Improving ESRI ArcSDE Client Performance With Spatial Culling

DJ Bauch

Abstract

This paper will report on the examination of the impact on performance of ArcView map documents containing large numbers of ArcSDE layers achieved through spatial culling of the layers on the client. The work includes custom ArcObjects behavior that uses envelopes to minimize the round-trip traffic to the ArcSDE server by dynamically disabling layers with spatial extents that do not intersect the view.

Background

People's perception of space is very much like their perception of time. Time in a motion picture, for example, does not progress linearly. Instead, we may see a conversation and the actors in a room – watching their every movement and the expressions on their faces, listening to the nuances of their voices. A few frames further into the movie, and it's suddenly five years later, and we see the same characters, now five years older, getting on a train. What has happened in the intervening five years? In the telling of the story of the movie, it really isn't important.

So too, I am seated at my desk in my office for eight hours a day, concerned about the placement of the items on my desk, the height of my chair, the colors of the tiny pixels on my computer screen, and all the other little details that help to make me comfortable and productive. At the end of my shift, I get up from my chair, walk to the car, and drive home.

On the way home, I may pass a restaurant – maybe it was a Burger King, maybe a Wendy's. Perhaps there were customers inside, but I wouldn't know. Each building I pass is just another vague architectural feature on the landscape. Those restaurants I passed can best be lumped into the category of "places I didn't stop to eat." In my consciousness, the details of the features between my office and home are not important.

Upon arriving home, I may look for the television remote control, or find a comfortable chair and sit to read a book. Once again I become interested in the little details that help to make me comfortable – but am perhaps less concerned with remaining productive.

Analogously, we're not at all interested in the fine details the feature classes that don't intersect the area of the world we may be examining with ArcMap. As we move around from city to city, we're interested in the details first of one particular city, and then another, but not so much in what may fall in between.

Feature Data in Shapefiles and Legacy Formats

Since many ArcGIS users may have a background using older software, such as ArcView 3.2, they may be presumed to have some familiarity with ESRI shapefiles, and also potentially a great deal of legacy data in this format. Even in the current software, some extensions, like the Military Analyst, produce shapefiles. Defense users also have access to a great deal of vector data available in Vector Product Format (VPF). To an ArcMap or ArcCatalog user, VPF files behave in a fashion similar to shapefiles.

Unless the user makes an effort to correct the situation, these shapefiles typically don't include a spatial reference. As a consequence, ArcGIS assumes a default spatial reference with an unspecified datum and with bounding coordinates as follows:

X/Y Domain		
	Min X:	-10,000.0
	Min Y:	-10,000.0
	Max X:	11,474.836450
	Max Y:	11,474.836450
	Scale:	100,000.0
M Domain		
	Min:	0.0
	Max:	21474.836450
	Scale:	100,000.0
Z Domain		
	Min:	0.0
	Max:	21474.836450
	Scale:	100,000.0

Although these X and Y bounds may seem odd, they do produce the effect of yielding the same precision in the x, y, z, and m dimensions. This default spatial reference also allows features to be placed anywhere in the world. For display purposes, ArcMap assumes the NAD27 datum whenever it encounters spatial data with an unspecified datum. The outcome of uploading a VPF file to ArcSDE is the same, even though all VPF data implicitly uses the WGS84 datum.

If a shapefile or VPF file is loaded into the geodatabase without setting a spatial reference system, a default spatial reference is assigned. Standalone feature classes are assigned a spatial reference sized according to the extent of the features in the shapefile at the time the data is loaded. Feature classes created within a feature dataset are assigned the spatial reference of the feature dataset. Any features that extend beyond the bounds of the feature dataset's spatial reference will be ignored. To an ArcMap user, the feature class may be used very much like the shapefile or VPF file, albeit usually with a perceptible performance improvement. In the case of VPF data, there is an additional benefit accrued by converting the data to an ArcSDE feature class. The feature class becomes editable. ArcInfo does not include the ability to edit VPF files as it does shapefiles.

Organization of Feature Data in ArcSDE

Vector features may be loaded into ArcSDE either as stand-alone feature classes or as members of a feature dataset. Each feature class or feature dataset has associated with it a spatial reference. The spatial reference specifies a coordinate system and bounding box, which implicitly specifies a precision for the data within the feature class. There's a tradeoff between the size of the bounding box and the precision achievable for specifying the positions of the features.

Feature datasets are collections of feature classes that share a common spatial reference system. Because of the perceptual characteristics described in this paper's background section, users should typically organize their data into feature datasets.

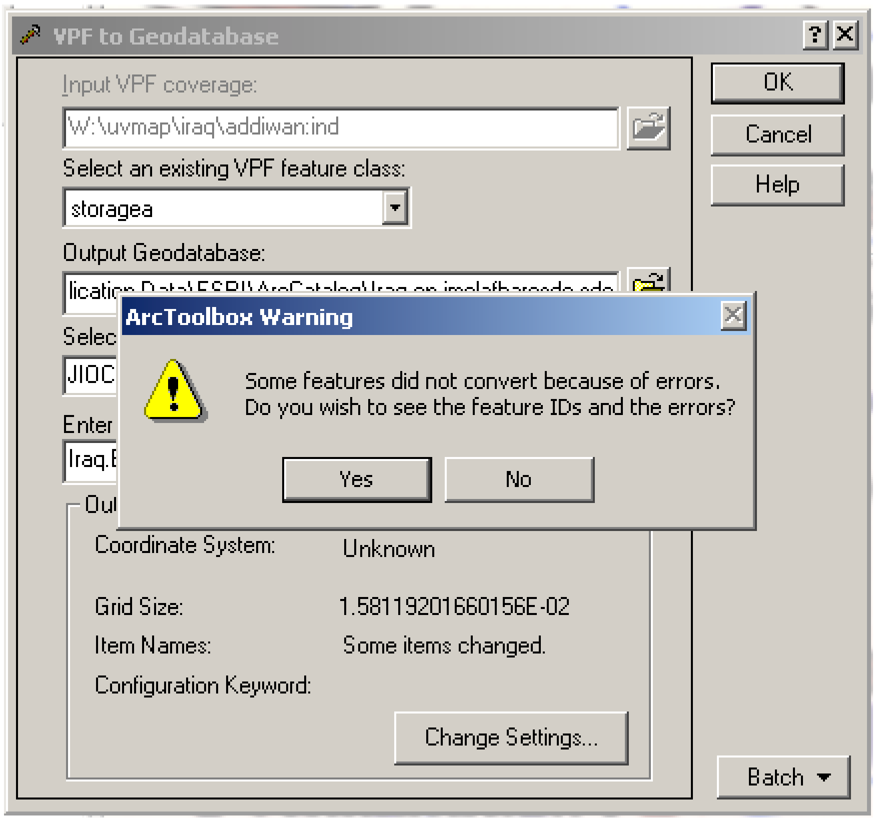
It may be tempting to create these feature datasets for maximum flexibility by associating them with a spatial reference that spans the union of all of the feature classes that may be incorporated into a map document. This temptation should be resisted.

There are also several reasons that a user would create tighter spatial references for the various regions of interest. A couple of these reasons can be illustrated by observing what happens if a user attempts to load feature data for different regions from legacy data sources into a common spatial reference, such as the default worldwide spatial reference system.

If there are many feature classes incorporated into a map, it is not uncommon – in the course of panning and zooming through the map – for some of the features to be in areas of the map not currently being viewed. It seems reasonable to completely ignore for rendering those feature datasets that include those features that are not being viewed.

When creating feature classes from VPF legacy data using the default spatial reference, some anomalous situations arise. These anomalies may form the foundation of an argument for being more careful about specifying spatial references.

When creating polygon feature classes within a feature dataset with a spatial reference that has an excessively large envelope, sometimes the upload will terminate with an ArcToolbox Warning as shown in Figure 1. When this warning appears, a feature class has been created and some, but not all, of the features from the shapefile or VPF file have been converted. The problem may be avoided by creating the feature dataset with a tighter envelope (that is, with greater spatial precision).



VPF to Geodatabase

Input VPF coverage:
W:\uvmap\iraq\addiwan:ind

Select an existing VPF feature class:
storagea

Output Geodatabase:
lication Data\ESRI\ArcCatalog\Iraq on jmolafbarcode.sde

ArcToolbox Warning

Some features did not convert because of errors.
Do you wish to see the feature IDs and the errors?

Coordinate System: Unknown

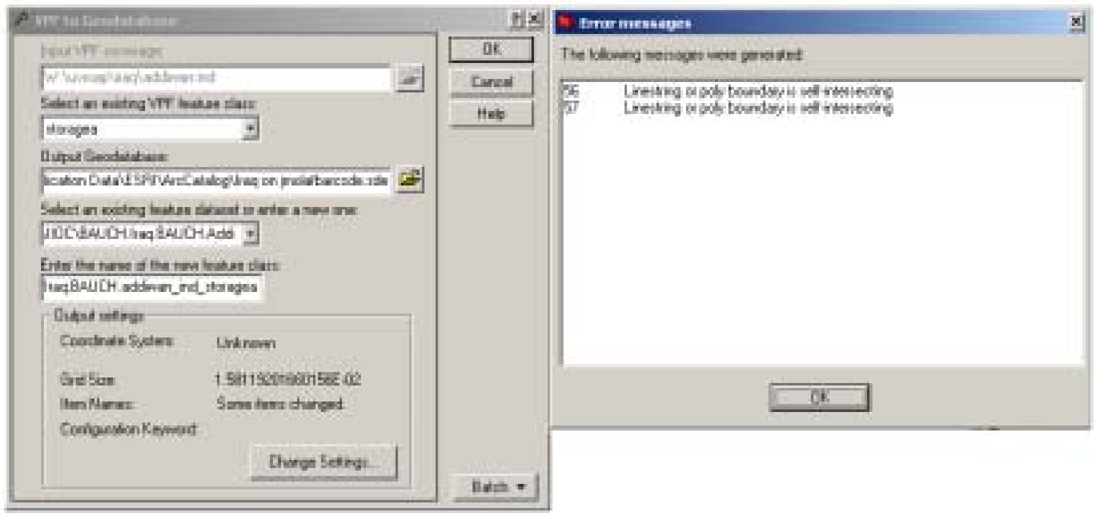
Grid Size: 1.58119201660156E-02

Item Names: Some items changed.

Configuration Keyword:

Figure 1 Some features may not convert

Choosing to see the feature IDs and the errors produces another dialog. This dialog, shown in Figure 2, sheds some light on the cause of the problem. The problem occurs because of the loss of precision associated with reprojecting the features into an insufficiently precise spatial reference.



VPF to Geodatabase

Input VPF coverage:
W:\uvmap\iraq\addiwan:ind

Select an existing VPF feature class:
storagea

Output Geodatabase:
lication Data\ESRI\ArcCatalog\Iraq on jmolafbarcode.sde

Select an existing feature dataset or enter a new one:
JIOC\BAUCH.Iraq.BAUCH.Addi

Enter the name of the new feature class:
IraqBAUCH.addiwan_ind_storagea

Output settings

Coordinate System: Unknown

Grid Size: 1.58119201660156E-02

Item Names: Some items changed.

Configuration Keyword:

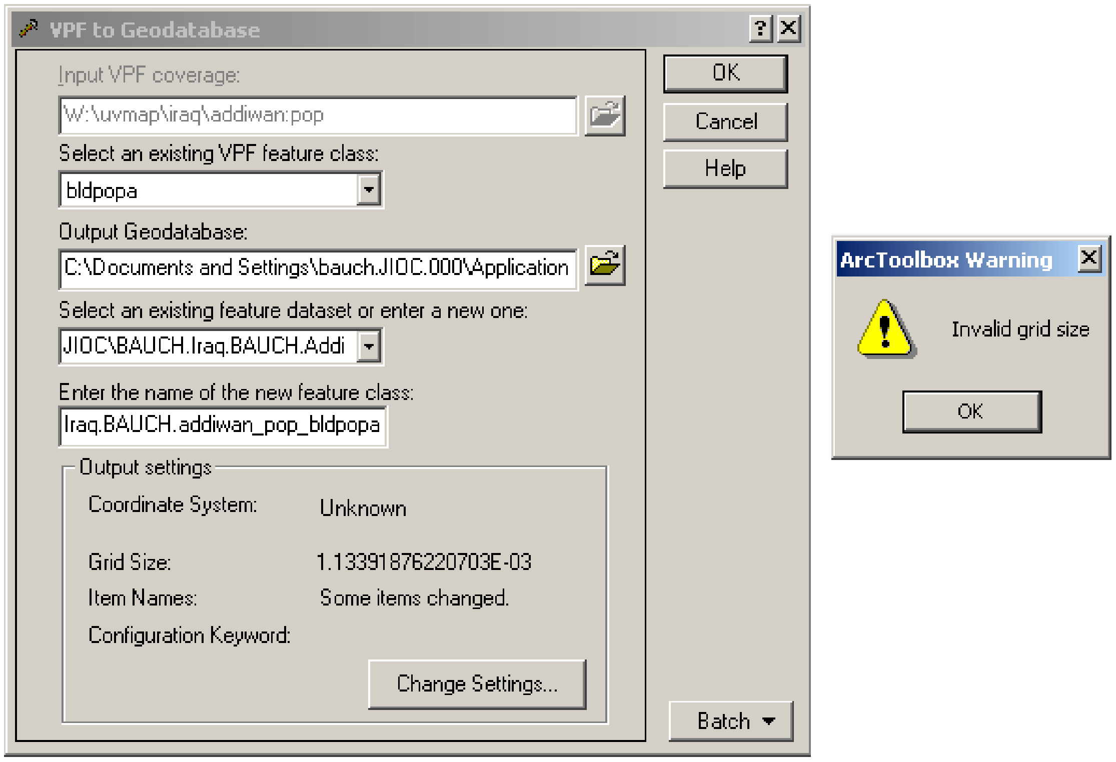
Error messages

The following messages were generated:

```
56        Linestring or poly boundary is self-intersecting
57        Linestring or poly boundary is self-intersecting
```

Figure 2 Self-intersecting polygons

Another error that may occur is that ArcGIS may detect an invalid grid size when attempting to create a feature class. In this case, the feature class will not be created, and no features will be converted.


VPF to Geodatabase

Input VPF coverage: W:\uvmap\iraq\addiwan:pop

Select an existing VPF feature class: bldpopa

Output Geodatabase: C:\Documents and Settings\bauch.JIOC.000\Application

Select an existing feature dataset or enter a new one: JIOC\BAUCH.Iraq.BAUCH.Addi

Enter the name of the new feature class: Iraq.BAUCH.addiwan_pop_bldpopa

Output settings

Coordinate System: Unknown

Grid Size: 1.13391876220703E-03

Item Names: Some items changed.

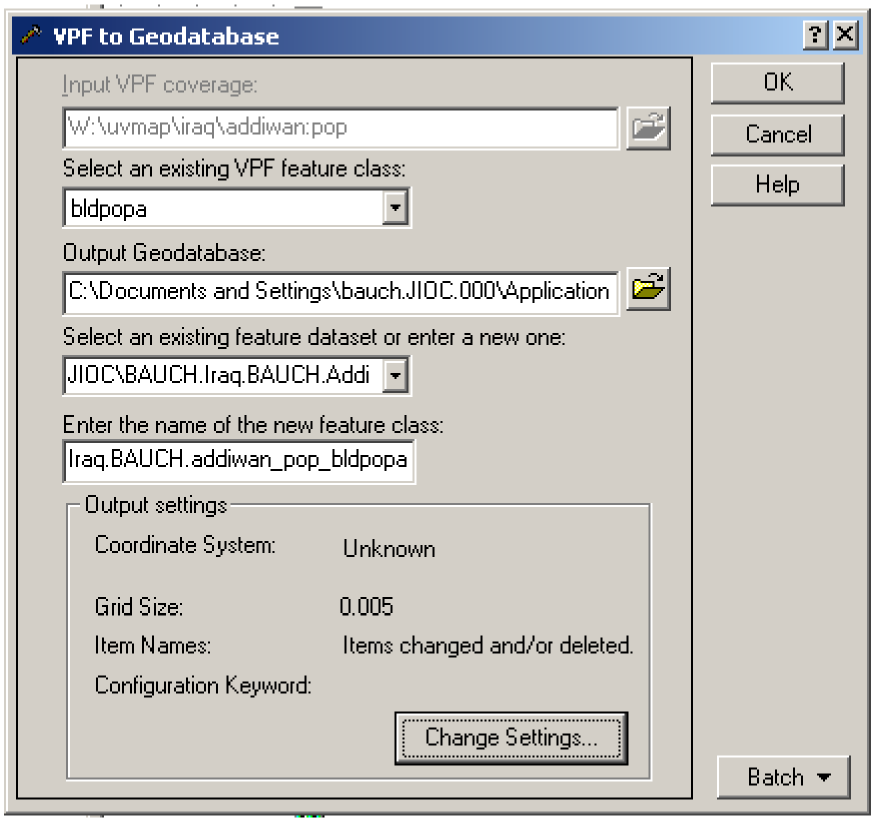
Configuration Keyword:

ArcToolbox Warning

Invalid grid size

Figure 3 Invalid grid size

This error may be corrected by adjusting the grid size, as illustrated in Figure 4. Although increasing the grid size solves the problem, it is not the best approach. Decreasing the size of the spatial reference envelope, and thereby increasing the achieved precision also solves the problem.


VPF to Geodatabase

Input VPF coverage: W:\uvmap\iraq\addiwan:pop

Select an existing VPF feature class: bldpopa

Output Geodatabase: C:\Documents and Settings\bauch.JIOC.000\Application

Select an existing feature dataset or enter a new one: JIOC\BAUCH.Iraq.BAUCH.Addi

Enter the name of the new feature class: Iraq.BAUCH.addiwan_pop_bldpopa

Output settings

Coordinate System: Unknown

Grid Size: 0.005

Item Names: Items changed and/or deleted.

Configuration Keyword:

Figure 4 Changing the grid size

Whether or not the user specifies a spatial reference when loading data into ArcSDE, the feature class will have a spatial reference associated with it. In ArcSDE, the characteristics of the spatial reference are not stored in the same format as seen in the desktop dialogs. The format of a spatial reference record in ArcSDE is shown in Table 1.

Column	Type	Constraints
srid	int	Not Null
description	varchar(64)	Null
auth_name	varchar(255)	Null
auth_srid	int	Null
falsex	float	Not Null
falsey	float	Not Null
xyunits	float	Not Null
falsez	float	Not Null
zunits	float	Not Null
falsem	float	Not Null
munits	float	Not Null
srtext	varchar(1024)	Not Null

Table 1 ArcSDE spatial references table structure

The minimum x and y coordinates have become falsex and falsey. The maximum x and y values are not stored. Instead they are implicitly derivable from the xyunits. The maximum x value is falsex + .5 / xyunits and the maximum y value is falsey + .5 / xyunits. The same thing is done with z and m. So, what's stored in the spatial reference is actually the reciprocal of xyunits. Thus, larger values for xyunits correspond to greater precision for the coordinates and to smaller envelopes for the feature classes.

The envelopes of the individual feature classes are stored in the ArcSDE layers table. The envelopes of the feature datasets are not stored, but can be no larger than the envelopes of the spatial references associated with the feature datasets.

Column	Type	Constraints
layer_id	int	Not Null
description	varchar(65)	Null
database_name	varchar(32)	Not Null
table_name	varchar(32)	Not Null
owner	varcar(32)	Not Null
spatial_column	varchar(32)	Not Null
eflags	int	Not Null
layer_mask	int	Not Null
gsize1	float	Not Null
gsize2	float	Not Null
gsize3	float	Not Null
minx	float	Not Null
miny	float	Not Null
maxx	float	Not Null
maxy	float	Not Null
cdate	int	Not Null
layer_config	varchar(32)	Null
optimal_array_size	int	Null
stats_date	int	Null
minimum_id	int	Null
srid	int	Not Null
base_layer_id	int	Null

Table 2 ArcSDE layers table structure

Case Study

Hardware / Software Environment

Server: Microsoft SQL Server 2000 with ArcSDE 8.3, Windows 2000 Server, 4 GB RAM, four 2.8 GHz Xeon processors.

Client: ArcGIS 8.3, Windows 2000 Professional, 1 GB RAM, two 1.7 GHz Xeon processors.

Features

Urban Vector Map (UVMAP) data is VPF data produced from city maps. UVMAP data is available for cities in Iraq, and is suitable for use at scales of about 1:5,000 to 1:15,000. The number of feature classes available in VPF format for twenty cities in Iraq is shown in Table 3. When using this data, a reasonable approach is to create feature datasets for each city with spatial references with an envelope large enough to include all of the features for that city, yet small enough so that the envelope does not intersect the view when the user is viewing some other city. It also seems reasonable to incorporate these features in a map by having a group layer for each city that includes the 30 to 70 feature classes for that city.

City Name	Number of feature classes
Ad Diwan	44
Al Amarah	50
Al Basrah	69
Al Faw	30
Al Hillah	61
Al Kut	59
Al Qurnah	42
An Najaf	64
An Nasiriyah	39
Arbil	46
As-Samawah	40
Azzubayr	44
Baghdad	68
Dihok	46
Karbala	51
Karkuk	61
Safwan	46
Suqash	31
Tikrit	59
Ummqasr	51
Total	1,001

Table 3 Feature classes for cities in Iraq

The coverage of these feature classes is as shown in Figure 5. As can be seen in the figure, the feature classes are spread throughout eastern Iraq and typically do not intersect or even come very close to one another, except in the extreme southeast, where Iraq, Kuwait, and Iran come together.

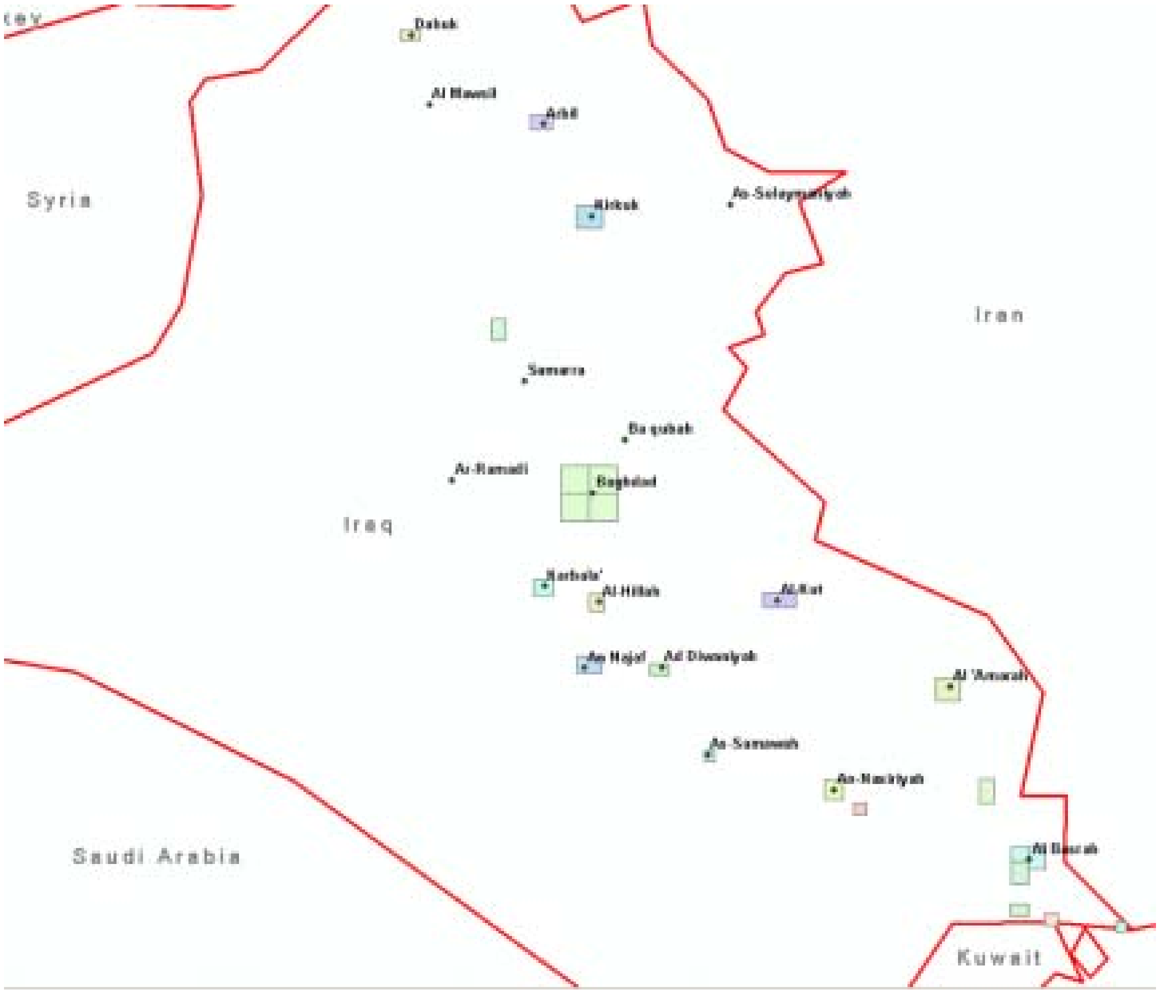


Figure 5 UVMAP coverage for Iraq

It is possible to imagine a typical use of a map that incorporates these feature classes. The user may be browsing along the major roads in Iraq (illustrated in Figure 6), or along the Tigris (Nahr Dijlah) or Euphrates (Nahr Al Furat) rivers, which also pass through most of these cities. In between cities, the user is probably not interested in perusing the map at the same level of detail as might be of interest in the cities. Also, when the user is looking at Baghdad, the other 19 cities are not of as much interest.

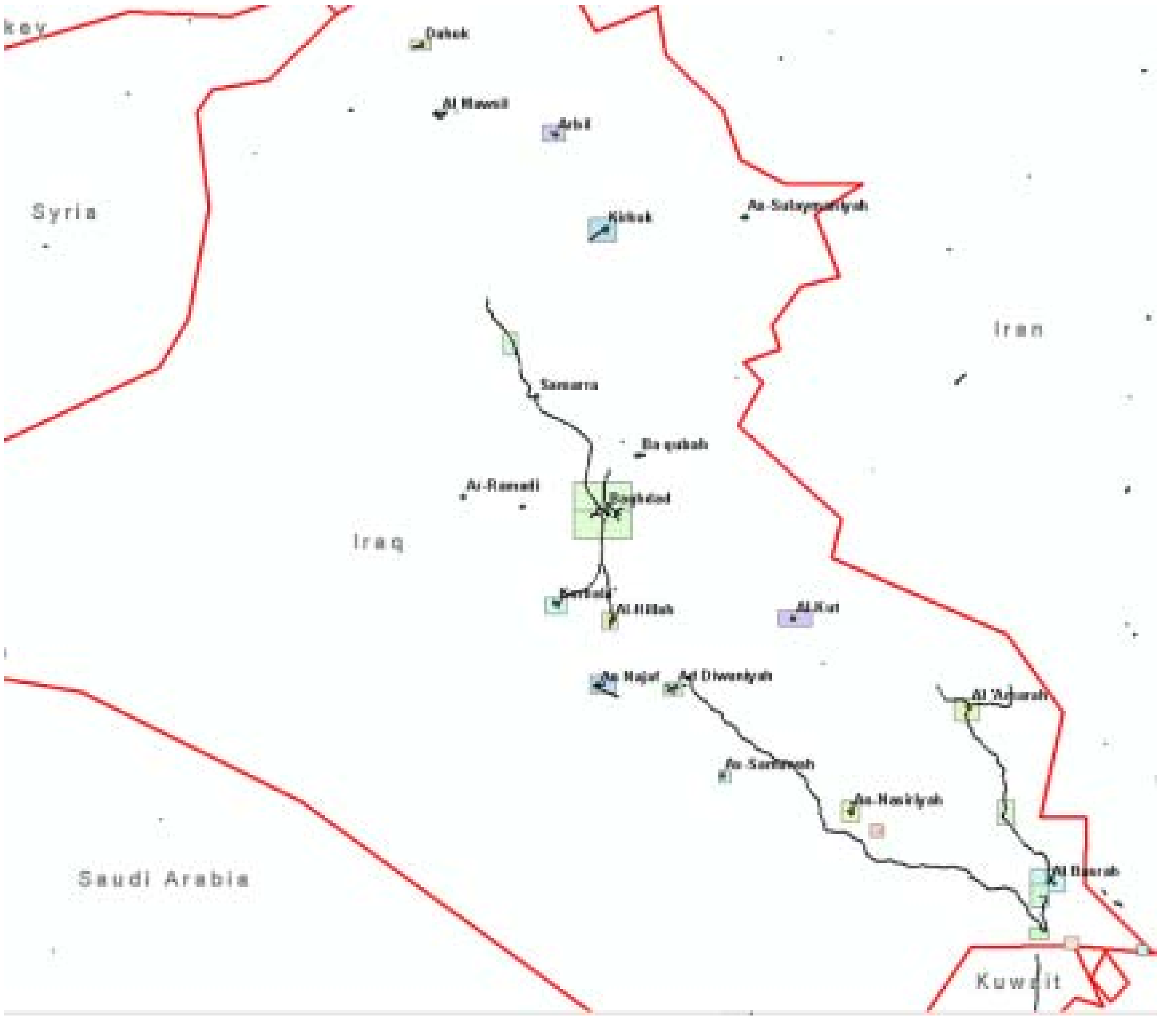


Figure 6 Major roads in Iraq

In order to determine that the features within a city won't be drawn, ArcMap examines the spatial extents of the features. These extents are recorded in the ArcSDE layers table. A Microsoft SQL Profiler trace of the activities that occur within ArcSDE reveals that ArcSDE does not consider the ArcSDE layers table as the authoritative source, even though the layers table is consulted to acquire the table numbers for the feature class's F-table and S-table. Instead, ArcSDE queries against the feature class's business table and the associated F-table. In cases where the spatial reference of the layer is not currently in view, processing ends there. When the spatial reference of the layer does intersect the view, queries are also conducted against the S-table. While these results are informative, it is actually our goal to avoid consulting any of these tables if we can avoid it.

Reducing the Round-Trip Traffic

It seems desirable to avoid the overhead of conducting several round-trips to the ArcSDE server just to determine that features won't be drawn because they cannot possibly be seen in the current view. One approach to achieving this goal is to introduce a customized renderer that takes greater advantage of the spatial characteristics of typical feature data. ArcObjects already includes a `FeatureRenderer`[1] that can be used as a model. This `FeatureRenderer` is called `ScaleDependentRenderer`. Unlike most of the other ESRI-provided renderers, the `ScaleDependentRenderer` does no rendering on its own. Instead, it is a container for other renderers. It delegates the actual rendering to one of these other renderers, depending on the scale of the map view. A `ScaleDependentRenderer` is used as in the following VBA code:[2]

```
Dim pDoc As IMxDocument, pLayer0 As IGeoFeatureLayer
Dim pLayer1 As IGeoFeatureLayer, pLayer2 As IGeoFeatureLayer
Dim pScaleDependentRenderer As IScaleDependentRenderer
Set pDoc = Document
Set pLayer0 = pDoc.FocusMap.Layer(0) ' Get a handle on each layer
Set pLayer1 = pDoc.FocusMap.Layer(1)
Set pLayer2 = pDoc.FocusMap.Layer(2)

Set pScaleDependentRenderer = New ScaleDependentRenderer
With pScaleDependentRenderer
  .AddRenderer pLayer1.Renderer ' Add in the detail renderer
  .Break(0) = 12000000
  .AddRenderer pLayer2.Renderer ' Add in the less detailed renderer
  .Break(1) = 1000000000
End With

' Set the scale break renderer into the first layer.
Set pLayer0.Renderer = pScaleDependentRenderer
```

The interface `IScaleDependentRenderer`, described in C# syntax, is as follows:

```
interface IScaleDependentRenderer {
  // Properties
  void set_Break(int Index, double Break);
  double get_Break(int Index);
  object get_Renderer(int Index);
  int RendererCount { get; set; }
  // Methods
  void AddRenderer(ESRI.ArcObjects.Core.IFeatureRenderer Renderer);
  void MoveRenderer(ESRI.ArcObjects.Core.IFeatureRenderer Renderer, int toIndex);
  void RemoveRenderer(ESRI.ArcObjects.Core.IFeatureRenderer Renderer);
}
```

We could also have an `EnvelopeDependentRenderer` that implements an interface similar to `ScaleDependentRenderer`. Instead of the indexed property **`Break`**, the `EnvelopeDependentRenderer` would have the indexed property **`Extent`**. The interface would be defined, in C# syntax, as follows:

```
interface IEnvelopeDependentRenderer {
  // Properties
  void set_Extent(int Index, ESRI.ArcObjects.Core.IEnvelope Extent);
  ESRI.ArcObjects.Core.IEnvelope get_Extent(int Index);
  object get_Renderer(int Index);
  int RendererCount { get; set; }
  // Methods
  void AddRenderer(ESRI.ArcObjects.Core.IFeatureRenderer Renderer);
  void MoveRenderer(ESRI.ArcObjects.Core.IFeatureRenderer Renderer, int toIndex);
  void RemoveRenderer(ESRI.ArcObjects.Core.IFeatureRenderer Renderer);
```

[1] The Courier typeface is used to indicate code fragments, including type names.
[2] Exploring ArcObjects, p. 473.

```
}
```
This interface, and the CoClass that implements it, may be used in a fashion similar to the IScaleDependentRenderer, as illustrated in this code fragment:

```
Dim pDoc As IMxDocument, pLayer0 As IGeoFeatureLayer
Dim pLayer1 As IGeoFeatureLayer, pLayer2 As IGeoFeatureLayer
Dim pEnvelopeDependentRenderer As IEnvelopeDependentRenderer
Set pDoc = Document
Set pLayer0 = pDoc.FocusMap.Layer(0) ' Get a handle on each layer
Set pLayer1 = pDoc.FocusMap.Layer(1)
Set pLayer2 = pDoc.FocusMap.Layer(2)

Set pEnvelopeDependentRenderer = New EnvelopeDependentRenderer
With pEnvelopeDependentRenderer
  .AddRenderer pLayer1.Renderer ' Add in the detail renderer
  .Extent(0) = GetLayerExtent(pLayer1)
  .AddRenderer pLayer2.Renderer ' Add in the less detailed renderer
  .Extent(1) = GetLayerExtent(pLayer2)
End With

' Set the Envelope dependent renderer into the first layer.
Set pLayer0.Renderer = pEnvelopeDependentRenderer
```

Assuming, as a worst-case, that the features within a layer could extend all the way out to the extent of the spatial reference, the `GetLayerExtent` function could be implemented as:

```
Function GetLayerExtent(pLayer As IGeoFeatureLayer) As IEnvelope
  Dim pFeatureClass As IFeatureClass
  Dim pFeatureDataset As IGeoDataset
  Dim pSR As ISpatialReference
  Set pFeatureClass = pLayer.FeatureClass
  Set pFeatureDataset = pFeatureClass.FeatureDataset
  Set pSR = pFeatureDaset.SpatialReference
  Dim falseX as Double, falseY as Double, xyUnits as Double
  pSR.GetFalseOriginAndUnits falseX, falseY, xyUnits
  dim expanse as Double
  expanse = .5 / xyUnits
  Set GetLayerExtent = new Envelope
  GetLayerExtent.PutCoords falseX, falseY, falseX + expanse, falseY + expanse
End Function
```

Assuming that the features within a layer, even if new features are added, will still be within the bounds established by the existing features leads to the simpler implementation of `GetLayerExtent` shown below:

```
Function GetLayerExtent(pLayer as IGeoFeatureLayer) As IEnvelope
  Dim pFeatureClass As IFeatureClass
  Dim pFeatureDataset As IGeoDataset
  Set pFeatureClass = pLayer.FeatureClass
  Set pFeatureDataset = pFeatureClass.FeatureDataset
  Set GetLayerExtent = pFeatureDataset.Extent
End Function
```

We have another option, assuming that we've structured the map document as indicated by the structure of the source data, with twenty group layers that correspond to the twenty cities of interest and about thirty to sixty layers within each group layer. We should be able to set the `AreaOfInterest` property of each of the group layers to reflect the extent of the feature classes within the group layer. We then would use custom code for the group to avoid the default rendering behavior for the group when the group does not intersect the current view.

Conclusions

For a number of reasons, it makes sense to collect feature classes within feature datasets with small and precise spatial references. Two types of anomalies that may be avoided by taking this approach were illustrated in this paper. In addition to avoiding error, however, an additional consideration is to improve the performance of ArcMap by enabling the client to perform spatial culling of the features without incurring queries to the server.

ArcMap does this culling on a feature class by feature class basis. With careful construction of feature datasets, however, it may be possible to further enhance the culling behavior, and therefore the redrawing performance of a map document by taking advantage of the spatial references of those feature datasets.

Wish List

Since VPF data implicitly uses the WGS84 datum, it would make sense to automatically assume that datum when a default spatial reference is created for it. Currently, the datum remains unspecified, which ultimately means that the default is NAD27.

In many circumstances, it would make sense to create a feature with a NULL shape rather than to ignore the entire record when creating a feature class from data with latitude and longitude columns. In particular, users frequently use coordinates of 0,0 to indicate either that a location is unknown or not applicable.

References

ArcObjects Developer's Guide, Environmental Systems Research Institute, Inc., 1999. ISBN 1-879102-71-4

Exploring ArcObjects, Vol 1 – Applications and Cartography. Michael Zeiler, editor. Environmental Systems Research Institute, Inc., 2001. ISBN 1-58948-001-5.

Exploring ArcObjects, Vol 2 – Geographic Data Management. Michael Zeiler, editor. Environmental Systems Research Institute, Inc., 2001. ISBN 1-58948-002-3.

Author Information

D J Bauch
Contractor, SAIC Information Operations Division
Joint Information Operations Center/J62
2 Hall Blvd, Suite 217
San Antonio, TX 78243-7008
Phone: 210 977 4771
Fax: 210 977 4021
Email: Danny.Bauch@jioc.osis.gov