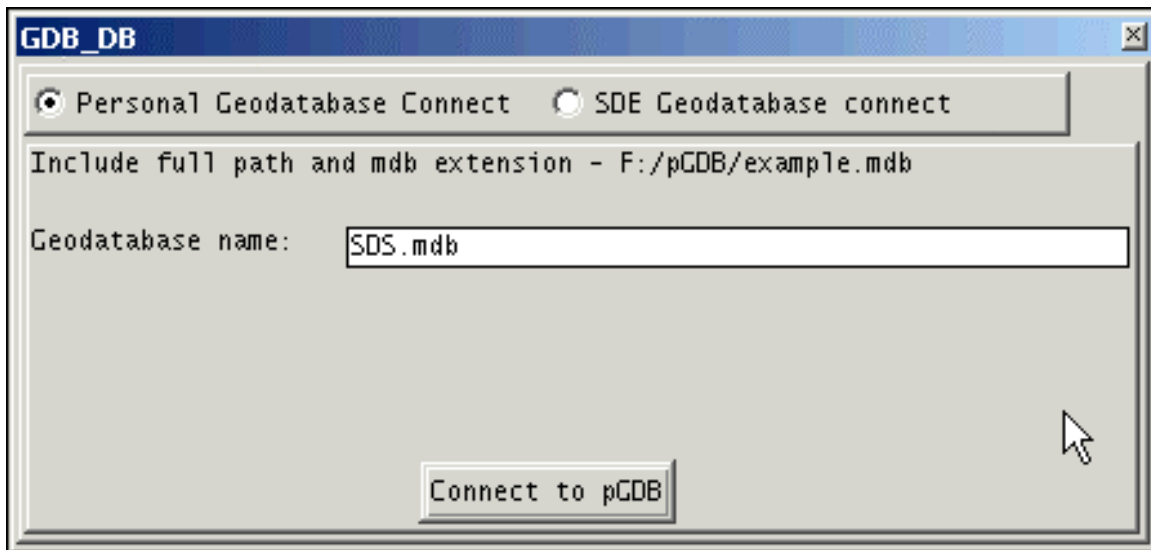


Figure 1. Personal Geodatabase connection

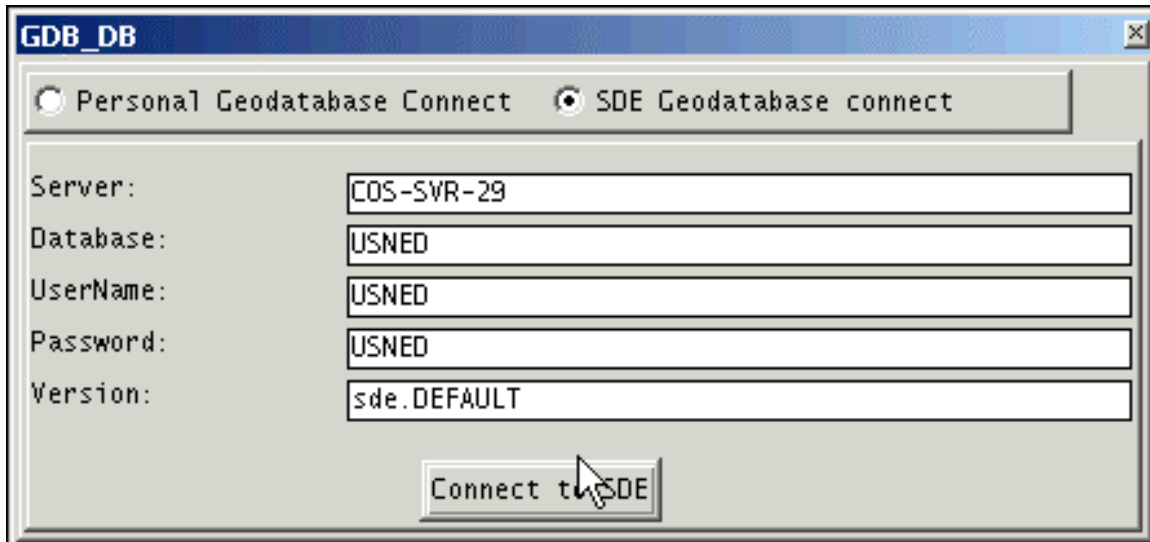


Figure 2. SDE Geodatabase connection

The need to handle circular arcs was a requirement from the start of the design stage. Sanborn's current software handles circular arcs and stores them internally as 3 point arcs. So the 1st step was to see if ArcObjects supported 3 point circular arcs. In checking the documentation it was discovered that ArcObjects supported the construction of circular arcs with various types of inputs, one being the 3 point arc. Below is a snippet of code for the circular arc processing using COM/C++ from Sanborn's Geodatabase translator code.

```

case arc_elem:          // *** Arc element
// *** Arc (or circle) geometry.
{
// *** Declare helper objects
CComPtr<IConstructCircularArc> pConstArc
hrx = pConstArc.CoCreateInstance(CLSID_CircularArc, NULL, CLSCTX_INPROC_SERVER);
CComPtr<ISegmentCollection> pSegColl;
hrx = pSegColl.CoCreateInstance(CLSID_Polyline, NULL, CLSCTX_INPROC_SERVER);

// *** Handle Z value if feature class supports it.
if(FeatureClassHasZ) {
    CComQIPtr<IZAware> pZA(pSegColl);
    pZA->put_ZAware(IAware);
}

// *** Get the 1st, last and middle points
CComPtr<IPoint> fromPnt;
hrx = fromPnt.CoCreateInstance(CLSID_Point, NULL, CLSCTX_INPROC_SERVER);
  
```

```

fromPnt->put_X(currentElementPointer->d.p[0]->x);
fromPnt->put_Y(currentElementPointer ->d.p[0]->y);

CComPtr<IPoint> middlePnt;
hrx = middlePnt.CoCreateInstance(CLSID_Point, NULL, CLSCTX_INPROC_SERVER);
middlePnt->put_X(currentElementPointer ->d.p[1]->x);
middlePnt->put_Y(currentElementPointer ->d.p[1]->y);

CComPtr<IPoint> toPnt;
hrx = toPnt.CoCreateInstance(CLSID_Point, NULL, CLSCTX_INPROC_SERVER);
toPnt->put_X(currentElementPointer ->d.p[2]->x);
toPnt->put_Y(currentElementPointer ->d.p[2]->y);

// *** Handle Z value if feature class supports it.
if(FeatureClassHasZ) {
    CComQIPtr<IZAware> pZAF(fromPnt);
    pZAF->put_ZAware(IZAware);
    fromPnt->put_Z(currentElementPointer ->d.p[0]->z);

    CComQIPtr<IZAware> pZAM(middlePnt);
    pZAM->put_ZAware(IZAware);
    middlePnt->put_Z(currentElementPointer ->d.p[0]->z);

    CComQIPtr<IZAware> pZAT(toPnt);
    pZAT->put_ZAware(IZAware);
    toPnt->put_Z(currentElementPointer ->d.p[0]->z);
}

VARIANT_BOOL UseCntPnt = VARIANT_TRUE;

// *** Construct the arc element using the 3 points
hr = pConstArc->ConstructThreePoints(fromPnt, middlePnt, toPnt, UseCntPnt);

CComQIPtr<ICurve> pCurve(pConstArc);
hrx.PointerMessage(pCurve, "Failed to create a curve from the constuction arc.");

CComQIPtr<ICircularArc> pCirArc(pCurve);
hrx.PointerMessage(pCirArc, "Failed to create a circular arc from the constuction arc.");

CComQIPtr<ISegment> pSegment(pCirArc);
hrx.PointerMessage(pSegment, "Failed to create a segment from the constuction arc.");

pSegColl->AddSegment(pSegment);

// *** Put the geometry in the feature.
CComQIPtr<IGeometry> geom(pSegColl);

// *** Add attributes to geodatabase
hr = PutFields(currentElementPointer, new_feature);

hr = new_feature->putref_Shape(geom);
CHECK_HRESULT_IN_PUT("Failed to set feature's arc path geometry.");
break;
}

```

Below is one of the many examples from the ArcGIS Developer Help. The IConstructCircularArc Interface has a wide variety of parameters for the constructions for circular arcs and is well documented. For the Sanborn translator we used the ConstructThreePoints Method from this interface. Below is a description of what that method does from the documentation.

Given a [From Point](#), a [Thru Point](#), and a [To Point](#), the unique [CircularArc](#) defined by those points is constructed. The From and To Points become the From and To Points of the CircularArc. The Thru Point is a point that lies somewhere on the CircularArc. For every three points, a single well-defined CircularArc can be created. (The only exception is if all three points are [colinear](#) and the Thru Point is not between the From and To Points.)

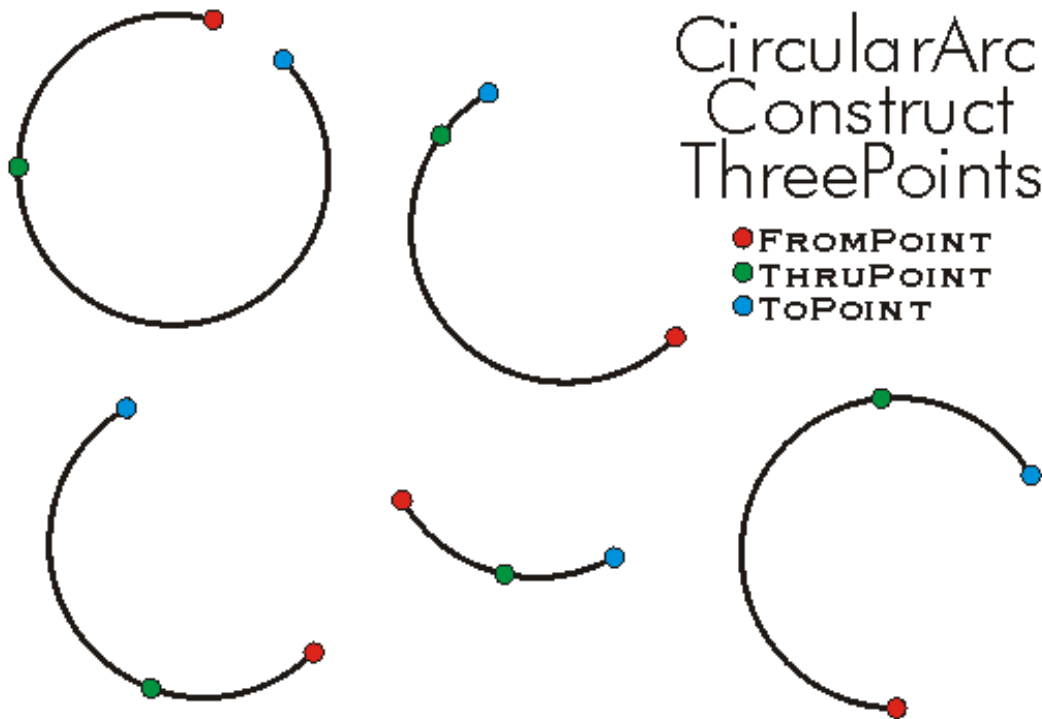


Figure 3. Circular Arc Example from ArcGIS Developer Help

The processing of the annotation was a little tricky just due to the fact that this was not addressed when Sanborn developed its shapefile translator a couple of years earlier. At the current release of the Geodatabase translator, annotation translations are working but Sanborn still has issues with matching fonts between their software annotation and the Geodatabase annotation. Also of issue, was the handling of reference scales and several other minor items. Below is a sample of how the scale and rotation was handled within the annotation code.

```
// *** Re-adjust text justification point to center lower
txt_rect tr;
pnt txt_just_pnt;
Get_text_position(just_cen_cen, currentElementPointer, currentElementPointer ->d.t->op, &tr);
Get_text_just_point(just_cen_low, &tr, &txt_just_pnt);

// *** Now turn the point into a geometry and add the shape to the feature.
CComPtr<IPoint> pOrgPnt;
hr = pOrgPnt.CoCreateInstance(CLSID_Point, NULL, CLSCTX_INPROC_SERVER);
hr = pOrgPnt->put_X(txt_just_pnt.x);
hr = pOrgPnt->put_Y(txt_just_pnt.y);
CComQIPtr<IGeometry> geom(pOrgPnt);
hrx.PointerMessage(geom, "Failed to create geometry for element.");

hr = anno_elem->put_Geometry(geom);
CHECK_HRESULT_IN_PUT("Failed to set feature's point geometry.");

// *** QI for 2D transformation.
CComQIPtr<ITransform2D> pTransform2D(text_elem);
hrx.PointerMessage(pTransform2D, "Failed to identify 2D Transform.");
hr = pTransform2D->Rotate(pOrgPnt, currentElementPointer ->d.t->rot.z);
hr = pTransform2D->Scale(pOrgPnt, currentElementPointer ->d.t->scl.x,
                        currentElementPointer ->d.t->scl.y);
```

CURRENT ENHANCEMENTS BEING IMPLEMENTED

Sanborn needs to be able to handle relationships and since a database supports this functionality very well, this is assumed to be a minor problem to add this into the current Geodatabase translator code. Sanborn's software supports a concept close to graphical relationships through what it calls group lists. Sanborn is now adding this functionality to the Geodatabase translator.

The other item that Sanborn wishes to support right away is Feature Linked Annotation. This has been something that Sanborn has been able to be emulated in its software for many years and Sanborn feels that the ability to translate it to the Geodatabase directly is very important.

FUTURE WORK

Additional improvements are required to fully justify the development of a completely new translator. Other items that must be more fully designed and considered for the Geodatabase translator are listed below.

- Topology – Sanborn software supports topology, but it may not make since to include it as part of this translator. This will have to be a future design decision.
- TrueType Fonts support from Sanborn's code
- Linear referencing
- Interface with Maplex
- Standard Data Models support
- Geodatabase XML support especially as a import into Sanborn's software and maybe also as a export replacement.
- ArcPAD translations
- Direct Connect ability
- ArcObjects 9.x conversion
- Handling of raster datasets
- Better wizard interfaces with pull down list and combo boxes for feature datasets and feature classes.
- Data Interoperability extension and how this could add to the translator
- .NET and C# investigation
- Performance testing
- Others

CONCLUSION

Sanborn has successfully implemented a one way translator from its software out to a Geodatabase. While testing has proven very successful, much work remains. The translator has already been used for production jobs and will see much more use in the future. The openness of ArcGIS and the completeness for ArcObjects made this task relatively painless.

ACKNOWLEDGEMENTS

Thank you to Sanborn for giving me time and materials to work on the translator and on the support of this paper.

REFERENCES

ESRI, XML Schema of the Geodatabase, February 2004.

ESRI, Exploring ArcObjects, 2001.

ESRI, System Design Strategies. March 2004.

ESRI, Shapefile Technical Description, July 1998

ESRI, Cartography: Capabilities and Trends, June 2004

Steven D. Straka
Senior Programmer Analyst
Sanborn
sstraka@sanborn.com