British Geological Survey

NATURAL ENVIRONMENT RESEARCH COUNCIL

3DG Systems Research and Development: Research into Emerging Spatial Technologies

Information Systems Development Programme

Internal Report IR/06/051

BRITISH GEOLOGICAL SURVEY

INFORMATION SYSTEMS DEVELOPMENT PROGRAMME

INTERNAL REPORT IR/06/051

3DG Systems Research and Development: Research into Emerging Spatial Technologies

A Marchant, K Adlam, P Bell, T Duffy, N A Smith

Keywords

ArcGIS, ArcGIS Server, Web Services, ORACLE Spatial, Disconnected Editing, Remote Data Transfer

Bibliographical reference

Marchant A, Adlam K, Bell P, Duffy T, Smith N A. 2006. 3DG Systems and Development: Research into Emerging Spatial Technoogies. *British Geological Survey Internal Report*, IR/06/051.

Keyworth, Nottingham British Geological Survey 2006

BRITISH GEOLOGICAL SURVEY

The full range of Survey publications is available from the BGS Sales Desks at Nottingham, Edinburgh and London; see contact details below or shop online at www.geologyshop.com

The London Information Office also maintains a reference collection of BGS publications including maps for consultation.

The Survey publishes an annual catalogue of its maps and other publications; this catalogue is available from any of the BGS Sales Desks.

The British Geological Survey carries out the geological survey of Great Britain and Northern Ireland (the latter as an agency service for the government of Northern Ireland), and of the surrounding continental shelf, as well as its basic research projects. It also undertakes programmes of British technical aid in geology in developing countries as arranged by the Department for International Development and other agencies.

The British Geological Survey is a component body of the Natural Environment Research Council.

British Geological Survey offices

Keyworth, Nottingham NG12 5GG

☎ 0115-936 3241 Fax 0115-936 3488
e-mail: sales@bgs.ac.uk
www.bgs.ac.uk
Shop online at: www.geologyshop.com

Murchison House, West Mains Road, Edinburgh EH9 3LA

☎ 0131-667 1000 Fax 0131-668 2683
e-mail: scotsales@bgs.ac.uk

London Information Office at the Natural History Museum (Earth Galleries), Exhibition Road, South Kensington, London SW7 2DE

☎ 020-7589 4090 Fax 020-7584 8270
☎ 020-7942 5344/45 email: bgslondon@bgs.ac.uk

Forde House, Park Five Business Centre, Harrier Way, Sowton, Exeter, Devon EX2 7HU

☎ 01392-445271 Fax 01392-445371

Geological Survey of Northern Ireland, Colby House, Stranmillis Court, Belfast BT9 5BF

☎ 028-9038 8462 Fax 028-9038 8461

Maclean Building, Crowmarsh Gifford, Wallingford, Oxfordshire OX10 8BB

☎ 01491-838800 Fax 01491-692345

Columbus House, Greenmeadow Springs, Tongwynlais, Cardiff, CF15 7NE

☎ 029–2052 1962 Fax 029–2052 1963

Parent Body

Natural Environment Research Council, Polaris House, North Star Avenue, Swindon, Wiltshire SN2 1EU

☎ 01793-411500 Fax 01793-411501
www.nerc.ac.uk

Acknowledgements

Ben Wood is gratefully acknowledged for Java development.

Jason Careless is gratefully acknowledged for strategic advice.

Contents

Summary

This report documents the findings of work, carried out under the 3DG Systems Development and Support project, to investigate a number of emerging GIS and spatial technologies.

The technologies investigated were:

1. **ArcGIS Server:** a development framework for creating advanced GIS Web Applications and Web Services.
2. **Oracle Spatial**: a method for spatially enabling corporate datasets that are currently held in Oracle tables.
3. **Disconnected Editing and Versioning**: methods for managing offline editing of datasets held in corporate databases.
4. **Remote Data Transfer**: methods for the transmission of data to and from field geologists using the MIDAS application.

The report looks at each of these technologies in detail, highlighting advantages and disadvantages over currently used BGS technologies and documenting trial applications that have been developed to test functionality. The code for these trial applications is contained in the Appendices at the end of the report.

As a result of this work, it is recommended that ArcGIS Server has a place in the BGS application architecture and should be considered further by the Information Architecture Steering Committee (IASC). It is recognised that more research needs to be undertaken on the performance of ArcGIS Server, in particular into scalability and stability issues.

The possibility of using ArcGIS Server to develop Web services should be seriously considered, as this would result in GIS specialists developing GIS functionality using their preferred development environment of ArcObjects. Web application developers would then be able to easily consume these ArcGIS Server Web services to enhance the functionality of their applications.

Additionally, if it is shown through a staff survey that there is a need for an internal web based GDI with advanced GIS functionality then it is recommended that ArcGIS Server be used as the development environment for the new application. This would result in BGS staff gaining more experience in programming ArcGIS Server as well as the opportunity to test performance and stability of a major implementation. By using ArcGIS Server, it will be possible to implement more advanced GIS functionality than is currently available through the GeoIndex application.

The research into Oracle Spatial concludes that BGS should proceed to spatially enable Oracle tables using Oracle Spatial/Locator. For new tables, spatial columns should be added directly for simplicity and performance, whilst for existing tables, which may be compromised by altering their structure, spatial columns should be held in related tables.

The best use of disconnected editing and versioning in BGS is in the area of field data capture. However since the current field data capture system uses a hybrid ArcGIS and Access solution, disconnected editing and versioning is not appropriate. It is therefore recommended that further

research on the use of disconnected editing and versioning is not pursued unless the existing field capture system is redesigned in a pure ESRI environment.

The Remote Data Transfer research concludes that it is possible and practical to develop a system that would allow users to upload their data from the field accommodation for use in a viewing application. A viewing application would be of use to field geologists as it would allow them to view data collected in other field areas that may be relevant to the area they are working in. With regard to a system that will allow users to download data whilst they are out of the office it is, again, possible to develop such a system and this would no doubt prove useful in situations where geologists require additional datasets once they are out of the office. However, more investigation should be done as to whether it is necessary to allow users to download another geologists actual field data. It is thought unlikely that this would be necessary or practical.

1 Introduction

This report documents research and development work carried out under the 3DG Systems Development and Support project.

Research was carried out on the following technologies:

1.1 ARCGIS SERVER

ArcGIS Server is a development framework for creating Web GIS solutions. It allows developers working in .NET or Java to implement: centralised internal GIS products, GIS applications accessible via the Internet and Web Services that can be consumed by other Web applications. As ArcGIS Server solutions are built using ArcObjects, complex GIS operations can be implemented, beyond those currently possible using Web GIS solutions such as ArcIMS.

1.2 ORACLE SPATIAL

The BGS's corporate GIS applications such as the GDI and GSD require access to datasets stored in Oracle tables, such as SOBI. Currently this is achieved by copying data at regular intervals from Oracle tables to ESRI shapefiles. As well as the time taken to perform this copying, this method has the disadvantage that the data used by GIS products is not necessarily up-to-date. Oracle Spatial has the potential to allow existing Oracle tables to be spatially enabled, thus allowing GIS products to directly access these datasets.

1.3 DISCONNECTED EDITING AND VERSIONING

Disconnected editing is designed to allow a selected set of spatial features to be extracted from a master ESRI SDE geodatabase (Check-out), edited offline on a laptop or other portable device, and returned to the master database at a later date (Check-in).

Versioning allows multiple versions of spatial data to be held without replication, only the changes are stored. Users can edit the same features and rows without the need for traditional locking mechanisms.

1.4 REMOTE DATA TRANSFER

A number of new methods for the transmission of data to and from field geologists using the Mobile Integrated Data Acquisition System (MIDAS) application were investigated. These include: Secure Tokens / Digital Certificates, Extranet Connection, ArcIMS and ArcGIS Server applications and data transfer via GPRS.

1.5 PROJECT RESPONSIBILITIES

The following BGS staff have worked on the project:

- Keith Adlam: Research into ArcGIS Server. In particular looking at the possible use of the technology in developing an internal GDI product. Research into Oracle Spatial. Research into Disconnected Editing and Versioning.
- Andy Marchant: Research into ArcGIS Server. In particular looking at the possible use of the technology in developing Web GIS applications. Beta testing of ArcGIS Server 9.2.

- Patrick Bell: Research into ArcGIS Server. In particular looking at the possible use of the technology in developing Web Services.
- Tim Duffy: Research into ArcGIS Server using the Java ADF
- Nikki Smith: Research into remote data transfer options for the MIDAS project

2 Research into ArcGIS Server

2.1 WHAT IS ARCGIS SERVER?

ArcGIS Server is not an “out of the box” product. It is a framework for building GIS Web applications and Web Services that are centrally managed, support multiple users, include advanced GIS functionality, and are built and delivered using industry standards. Using either the Java or .NET development environments, developers can create Web GIS solutions with far more GIS functionality than was previously possible. Users are able to access these solutions through a variety of clients including a standard Internet browser.

As well as creating Web GIS applications accessible to external customers via the Internet, ArcGIS Server also allows the creation of centrally managed internal GIS, saving time and costs in implementing GIS at every desktop and allowing GIS services to be authored and managed in a centralised environment. Centrally managed GIS systems reduce the costs of deploying GIS throughout an organisation, make administration tasks easier, and provide for easier integration into other centrally managed IT systems such as corporate databases.

Applications are built using the same software objects (ArcObjects) that are used to build desktop ArcGIS applications. Therefore it is theoretically possible to recreate all ArcGIS desktop functionality in ArcGIS Server. Developer time constraints and network performance issues would make this impractical, but it highlights the potential for adding complex GIS functionality to an ArcGIS Server application. The common use of ArcObjects also means that code created in the desktop environment can easily be converted for use in ArcGIS Server applications.

Prior to ArcGIS Server, ESRI’s only Web GIS product was ArcIMS. ArcIMS is a scalable Internet Map Server for publishing maps, data and metadata over the Web using standard Internet Protocols. ArcGIS Server is not intended to replace ArcIMS, as ArcIMS will continue to be the most suitable technology for map publishing. Instead ArcGIS Server should be used only where it is necessary to provide advanced GIS functionality in a Web GIS application.

ArcGIS Server developers have access to a set of visual Web controls that simplify programming for including mapping and GIS functionality in Web applications. The ArcGIS Server Application Development Framework (ADF) includes the following Web controls to assist with Web application development:

- Map
- Page Layout
- Overview Map
- Table of Contents
- Geocode
- North Arrow
- Scale Bar

- Toolbar (.NET only)
- Impersonation (.NET only)
- Context (Java only)
- Identify Results (Java only)

These controls are available as .NET Web controls and Java Web controls exposed as JavaServer Pages tags. These controls can be combined with other Web controls and components to create Web applications.

ArcGIS Server provides a set of Web application templates as a starting point for developers who want to build applications using the Web controls. The Web application templates include:

- Map Viewer template, which provides basic map display capabilities
- Search template, which provides a search interface for finding features on a map
- Page Layout template, which displays the entire page layout for a map
- Thematic template, which adds thematic mapping capabilities on top of the Map Viewer template
- Geocode template, which provides an interface for finding map locations using an address
- Buffer selection template, which allows you to find features in one layer of the map based on their location relative to features in another layer
- Web Service Catalog, which creates a catalog of ArcGIS Server MapServer and GeocodeServer Web services

The ArcGIS Server .NET Application Development Framework (ADF) runs on Microsoft Windows Server (2003 and 2000) and supports Internet Information Services (IIS). The ArcGIS Server ADF for Java runs on Microsoft Windows Server as well as a variety of UNIX platforms and supports numerous Web servers.

The version of ArcGIS Server used to compile this report was 9.1. However late in the project it was possible to undertake some beta testing of version 9.2 and the initial results of this testing are documented in this report.

2.2 COMPONENTS OF ARCGIS SERVER

ArcGIS Server is made up of 3 components:

1. The Client
2. The Web Server
3. The GIS server

2.2.1 Client

The client as an application that has network capabilities, for example an Internet browser or ArcGIS desktop

2.2.2 Web Server

The Web Server is the software that is responsible for receiving incoming requests from the client and hosting Web applications and services (such as IIS). For the Web Server to host GIS Web applications and services the ArcGIS Server Application Developer Framework (ADF) must also be installed on the Web Server machine.

ArcGIS Server Application Developer Framework (ADF)

The ADF is available in two development environments (JAVA and .NET), both of which may be installed on the same WebServer. The ADF includes a software developer kit containing software objects, web controls, web application templates, developer help and code samples.

2.2.3 GIS Server

The final tier to ArcGIS Server is the GIS Server. The GIS server is responsible for hosting, running and managing server objects, and can be placed on the same machine or on different machines from the Web Server. It includes a SOM and one or more SOCs:

Server object

A server object is a coarse-grained object that manages and serves as a GIS resource. Server objects can be either MapServer objects (maps) or GeocodeServer objects (locators). A MapServer object points to an ArcGIS Desktop project (.mxd). The server objects are themselves ArcObjects components and provide access to the finer-grained ArcObjects such as the map, layers and features etc.

The Server Object Manager (SOM)

The SOM manages the set of server objects that are distributed across one or more container machines. When an application makes a direct connection to a GIS server over a LAN or WAN, it is making a connection to the SOM.

The Server Object Containers (SOC)

The container machine or machines that actually host the server objects that are managed by the SOM.

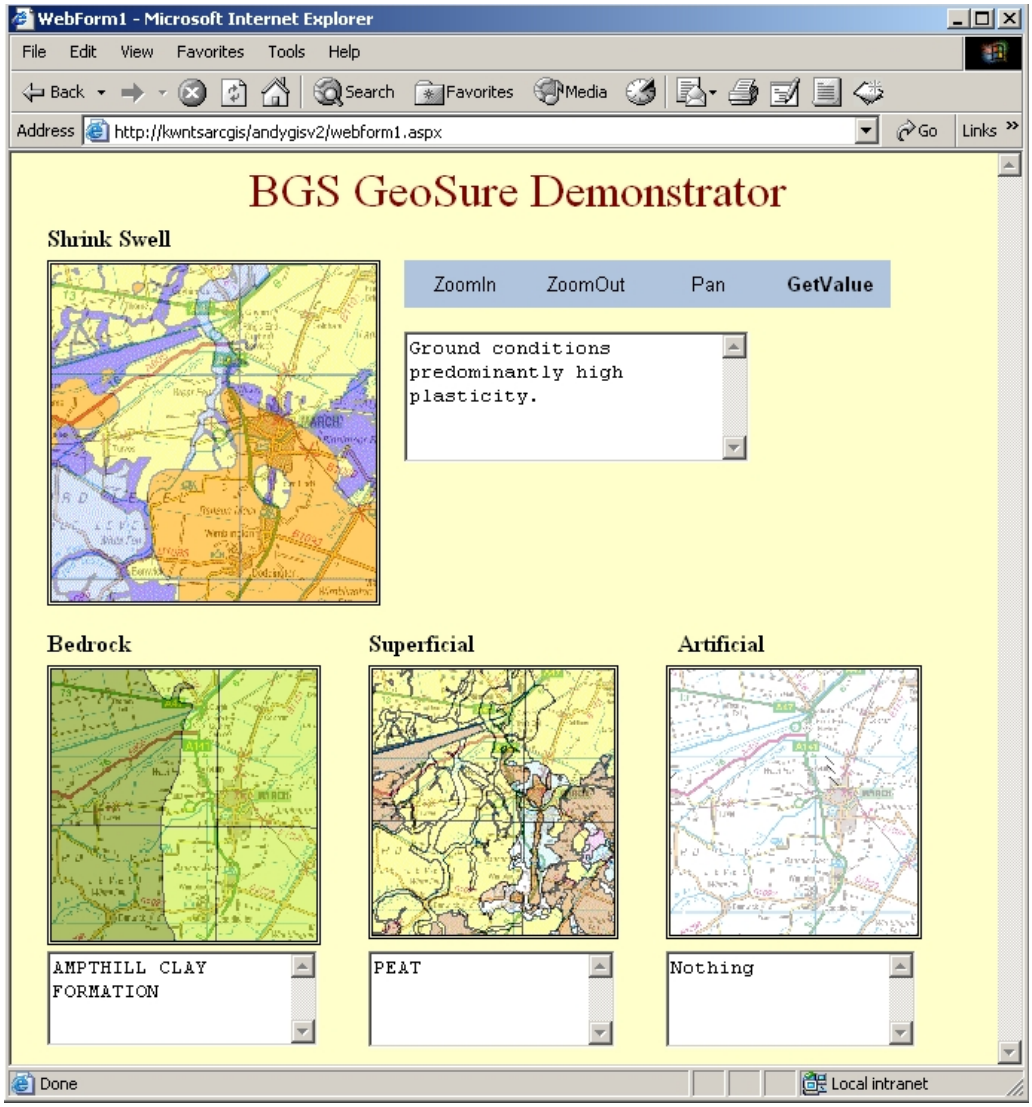
2.3 HOW CAN ARCGIS SERVER BE USED?

There are a number of scenarios in which ArcGIS Server can be used:

- Creating Web Applications
- Creating Web Services
- ArcGIS Desktop Applications
- Centrally Managed GIS

2.3.1 Web Applications

An ArcGIS Server Web Application is a customised application created and designed to run over the Internet or Intranet, which is accessed using a web browser such as Internet Explorer. An example is shown below of an application that allows users to query the BGS GeoSure Shrink Swell dataset. As well as returning the likelihood of Shrink Swell conditions at a location, it also returns the Bedrock, Superficial deposits and Artificial deposits from the DigMap50 dataset.



2.3.2 Web Services

A Web Service is a software component accessible over the World Wide Web for use in other applications. Web services are built using industry standards such as XML and SOAP and thus are not dependent on any particular operating system or programming language, allowing access through a wide range of applications.

An example of an ArcGIS Server Web Service is a tool for calculating scores for each of the six BGS GeoSure datasets, given an Easting and Northing. The results returned by this Web Service are shown below in XML:

```
<?xml version="1.0" encoding="utf-8" ?>
- <GeoSureScores
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://tempuri.org/GeoSureWebService/GetGeoSureV
    alues">
    <ShrinkSwell>D</ShrinkSwell>
    <Compressible>A</Compressible>
    <Collapsible>No Score</Collapsible>
    <RunningSand>A</RunningSand>
    <Dissolution>No Score</Dissolution>
```

Such a Web Service can then be consumed by a Web application client written in, for example: ColdFusion or Java.

A **Web Service catalog** is used to manage ArcGIS Server Web services. A Web service catalog is itself a Web service with a distinct endpoint (URL) and can be queried to obtain the list of Web services in the catalog and their URLs.

2.3.3 ArcGIS Desktop Applications

Users can connect to ArcGIS Server using ArcGIS Desktop applications to make use of map and geocode server objects running on the server. Users can also specify the URL of a Web service catalog to indirectly connect to a GIS server over the Internet to make use of map and geocode server objects exposed by that Web service catalog.

2.3.4 Centrally Managed GIS

ArcGIS Server allows the creation of a centrally managed GIS, saving time and costs in implementing GIS at every desktop and allowing GIS services to be authored and managed in a centralised environment.

To implement a centrally managed GIS architecture, a GIS application would be developed in ArcGIS Server and then accessed by users throughout an organisation using their Web browsers e.g. Internet Explorer. This would have the advantage of improving access to GIS functionality for staff who are more familiar with Web browsers then GIS user interfaces. A BGS example of such an application might be an internal Web based GDI (Geospatial Data Index).

Additionally it would be possible to implement processor intensive GIS tasks such as geoprocessing as centrally managed ArcGIS Server applications. In this way the operation would be carried out on a high performance server machine rather then tying up the resources of the users local PC.

2.3.5 ArcObjects

The common feature to all of the above applications is ArcObjects. In each case ArcObjects will allow developers to code GIS functionality. However whereas a developer in ArcMap has access to a number of high-level objects such as ‘Editor’ and ‘Geoprocessor’, these objects are not available in ArcGIS Server 9.1. Any editing or geoprocessing functionality would have to be built from scratch using fine-grained ArcObjects and would therefore take a lot longer. This situation will improve with future releases, e.g. a high-level geoprocessing object is included in ArcGIS Server 9.2

Developing Web Applications and Web Services will be described in more detail in later sections.

2.4 INSTALLING ARCGIS SERVER

The ArcGIS Server software was loaded on a single server (KWNTSARCGIS) in a simple configuration for research and development purposes. Many configurations are possible allowing production systems to be configured using multiple servers to enable scalability.

The following components were loaded on the server:

- Server Manager
- Container Manager
- .Net Application Development Framework (ADF)
- Java Application Development Framework (ADF)
- Web Server (IIS)
- Servlet Engine (TomCat)
- Visual Studio .Net 2003

The software was easily loaded on the server. The only issue causing a problem was the Post Installation procedure that configures the Server Manager and Container Manager accounts. This problem was related to our domain security policy. The Server Manager Account requires permission to run services and launch batch jobs.

2.5 MANAGING THE SERVER: POOLED AND NON-POOLED OBJECTS

The server is managed using ArcCatalog via a user account that is in the ArcGIS Administrators group (agsadmin). The administrator can add (configure), remove, start, stop and pause server objects, monitor performance, display statistics, set query and output limits, add and remove container machines, configure server directories and set log file location.

After initial configuration, administration mainly involves adding new server objects. This process is easily carried out via the Add Server Object Wizard. The configuration involves setting the following parameters: name, server type (Map or Geocode), description, startup type (automatic or manual), path to map document or locator, dataframe, output directory, image type, pooling type, isolation and recycling. Many of the parameters can be defaulted, the main parameters to be set are the name, type, path and pooling type. The pooling type, pooled or non-pooled determines the uses to which the server object can be put. A pooled object is pre-created on the server allowing applications immediate access. When the application has finished with the server object it is returned to the pool ready for the next application to use. The use of pooled objects is restricted to applications that do not need to change the state of the object on the server. A non-pooled object is created when required and destroyed when the application has

finished using it. Non-pooled objects require greater resources, but have the advantage that they can be used by applications that need to change the state of the server object e.g. applications that add, remove, edit or symbolise layers.

2.6 USING SERVER OBJECTS

After installation a number of Map server objects were configured on the development server. This proved to be very easy, and server objects were easy to display in ArcMap. The next step was to incorporate the server objects into server applications. Two types of application were developed as part of this research (both are documented in the following sections of this report):

1. Server-based Web applications developed within the .Net Application Development Framework (ADF).
2. Web Services that expose GIS functionality to other Internet applications.

2.6.1 Impersonation

Impersonation is necessary to set account permissions on a Web Application or Web Service to access the GIS server. If impersonation is not set up as a user account that is in the ArcGIS Users group (agsusers) an access denied error will be returned.

The easiest way to set up Impersonation is to use the Impersonation Control. Once added to a Web Application, the host, username and password can all be entered via the properties of this control. This information is then encrypted making it unreadable to both users and developers.

It may not be possible to use the Impersonation Control, for example when building a Web Service there is no user interface to place a Web Control on. In this case impersonation is achieved by changing the identity tag in the web.config file as shown below:

```
<identity impersonate="true" userName="user" password="passwd" />
```

However there is a security issue in implementing impersonation in this way. Whilst any users accessing the Web Service cannot access the username and password, this information is stored as text in the development project, so that any developer with access to the project will be able read it.

2.7 DEVELOPING GIS WEB APPLICATIONS USING ARCGIS SERVER

The ADF contains a number of templates for rapid Web Application development:

- Map Viewer
- Page Layout application
- Search Application
- Thematic Map application
- Geocoding application
- Buffer Selection application
- Web Service Catalog application

The templates proved easy to use and provide simple functionality. A number of examples in the ArcGIS Server Administrator and Developer Guide were implemented to gain experience and to begin to learn the coding patterns. Many of the examples are in C# .NET but these were easily converted to VB .NET (our preferred language).

However to make full use of ArcGIS Server, developers will need to extend these templates to produce customised applications. This can be done in two ways:

1. Using the course-grained objects that are provided by the ADF. Each Web Control has an associated course gained object (called a convenience class). For example the map control's convenience class is *WebMap*. This class provides a number of methods, including: panning and zooming the map display, identifying features returning a list of attributes, and finding features by their attributes.
2. Using fine-grained ArcObjects. For the course-grained objects described above it is possible to retrieve fine-grained ArcObjects classes such as Map, FeatureLayer and FeatureClass. It is then possible to develop an application using standard ArcObjects. This method has the advantage that the objects and interfaces used are familiar to ArcGIS desktop developers. An example is shown below where the fine grained ArcObjects Interface *Ilayer* is created from the *WebMap* convenience class, through the *ImapServerObjects* interface:

```
Dim mapServer As IMapServer = WebMap.MapServer
Dim mapDescription As IMapDescription = WebMap.MapDescription
Dim mapName As String = mapDescription.Name
Dim mso As IMapServerObjects = CType(mapServer, IMapServerObjects)

Dim pLayer As Ilayer
'set player to be the first layer in the legend i.e. layer 0
pLayer = mso.Layer(mapName, 0)
```

2.8 THE BASICS OF PROGRAMMING ARCGIS SERVER TO CREATE WEB GIS APPLICATIONS

2.8.1 Working with Coarse-Grained objects and Fine-Grained ArcObjects

When developing Web applications in ArcGIS Server it is vital to consider how the server code will affect the server objects.

To understand this relationship it is important to introduce the concept of the *Map Description* (or *Page Description* for Page Layout objects). This section will discuss only the *Map Description* in relation to a Map Controls, but the same principle applies to the *Page Description* for Page Layout Controls.

The *Map Description* holds the current state of the Map Control including such properties as the visible layers and the current extent. When the Map Control is refreshed the properties of the associated Map server object are read first and then overwritten by those held in the *Map Description*. As a result, any changes made to the *Map Description* during a session affect the way the map displays but do not change the Map Server object, and are lost when the session terminates.

The properties of the *Map Description* can be altered by calling methods exposed by the coarse-grained *WebMap*, *MapServer* and *MapDescription* classes. For instance, layer visibility can be changed through *ILayerDescripton::Visible* (*ILayerDescription* is obtained from *IMapDescription::LayerDescriptions*). Although, as stated above, these changes to the coarse-grained objects only change the *Map Description* and not the Map Server objects, it is possible to force the Map server object to update to the properties held in the Map Description using *WebMap::ApplyMapDescriptionToServer*.

As shown in the previous section the *IMapServerObjects* interface is the key to getting hold of fine-grained ArcObjects. This interface includes the methods *Map, Layer* and *Page Layout*, which allow the developer to get hold of the *IMap, ILayer* and *IPageLayout* interfaces respectively. However working with these fine-grained ArcObjects can permanently change the state of the associated Map Server object, depending on what methods are called (for example adding and removing layers or changing renderers permanently changes the state of the Map server object). For this reason, non-pooled map server objects should be used if changes are to be made to the fine-grained ArcObjects.

One added complication of the properties of a Map Control being held in the *Map Description*, is that if a developer makes a change to the Map Server object through fine-grained ArcObjects e.g. the extent of the map is changed through *IActiveView::Extent,* it might be reasonable to expect that refreshing the Map Control will cause it to zoom to this new extent. However this is not the case as the *Map Description* also holds the extent of the Map Control (through *ImapDescription::MapArea*) and as described previously *Map Description* properties get applied after the map server object properties. The following steps show how to make changes to the fine-grained ArcObjects and then see these changes reflected in the Web Application:

1. Call the WebMap::*ApplyMapDescriptionToServer* to update the Map server object with the properties held in the *Map Description*.
2. Make the changes to the fine-grained ArcObjects.
3. *Call WebMap::RefreshServerObjects* to make the Map Description refresh its properties with the current state held by the fine-grained ArcObjects in the server instance.

2.8.2 Managing the WebMap – Pooled and non-pooled objects

Instantiating a *WebMap* object ties up a MapServer object instance, so it is important how the code in the Web Application manages this object and in particular that the object is released when it is no longer needed. When working with pooled objects if the application does not release the *WebMap* then that MapServer object instance will become unavailable. If, for example, a Map Server object has four pooled instances, once the application has been run four times all the instances of the object will be tied up meaning that any further requests to the application will hang until the SOM refreshes that MapServer object (by default every 10 hours).

The *WebMap* is managed differently for pooled and non-pooled objects. For pooled objects it should be release as soon as the operation on it is complete. For example:

```
Dim webMap As WebMap = map1.CreateWebMap
Try
   ……..……
Finally
   webMap.Dispose()
End Try
```

For non-pooled objects the *WebMap* should exist for the lifetime of that application. It should be held as a Session variable and only released in Session_End. For example:

```
Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
   Dim o As Object
   Dim i As Integer
   For i = 0 To Session.Count - 1
      obj = Session(i)
     If TypeOf obj Is WebMap Then
        Dim WebMap As WebMap = obj
             WebMap.ReleaseServerContext()
     End If
   Next i
   Session.RemoveAll()
End Sub
```

2.8.3 Use of the New Keyword

The ADF runtime does not install ArcObjects, so applications do not have the ability to create local ArcObjects. All ArcObjects that an application uses should be created within a server context using the *CreateObject* method on *IServerContext*. For instance the following code is **incorrect** as it attempts to create local ArcObjects:

```
Dim pPoint As IPoint
Set pPoint = New Point
```

Instead the following code should be used to create the ArcObjects on the SOC machine:

```
Dim pPoint as IPoint
Set pPoint = pServerContext.CreateObject("esriGeometry.Point")
```

2.8.4 Manage the Lifetime of objects

.NET garbage collection can be unpredictable, so it is good practice to control the lifetime of objects. The *WebMap*, *WebGeocode* and *WebPageLayout* objects have a *ManageLifetime* method that should be used to force the release of objects that become out of scope. An example is shown below:

```
Dim webMap As WebMap = map1.CreateWebMap
Try
   Dim mapServer As IMapServer = WebMap.MapServer
   Dim mapDescription As IMapDescription = WebMap.MapDescription
   WebMap.ManageLifetime(mapDescription)
Finally
   webMap.Dispose()
End Try
```

2.8.5 Custom COM components

It is important when writing code using the fine-grained ArcObjects to realise that each call to an ArcObjects method results in a call from the ADF machine to the server which hosts ArcObjects (a SOC machine). Thus certain operations, such as iterating through a FeatureCursor on a layer that contains several hundred features will result in a large number of calls to and from the SOC

machine, which will detrimentally affect performance. However it is possible to package up sections of ArcObjects code as a custom COM component located on the SOC machine. Then when the application runs, a single call is made from the ADF machine to the ArcObjects code on the SOC machine. This code runs locally on the SOC machine (thus any iterations such as FeatureCursors occur quickly), and then once completed the result is sent back to the ADF machine. The following steps are required to create a custom COM component:

1. Create a COM class as if it was to be used in ArcGIS desktop. Possible development environments include: VB6, .NET or VC++
2. Register the COM component on all SOC machines
3. Reference the COM component in the Web application on the ADF machine. Use 'Add Reference' and browse to the .dll
4. Call the COM component from the web application. An example is shown below:

```
'Use Custom component remotely
Dim myMeasureTool As MeasureLineVB.IMeasureLine
myMeasureTool = serverContext.CreateObject("MeasureLineVB.MeasureLineTool")
value = myMeasureTool.MeasureLine(measureLine)
```

2.8.6 Adding New Custom Tools

Instead of using one of the prewritten tools available in the ADF templates, a developer may want to design a new tool. This is done in two parts:

1. Implementing the Client Side action
2. Implementing the Server-Side action

The client-side code (written in JavaScript) enables an end user to interact with a map or page layout control. Once this interaction is complete the JavaScript fires a server-side action (written in .NET / Java).

The most commonly used Client-Side actions are provided with the ADF. The developer can chose from:

- Draw Point
- Draw Line
- Draw Oval
- Draw Polygon
- Draw Polyline

- Draw Circle
- Drag Image
- Drag Rectangle

However the developer may want functionality not provided by these out-of-the-box tools. In this case it is possible to write JavaScript functions to create custom tools. A number of JavaScript functions that provide user interaction with the Web controls are available in the ADF JavaScript Library (for example a function exists to get the cursor x,y location). These functions can be called from a developers own JavaScript and can also be used as a template for creating new functionality. The ADF JavaScript Library can be found in:

<ArcGIS_installation_directory>\DotNet\VirtualRootDir\aspnet_client\esri_arcgis_server_web controls\9_1\JavaScript\ JavaScript_Library.htm

Server-Side code is written by creating a .NET class in the Web application, that implements the 'IMapServerToolAction' interface. This class is passed an argument that contains the results of the client-side action (e.g. a user drawn rectangle), which can then be used in ArcObjects code to implement the required server-side functionality.

Note that it is more efficient for tasks to be carried out client-side in JavaScript, as server-side code requires round trip communication between Client and Server.

2.9 DEVELOPING WEB SERVICES

2.9.1 An introduction to Web services

The W3C Consortium describe Web services as follows:

"The World Wide Web is more and more used for application to application communication. The programmatic interfaces made available are referred to as **Web services***.*

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions thanks to the use of XML. They can be combined in a loosely coupled way in order to achieve complex operations. Programs providing simple services can interact with each other in order to deliver sophisticated added-value services."

A Web service is a reusable function built by one person/organisation that is exposed on the Internet for other persons/organisations to use within their applications.

A Web resource is any object e.g. a Web page, an image or an application etc that is accessible through the Web using standard internet protocols. A service is a piece of software that does work for other software i.e. it is a resource that is designed to be consumed by software rather than by humans and exposes its functionality through an application programming interface (*API*). A Web service has the characteristics of both a Web resource and a service. It is an application that exposes its functionality through an *API*, and it is a Web resource that is designed to be consumed by software rather than by a human sitting at a browser.

Web services are platform and language independent. You can develop a Web service using any language, and you can deploy it on any platform. More to the point, any Web service can be accessed by any other application, regardless of either's language or platform.

Web services simplify the process of making applications talk to each other. Simplification results in lower development cost, faster development times and easier maintenance.

Traditional integration technologies (e.g. CORBA) rely on tightly coupled connections that can break if you make any modification to the application. In contrast, Web services support loosely coupled connections. Loose coupling minimizes the impact of changes to your applications. A Web service interface provides a layer of abstraction between the client and server. Loose coupling allows either piece to change without negatively affecting the other, as long as the interface remains unchanged. Loose coupling reduces the cost of maintenance and increases reusability.

Web services are a practical way of integrating disparate IT systems. They work using widely accepted technologies and are governed by commonly adopted standards. Web Services are applicable to any type of Web environment: Internet, intranet, or extranet. Web services can make it easier to securely connect to business partners or customers. However, a more powerful use of Web services might be in the integration of internal applications, enabling application integration and the reuse of existing code.

Applications can be written as discrete, self-contained objects that can service the needs of one application and, if appropriate, can also be reused to provide functions for other applications. Developers do not need to constantly recreate objects from scratch. Instead, they can plug in appropriate existing objects. This makes application development less expensive. It also enables specialist functionality to be created by the specialists themselves e.g. GIS functionality by the GIS specialists. This leads to the development of higher quality objects and hopefully to a higher standard of application development.

2.9.2 Advantages and Disadvantages of using Web services

Many of the advantages of Web services have already been alluded to. These include:

- **interoperability**: interoperability between various software applications running on disparate platforms.
- **use of open standards and protocols**: Protocols and data formats are text-based where possible, making it easy for developers to comprehend.
- **external integration**: easy integration of software and services from different companies and locations to provide an integrated service.
- **internal integration**: integration of internal applications to enable the reuse of services and components within a Corporate application infrastructure.
- **code reuse**: break down of logic and development of stand alone functions that can be called by many different applications.
- **independence**: code that can be used by many different applications, e.g. a Web service written in ASP.NET can be consumed by a JSP page.

However there are also disadvatages to be considered. These include:

- **performance**: due to the need to parse text-based XML, Web services may suffer from poor performance compared to other distributed computing approaches e.g. CORBA
- **availability**: as with Web sites, Web services will not be 100% available.
- **matching requirements**: any time you create a general service that will handle a variety of customers, you will run into specialized requirements. Some customers might require the one extra little feature that nobody else needs. Web services are envisioned as a "one size fits many customers" technology.

- **interfaces**: if you change existing methods, your customers' programs will break. This is a problem if you find that one of your existing methods is returning wrong answers and can't be repaired because the approach is fundamentally flawed.
- **immature standards**: Web services standards for features such as transactions are still in their infancy compared to more mature distributed computing open standards such as CORBA.
- **staffing and training**: because Web services are a fairly new solution, it can be somewhat difficult to find qualified and experienced staff to implement a workable solution.
- **security**: by utilizing HTTP, Web services can evade existing firewall security measures.
- **performance**: HTTP can provide performance problems due to its transactional nature whereby it is constantly creating and terminating connections between clients and servers.
- **guaranteed execution**: HTTP is not a reliable protocol and doesn't guarantee delivery or a response.

With such advantages and disadvantages in mind, a prudent approach might be to take up Web services slowly, starting with small implementations and building on this once an understanding of Web services has been reached. Due to the small and distributed nature of Web services, taking a slow and controlled approach is very practical.

2.10 DEVELOPING ARCGIS SERVER WEB SERVICES

2.10.1 Introduction

Developers can use ArcGIS Server to build focused Web services. Web services can be implemented within ArcGIS Server using the .NET or Java frameworks to perform GIS functions. Any development language that can use standard HTTP to invoke methods can consume ArcGIS Server Web services. The consumer can get the methods and types exposed by the Web service through its Web Service Description Language (WSDL).

Despite the newness of ArcGIS Server technology, examples already exist where Web services are being using to integrate GIS functionality into business processes. One such example is in the city of Indianapolis. The city's main call centre accepts 1300 calls a day and needs to be able to provide information on topics such as pothole repairs, rubbish collection etc. In order to provide the service, the city's GIS needed to be integrated with its call centre system. ArcGIS Server Web services facilitated this integration. The GIS integration uses two Web services. One provides geocoding, which forces a valid address to be used, and the other uses point-in-polygon spatial analysis functionality to answer the queries and automatically populate fields within the call centre system. The use of Web services means that this functionality can be used in other applications as well with no modification. The integration of the city's enterprise geodatabase with its call centre system via ArcGIS Server has allowed the city to better manage and accurately facilitate calls and determine how they were resolved.

2.10.2 ArcGIS Server example Web service

In order to gain familiarity with the implementation of Web services within ArcGIS Server, the example scenario outlined in chapter 7 of the ArcGIS Server Administrator and Developer Guide was implemented. This Web service finds all toxic waste sites within a certain distance of an address. It expects three input parameters (an address, a zip code and a distance) and returns an array of application defined toxic waste site objects. The example was implemented in .NET, but could just as easily have been implemented using Java.

Details of the example Web Service can be found in Appendix 1.

2.10.3 Building a BGS GeoSure Web service

Experiments with the sample ArcGIS Server Web service proved the concept of integrating GIS functionality into ColdFusion applications. The next stage was to develop a BGS example. GeoSure seemed an ideal candidate and a Web service was developed in ArcGIS Server that expected an input of an easting and a northing and returned a value of A-E for each of the six GeoSure hazards for that location. As part of the investigations into ArcGIS Server, an

application had already been developed that performed this same functionality. This provided an opportunity to see how quickly an existing piece of functionality could be converted into a Web service and consumed by an external application.

It took approximately one hour to convert the existing GeoSure application into a Web service and to write a ColdFusion application to consume its functionality. It is appreciated that the structure of the Web service generated might not be the most elegant, but the concept was successfully proven for the purposes of this research and development exercise.

An example client was also developed in Java to demonstrate consumption of ArcGIS Server Web services in another BGS development environment. As ColdFusion is in fact built on top of Java, the two methods are very similar behind the scenes, though the work required from the developer is different.

Details of the BGS GeoSure Web Service can be found in Appendix 2.

2.10.4 Building the GeoSure Web service using ColdFusion and ArcIMS

The GeoSure example was ideal for testing the principles of Web Services in ArcGIS Server. It provided a good example of a common function required by external customers. As an application with the same functionality had already been developed within ArcGIS Server, it also allowed us to discover how quickly existing functionality could be repackaged and reused in this way.

This example also provided another opportunity. ArcGIS Server provides advanced mapping, geoprocessing and spatial data management. It is a comprehensive platform for delivering enterprise GIS applications that are centrally managed and support multiple users. It provides comprehensive server-based GIS solutions built using ArcObjects. Our GeoSure function is relatively simple in that it is passed an easting and a northing and returns hazard values. This functionality is within the realm of ArcIMS, which is considered to be a publishing tool for GIS maps and data. Although ArcIMS cannot publish Web services directly, its various connectors enable its functionality to be integrated into technologies that can publish Web services such as ColdFusion and Java. These are technologies with which the BGS has more experience and it seemed reasonable to create the same Web service using these technologies and compare it with that published by ArcGIS Server.

Details of the GeoSure Web service using ColdFusion and ArcIMS can be found in Appendix 3.

2.10.5 Comparing the ArcGIS Server and ColdFusion/ArcIMS Web services

The GeoSure Web services published using ArcGIS Server and ColdFusion/ArcIMS took a similar length of time to develop from a starting point of having to convert existing code. Both function well and appear, given the limited use they have received, to be stable and responsive. It would be of limited value at this stage to do an in depth performance analysis on the two Web services as they are deployed on different machines with greatly differing specifications. (This is a result of making use of existing software installations. If it is felt that a detailed performance analysis is required then one could be done.)

As well as performance, other issues are equally as important in deciding on the strategic value of ArcGIS Server and ArcIMS in supplying GIS functionality in the form of Web services. ArcIMS has the advantage of being technology that is already in used within the BGS and in which we have expertise. A disadvantage is that it needs to be combined with another technology e.g. ColdFusion or Java in order to publish its functionality in the form of a Web service. There is also a limit to the complexity of GIS functionality that can be provided using ArcIMS. Although the functionality of ArcIMS can be extended beyond purely publishing information with the aid of additional technologies, there is a limit to what can be achieved. Combining multiple technologies in this way to provide a solution might tend towards a rather unwieldy application architecture.

ArcGIS Server involves the introduction of a new technology and the need to maintain it and a skills level in it. However, two arguments are put forward for its use in a BGS application architecture. The first is that it makes full use of ArcObjects and so potentially could offer any GIS functionality within one defined technology. The second, and perhaps more important, is that it would allow the GIS specialists to develop these GIS functions in their native development environments. ArcIMS development is very Web-orientated and is largely unfamiliar to GIS developers. However, ArcGIS Server would enable the GIS developers to develop the functionality in their usual .NET environments and make it available for other developers to use. This would mean that the people who know best how to write GIS functionality will be writing the functions. They will also be writing them in their familiar development environment and so the standard of the supplied functions will be maximised.

2.10.6 Summary of Web Services findings

The ability of ArcGIS Server to provide GIS functionality that can be integrated within other applications through the publication of Web services has been investigated. It is suggested that ArcGIS Server could successfully fulfil this development need because:

GIS specialists can develop GIS functionality using their preferred development environment (currently .NET). This will ensure a high standard to the functions developed. Other application developers can easily consume these ArcGIS Server Web services to enhance the functionality of their applications. This has been proven in both ColdFusion and Java, currently the two most important development environments within the BGS.

The functionality provided by such Web services appears stable, but it is acknowledged that a lot more testing in this area would be required.

The emergence of ArcGIS Server technology is very timely as it comes when the BGS has set up its Information Architecture Steering Committee (IASC) with the aims of gaining an integrated approach to database and application design across the BGS. One challenge facing the committee is integrating GIS application development and other application development into a single application architecture when they tend to employ different technologies. Web services may enable the two areas to come together without the expense and time that would be required in retraining one group to become proficient in another technology e.g. for the GIS developers to learn Java.

Web services are platform independent. This will enable disparate platforms to be used for different functions. Application architecture within BGS is currently concentrating on development environments, but might in the future turn its attention to the platforms on which the applications run. It might be proven that a ColdFusion application runs best on a Linux platform, but ArcGIS Server runs best on Windows. Web services would enable such cross-platform integration.

As well as improving the internal integration of application development, ArcGIS Server Web services might enable future provision of services to BGS customers. The BGS currently supplies data to customers on CD and whilst in some cases this would continue to be necessary, it might be able to reduce this by providing Web services. One example might again be the GeoSure data. Customers such as Sitescope use this information in their property reports for conveyancing purposes. Access to a Web service providing this information would ensure they obtain the latest available information from the BGS. It would also make it easier for the BGS to control access to its licensed data. Web services can be secured to ensure that only licensed users can access the information.

As described earlier, Web service technology has issues that need to be resolved. Investigations into the scalability and stability of ArcGIS Server Web services in a production environment have not been carried out. However, their ease and speed of development has been proven. This research and development exercise concludes that ArcGIS Server has a place in the BGS Application architecture and should be considered further by the IASC.

2.11 ARCGIS SERVER GIS FUNCTIONALITY TESTING

This section examines some important issues in developing GIS functionality, highlighted during the development of example applications in ArcGIS Server.

2.11.1 Editing in ArcGIS Server

ArcGIS Server does not have a high level editor like ArcMap. Any editing functionality needs to be built using fine-grained ArcObjects. A simple editing Application is described in the ArcGIS Server Administrator and Developer Guide.

This was translated from C# to VB .NET and successfully implemented on the development server. The application allows polygons to be captured along with a single attribute. No editing or delete functionality is present and this would need to be implemented by further use of fine-grained ArcObjects. This example does give a starting point in the development of a web-based editor but it also demonstrates that the development of full editing capabilities such as those in ArcMap would take a considerable amount of coding.

2.11.2 Geoprocessing in ArcGIS Server

Using ArcGIS 9.1 Desktop developers are able to get hold of a high level geoprocessing object using: pGP = CreateObject("esriGeoprocessing.GPDispatch.1"), which gives access to all the commands that are available in ArcToolbox. However this method was found not to work in ArcGIS server. This leaves two options for performing Geoprocessing in ArcGIS server 9.1:

1. Implementing all required geoprocessing using fine-grained ArcObjects
2. Using the IBasicGeoprocessor Interface, which performs a number of basic operations (Clip, Dissolve, Intersect, Merge, Union). A test application was written in ArcGIS Server, which was successfully able to use the IBasicGeoprocessor Interface.

Example code for using the IBasicGeoprocessor can be found in Appendix 4.

2.11.3 Web Service Catalogs

Web Service Catalogs allow developers to organise active server objects. Server objects can be added to one or more Web Service Catalogs. This would be done for one of two reasons:

1. To provides a way to make server objects accessible over the Internet via HTTP as Web services. ArcGIS Desktop users can directly connect to a Web Service Catalog over the Internet and utilize the server objects exposed through it and, for example, add a MapServer object to ArcMap.

2. To organize server objects for specific groups of people. Staff would only be given access privileges for web service catalogs that contain the server objects they will need to use.

A simple template is provided in the ArcGIS Server Administrator and Developer Guide. This template was successfully converted from C# to VB .NET, tested and a number of Web Service Catalogs were created.

2.12 ARCGIS SERVER OR ARCIMS?

Currently all BGS GIS Web applications are built using ArcIMS. The advantages and disadvantages of using this technology are summarised below:

Advantages	Disadvantages
ArcIMS is technology that is widely used within BGS and in which there is a lot of expertise.	There is a limit to what can be achieved in terms of GIS functionality using ArcIMS and ColdFusion / Java.
After several years of creating ArcIMS Web sites, security and performance issues are well understood.	ArcIMS development is very Web-orientated and is largely unfamiliar to GIS developers.
ArcIMS was designed for map presentation over the Internet, and performs this function very well	
Functionality of ArcIMS can be extended beyond purely publishing information with the aid of additional technologies such as ColdFusion or Java	

As a result of the testing of ArcGIS Server for the implementation of Web applications and Web services, the advantages and disadvantages are summarised below:

Advantages	Disadvantages
ArcGIS Server allows development using ArcObjects and so potentially could offer any GIS functionality within one defined technology. Possibilities include: • Editing • Geoprocessing • Cutting out and supplying data online	ArcGIS Server is new technology. BGS only has a little experience in application development. Security and Performance issues are not well understood
Allows GIS specialists to develop GIS functions in familiar development environments such as VB .NET	Few other organisations have implemented ArcGIS Server Web applications / Web Services, so the opportunity to learn through experience is limited
Web Services implemented in ArcObjects can be consumed by applications built by BGS Web developers in development environments with which they are familiar e.g. Coldfusion / Java	High-level programming objects that are available in ArcGIS Desktop are not available in ArcGIS Server (e.g. Editor and Geoprocessor). Their functionality can be replicated using ArcObjects but this involves a large amount of code and therefore time. It is likely, however, that additionally high-level objects will become available in future releases of ArcGIS Server.

The ESRI website (www.esri.com), describes the relationship between ArcIMS and ArcGIS Server as:

"ArcGIS Server is a server-based deployment of the ArcObjects component library (including extensions). This new product is for developers who want to build shared server applications that contain advanced GIS capabilities in both a client/server environment and a Web services environment. ArcObjects provides the source of the serverside functionality.

In contrast, ArcIMS is ESRI's Internet solution for the publishing of maps, data, and metadata. Its goal is to deliver data to many users on the Web. ArcIMS is designed for high throughput, high-performance mapping, metadata services, data streaming, and a variety of focused functions over the Internet (via XML, HTTP, etc.). ArcIMS is aimed toward developers looking for a traditional way to serve maps on the Internet and also build custom Internet solutions."

It is clear that ArcGIS Server will not replace ArcIMS and that both technologies have a part to play in delivering Web GIS. The nature of the product will determine which technology is used. For example:

- Map Display: ArcIMS
- Simple querying: ArcIMS & Cold fusion
- Complex GIS functionality: ArcGIS Server Web application / ArcGIS Server Web service

2.13 POTENTIAL ARCGIS SERVER APPLICATIONS

2.13.1 Web Applications

2.13.1.1 DEVELOPMENT OF A INTRANET/INTERNET GEOSCIENCE DATA INDEX

As part of this research some of the basic functionality of the Geoscience Data Index was implemented in ArcGIS server to establish how easy if would be to develop a Geoscience Data Index using ArcGIS Server, and to compare the advantages and disadvantages of using ArcGIS Server to the ArcIMS and ArcMap implementations.

The MapViewer template was used as a starting point for this development. This provided simple pan, zoom and identify functionality. The priority was to implement further navigation by gazetteer, national grid tile etc. The implementation of navigation by national grid square and by gazetteer was proven.

There was insufficient time to further develop the demo but the development did provide an excellent insight into the possibilities and drawbacks of using ArcGIS server.

ArcGIS Server comes with simple templates aimed at end users with little or no knowledge of GIS. The out of the box functionality allows simple navigation and query. Additional means of navigation can be added relatively easily without excessive coding. Additional means of query may take longer to add but is certainly possible. ArcGIS Server is very extensible and given time and expertise virtually any functionality could be built. However there is little high level functionality so a lot of coding would be involved to develop anything like the full functionality of the desktop based GDI. The refresh performance of the web based interface is poor compared to ArcMap but is similar to that of ArcIMS. The introduction of too many layers in a MapServer object slows the system considerably. This is analogous to the sort of issues we had with ArcIMS that resulted in us having to create groups of layers with each group having its own map service rather than having everything together.

The Internal Geoscience Data Index currently provides functionality to more than one user community. Some users require simple navigation and query while other require full desktop functionality. Many GIS users use the GDI as a starting point for their own development as many of the tools provide useful functionality.

Part of the GDI user community could be transferred to a simple web version that could be created to sit alongside the desktop version and look at exactly the same data. This would enable users not familiar with the ESRI desktop applications to do simple navigation and query of spatial data. However, such an interface could not provide for the entire GDI community.

To gain experience of developing Web applications using ArcGIS Server, it is recommended that a major internal application is developed first. If a staff survey shows the need for an internal Web based GDI with advanced GIS functionality, then this would be an ideal candidate, allowing developers to investigate the performance and security issues of ArcGIS Server with the long-term aim of making similar applications available externally.

2.13.1.2 SUPPLY OF BGS DATA

A long-term goal of BGS is to supply its digital datasets and updates online. It would be possible to implement an ArcGIS Server application, which allows the user to draw (e.g. a rectangle or polygon) or select (e.g. from a list of Local Authority boundaries) an area of interest. BGS data would then be cut out for this area and then made available to the customer for download. This method would then replace the need to send out CD's to customers.

2.13.2 Web Service Applications

2.13.2.1 VALUATIONS OFFICE GEOSURE APPLICATION

BGS has implemented an ArcIMS / Coldfusion Web application for the Valuations Office, which takes a grid reference or a UK address as an input and then returns scores from the six GeoSure datasets for that location. The scores are returned in pdf reports, either as text only or as a combination of text and map extracts. This application performs very well, but in the future as BGS provides more services of this type, it's likely that they will require more complex querying of GIS datasets, beyond that possible using ArcIMS.

Implementing such an application as an ArcGIS Server web service means that the Web user interface will continue to be produced by Web developers in their choice of development environment, whilst the GIS querying functionality can be implemented in ArcObjects by GIS developers. The Web application would simply call the GIS Web service through an agreed interface.

2.13.2.2 QUERYING OF BGS DATA BY EXTERNAL CUSTOMERS

Customers such as Sitescope use GeoSure information in their property reports for conveyancing purposes. Currently after each new release of GeoSure the data is sent to the customer on a CD. Access to a Web service providing this information would negate the need to send out CDs and ensure the customers uses the latest available information from the BGS. It would also make it easier for the BGS to control access to its licensed data. Web services can be secured to ensure that only licensed users can access the information

2.13.2.3 GEOREPORTS

Currently GeoReports is implemented as two distinct applications, a back office web application and an ArcView 3.3 GIS application. Orders are taken using the Web application and then enquiries staff run the GIS application to generate each report.

Through the use of ArcGIS Server Web Services it would be possible to produce a single GeoReports Web application. A Web Interface would take orders in a similar way to the current system, but would then also generate the GeoReport, calling a Web service each time a map or the results of a GIS query are required.

2.14 FUTURE RELEASES: ARCGIS SERVER 9.2

The main change to ArcGIS Server at version 9.2, is that the product will be split into two versions, based on a choice of development environments:

1. ArcGIS Server .Net ADF
2. ArcGIS Server Java ADF

The following additional functionality will also be included:

- A new "out of the box" client viewer (similar to ArcExplorer)
- A Globe server object for 3D applications
- A Geoprocessing server object
- A Geodata management object for working with geodatabases e.g. versioning
- Inclusion of the Maplex extension
- Inclusion of the Data Interoperability extension

2.14.1 Testing of ArcGIS Server 9.2 Beta

In February 2006, as part of ESRI's Beta testing program, BGS received the Beta version of ArcGIS Server 9.2. This has allowed a limited amount of functionality testing, the results of which are described below.

2.14.2 Installing ArcGIS Server 9.2 Beta

The installation of ArcGIS Server 9.2 Beta was relatively straightforward. As expected the ADF was split into the .NET ADF and the Java ADF both of which were installed alongside each other. However it soon became apparent that the .NET ADF no longer works with .NET 2003 and the .NET Framework 1.1. It was therefore necessary to install .NET 2005 and the .NET Framework 2.0.

2.14.3 First impressions of ArcGIS Server 9.2 Beta

ArcGIS Server 9.2 is a major change from 9.1, not only with the move from .NET 2003 to 2005 but also because the Web ADF has been completely re-architected to support building web applications and web services that access additional GIS servers such as ArcIMS.

Because the Web ADF has been completely re-architected, the new web controls and templates are completely different from those used to build 9.1 applications. However it is still possible to run applications built using the previous version, as the 9.2 Web ADF includes runtime support for existing applications. When an existing application was opened in Microsoft Visual Studio

2005, the 9.1 web controls appeared in that application, but it was not possible to create new applications with 9.1 controls because they are no longer available in the Toolbox; only the new 9.2 web controls are there.

This means that there will be a learning curve to undertake for developers who wish to create 9.2 applications. However the extra functionality available (in particular Geoprocessing) should mean that at this release ArcGIS Server is able to undertake tasks that are currently not possible using ArcIMS, with very little programming. For instance, the example described in section 2.14.5 shows how a web GIS application can call a geoprocessing model built with Model Builder.

2.14.4 Converting ArcGIS Server 9.1 applications to 9.2

Opening an application built for in .NET 2003 and ArcGIS Server 9.1, in .NET 2005 and ArcGIS Server 9.2 is straightforward, unless the application contains additional tools and buttons created by the developer, and the server side code is held within .NET classes. In this case converting the application does become a little more complicated. The steps required to perform this conversion are described below:

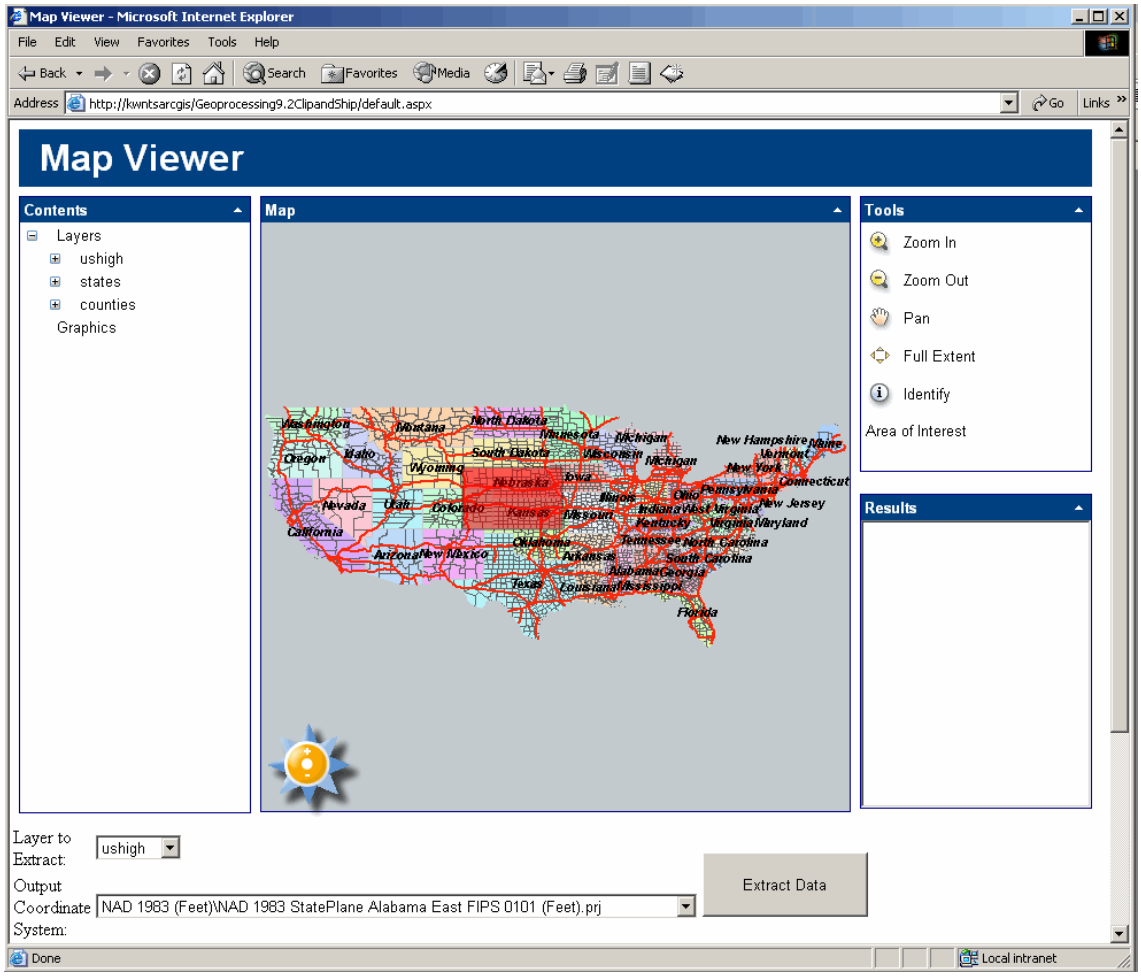
1. Open .NET 2005 and select Open Web Site from the file menu. Using the File dialog window open the ArcGIS Server 9.1 application created in .NET 2003.
2. A conversion wizard for converting .NET 2003 code to .NET 2005 starts automatically. Before any conversion is done a backup copy of the project is made. Once the conversion wizard completes, if the application contains no code stored in .NET classes (i.e. server side code for tools and buttons), then the application should run successfully, otherwise follow step 3 onwards.
3. Right click the solution in the Solution Explorer and click *Add*, then click *New Project*. Under *Templates* click *Class Library*. This creates a new project in the Solution to hold the tools implementation.
4. Copy the class code from the 2003 project (it will be visible in the Solution Explorer) to the new project.
5. Add all necessary references to the new project, making sure that all references used in the original 2003 project have been included. If these are not found using *Add Reference*, try *Add GIS reference*.
6. Right click on the website project in the Solution Explorer and click *Add Reference* followed by *Browse*. Point the File Dialog to the location of the .dll your new project has created. This will be under the bin\debug directory of the new project.
7. It should now be possible to add the tools to the toolbar via the *Properties* page of the toolbar control.

2.14.5 Geoprocessing in ArcGIS Server 9.2 Beta: Clip and Ship example

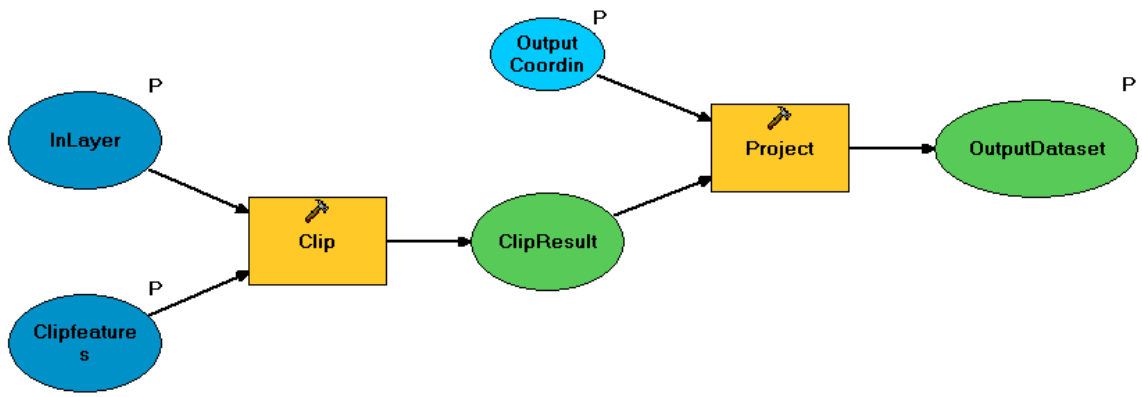
One of the samples supplied with the Beta version is a simple Clip and Ship. In order to gain familiarity with 9.2 this sample was successfully converted from C# to VB .NET.

The sample is an excellent example of how powerful geoprocessing could be in ArcGIS Server, not just for delivering BGS data but also for executing any pre written geoprocessing models/scripts through a web application. This is achieved by adding the toolbox (.tbx file) to a .NET project using the new *Add ArcGIS Toolbox Reference tool* option, which generates a managed .NET assembly for the tools held in the toolbox. It is then possible to use a geoprocessing object (similar to that used in Desktop), to run any of the tools held in the toolbox.

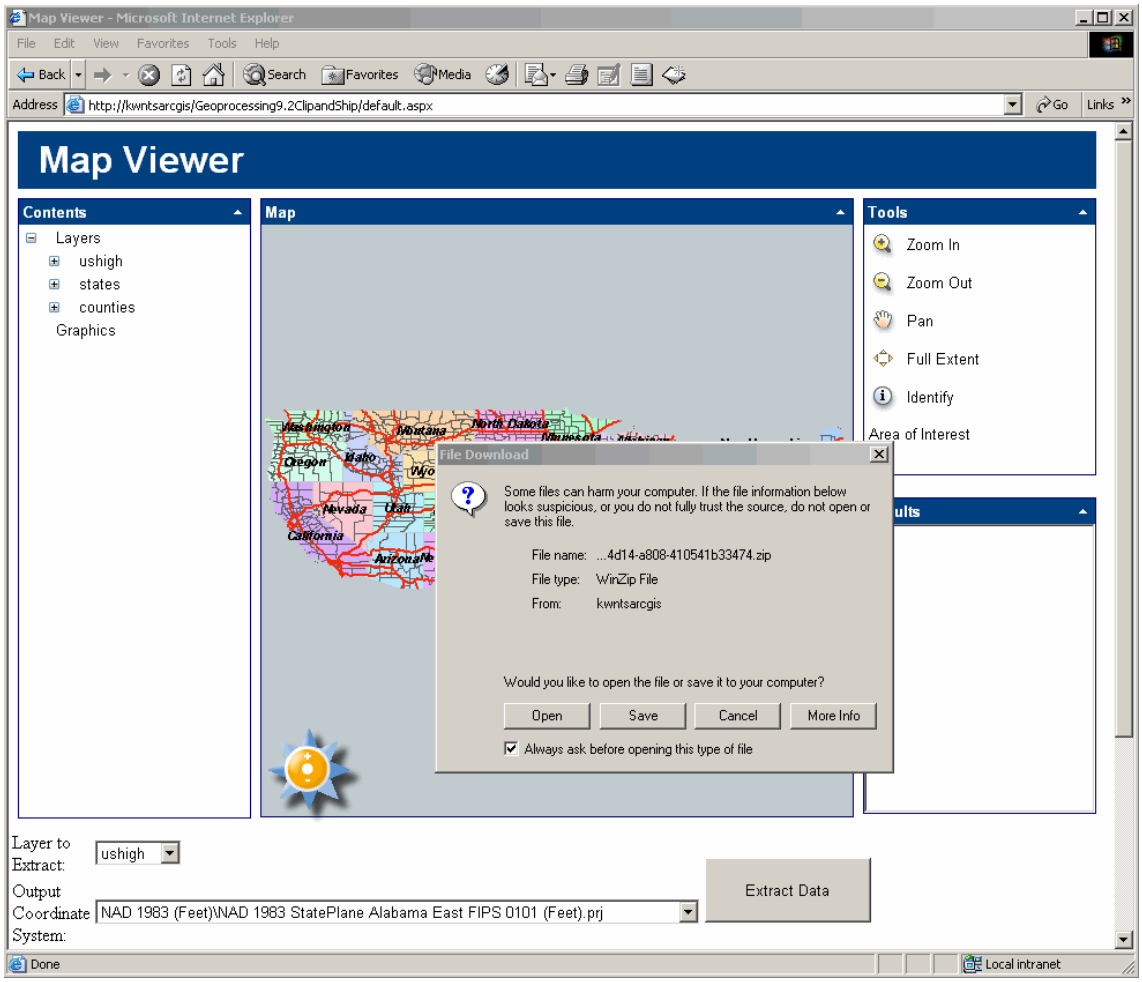
In this sample application, the user first specifies a rectangle defining the area of interest:



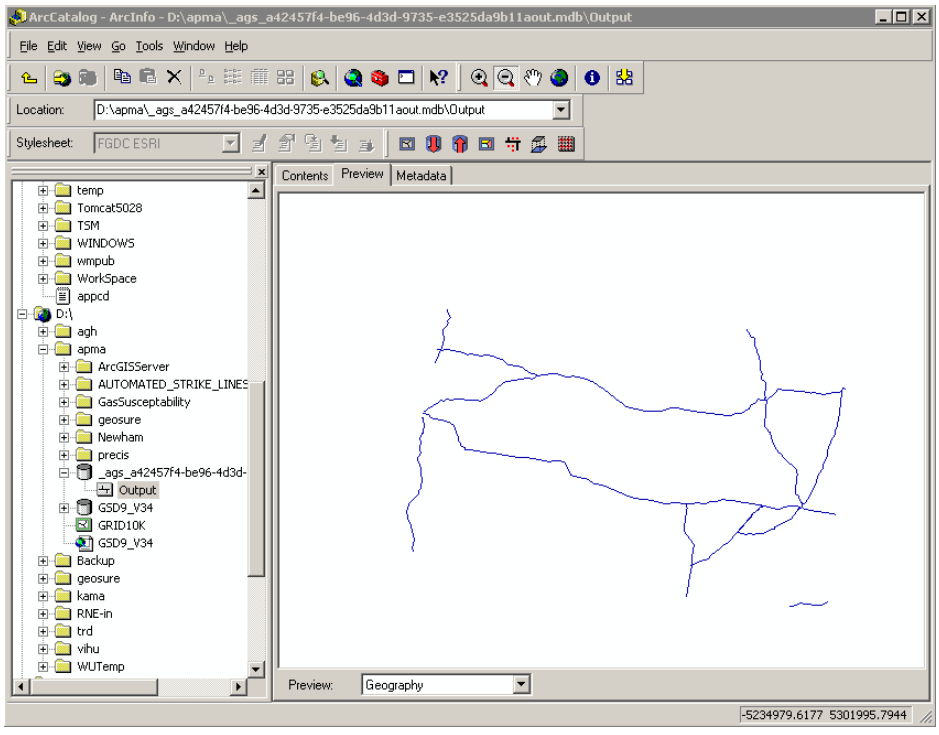
The application then runs a model built previously using ArcGIS Desktops Model Builder to clip the datasets to this rectangle and project the new data. The model is shown below:



The clipped and projected data is then zipped up, placed in a shared output directory and made available to the user for download:



The downloaded data can then be used as required. For example, below, it is previewed in ArcCatalog:



2.15 ARCGIS SERVER CONCLUSIONS

General Conclusions:

- ArcGIS Server is not an “out of the box” solution. It is a platform and a toolkit to allow developers to create server-side GIS applications.
- ArcGIS Server makes full use of ArcObjects and so can provide advanced GIS functionality.
- Creating simple functionality based on the templates provided is easy and extending these templates in limited ways can be easily achieved.
- In ArcGIS Server 9.1, no high level Geoprocessor or Editor objects exist as they do in ArcGIS Desktop. Therefore any Editing or Geoprocessing operations need to be coded using fine-grained ArcObjects. This will involve a significant amount of developer time.
- The screens refresh performance of web based GIS is poor compared to desktop applications.
- Web based GIS applications can reach a wider audience than desktop based systems
- The next release of ArcGIS Server (9.2) will include powerful additional functionality including Geoprocessing

Web Application Development Conclusions:

- ArcGIS Server could provide some Geoscience Data Index type functionality but could not provide the same performance and level of functionality provided by the desktop version without major development. A simple navigation and query interface, with some additional GIS functionality could provide functionality for a subset of users.

Web Services Conclusions:

- The BGS has set up its Information Architecture Steering Committee (IASC) with the aims of gaining an integrated approach to database and application design across the BGS. The publication of GIS functionality via Web services would enable the integration of GIS and other application architectures within the BGS without the need to change either’s current development environment, thus avoiding the associated retraining and cost issues inherent in such a switch.
- GIS specialists can develop GIS functionality using their preferred development environment (currently .NET and ArcObjects). This will ensure a high standard to the functions developed.
- Other application developers can easily consume these ArcGIS Server Web services to enhance the functionality of their applications. This has been proven in both ColdFusion and Java, currently the two most important development environments within the BGS.

- It has been proven that existing ArcGIS Server code can be very quickly repackaged to enable it to be published via Web services.
- Web services are platform independent. This will enable disparate platforms to be used for different functions. Application architecture within BGS is currently concentrating on development environments, but might in the future turn its attention to the platforms on which the applications run. It might be proven that a ColdFusion application runs best on a Linux platform, but ArcGIS Server runs best on Windows. Web services would enable such cross-platform integration.
- Web services could enable the BGS to provide GIS data and functionality more efficiently to external customers and to integrate such data and functionality with business partners to develop new value-added systems.

2.16 ARCGIS SERVER RECOMMENDATIONS

- More research needs to be undertaken on the performance of ArcGIS Server. In particular into scalability and stability issues.
- Until BGS has more experience in its use, it would be prudent to proceed with caution and only use ArcGIS Server to develop applications that cannot be developed using existing tried and tested technologies.
- The need for an Intranet Geoscience Data Index / GeoIndex is currently being investigated by means of a staff questionnaire. If this need is proven and users require significant GIS functionality, it is recommended that ArcGIS Server be used as the development environment for the new application. This would result in BGS staff gaining more experience in programming ArcGIS Server as well as the opportunity to test performance and stability of a major implementation. By using ArcGIS Server, it will be possible to implement more advanced GIS functionality than is currently available through the GeoIndex application.
- The possibility of using ArcGIS Server to develop Web Services should be seriously considered. This would result in GIS specialists developing GIS functionality using their preferred development environment of ArcObjects, whilst Web application developers would be able to easily consume these ArcGIS Server Web services to enhance the functionality of their applications.

3 Research into the use of Oracle Spatial/Locator in BGS

3.1 INTRODUCTION TO THE USE OF ORACLE SPATIAL

Oracle Spatial licenses have recently been acquired by BGS via a NERC deal. This research project was designed to evaluate the functionality of Oracle Spatial particularly with a view to eliminating the need to constantly copy data from Oracle tables to ESRI shapefiles to support GIS usage. The project considered various methods and mechanisms to spatially enable existing Oracle tables using Oracle Spatial and recommends the best way to achieve this goal.

3.2 METHODS CONSIDERED TO SPATIALLY ENABLE ORACLE TABLES

Three options were considered to spatially enable existing tables:

1) Adding a Geometry field directly to existing tables.
2) Creating a new table (geometry table) consisting of a geometry field and a unique identifier that can be related to an existing table via a view, where the new table is created in the same Oracle Schema as the table to be spatially enabled.
3) Create a new table consisting of a geometry field and a unique identifier that can be related to an existing table via a view, where the new table was created in a different Oracle Schema to that of the table to be spatially enabled.

The first of these three options was discounted as a means of spatially enabling existing tables. Adding a geometry field to existing tables may have undesirable and unknown effects on existing applications that have been written without the knowledge that a spatial field may be present. This method may however still be appropriate for new tables where there are no existing applications. The advantage of adding a spatial field directly to a table is that the triggers required to maintain the geometry are simpler.

Option two provides a method to spatially enable a table without affecting existing applications. The triggers required to keep the geometry field updated are slightly more complex than for option one. This is regarded as a small price to pay for isolating existing applications from changes to table structure. It was considered a good idea from a management point of view to keep the new spatial tables in the same schema as the tables being spatially enabled.

Option three is essentially the same as option two except that the spatial tables could be organised in a different schema to the tables being spatially enabled. This may be the best option if different staff were assigned to manage the spatial tables. It is however currently thought that managing the spatial tables alongside the tables that they spatially enable is a better option.

In conclusion, it has therefore been decided that option 2 currently provides the best model for enabling existing BGS tables.

3.3 MECHANISMS CONSIDERED FOR UPDATING GEOMETRY

Two mechanisms for keeping the geometry table updated were considered:

1) Using a trigger to track inserts, updates and deletes and to modify the contents of the geometry table appropriately.
2) Use a batch process to update the geometry table at a specified interval e.g. every night.

Option one provides the cleanest and simplest method of tracking changes. Once the trigger is in place all changes are immediately tracked and the spatial table is always up to date. There is little to go wrong with this method.

Option two provides an alternative method but is more complex and requires a method of determining the rows to be added, modified and deleted in the geometry table. This could be based on the contents of the date_entered, date_updated fields and on the deletion date from the associated history table. Such a mechanism would work until the batch job fails for some reason and updates to the geometry table are lost. A modified version of this option would be to update the geometry of all rows inserted, updated or deleted in the last 5 days. This would have the advantages that the batch job could fail up to four days in a row and the table would still be updated correctly on the fifth day. Another alternative would be track inserts and deletes using the contents of the primary key. Updates would still have to be tracked via the date_updated field:

Insert where Date_entered > sysdate –1

Update Where Date_updated > sysdate –1

Delete Where history_table.the_date > sysdate-1

An example of the code required for this option is given in Appendix 6.

3.4 TRIAL

The Single Onshore Borehole Index table (BGS.SOBI) was chosen as a test case. This table was spatially enabled using a separate geometry table and a trigger was added to the SOBI table to keep the geometry table updated. No problems were encountered during the first few days. An error was then encountered in an application that updated SOBI records. This error was not related to the particular application or to the spatial trigger but was the result of an internal Oracle error related to the insertion of a record into the spatial index. Similar errors have been reported on the Oracle forums and these relate to a deadlock situation when the R-Tree spatial index is being updated. It has been reported that this problem should not occur in Oracle version 10g. The spatial index was re-built and the trigger was re-written to reduce the likelihood of any deadlock situation arising from trigger itself (a delete immediately followed by an insert was converted to a single update statement). The trigger was re-enabled on the live database and to

date no further problems have occurred. It is recommended that we should wait until Oracle 10g is installed before we continue to spatially enable tables.

3.5 IMPACT ON GIS APPLICATIONS

The ability to have up to date spatial datasets based on our corporate Oracle tables is a great advantage over ESRI shapefile copies of the data that quickly become out of date.

ArcGIS and ArcView 3.3 clients both have the ability to use Oracle Spatial data that has been registered in SDE. There is however an issue with the ArcView3.3 version of the Geoscience Data Index in that some of the customisation does not support GeoDatabase themes. This can be rectified by further customisation but will require development time.

3.6 TYPES OF LOCATION DATA HELD IN EXISTING ORACLE TABLES

There are a number of different forms of location data that exist in current Oracle tables. The following list shows some examples of the main types of location data.

Locator Type	Examples	Comments
X-Y coordinates	205936,387654	This type of absolute spatial location is present in Oracle Tables that store point locations
National Grid Reference	TQ38762768	National grid references of this type cannot be directly used by GIS systems and have to be converted to numeric coordinates
Ordnance Survey tiles	TQ, TQ24NE, TQNENE, TQ1256 etc.	A location is implied by the tile name and it can be converted to a square polygon for use in GIS systems. This can either be done programmatically or by using a lookup table where the tile name and the pre calculated coordinates are stored.
Two Corner coordinates of rectangular areas	SWE=200000 SWN=300000 SEE=200520 SEN=300550	The storage of rectangular areas in this way occurs in a number of BGS tables where there was a requirement to capture areas using standard Oracle Interfaces such as Oracle Forms and Microsoft Access Applications. Such interfaces have no capability

		to capture complex line or polygon data, as a result only bounding rectangles have been captured.
Four corner coordinates of quadrilateral areas	SWE= 379115 SWN= 862079 SEE= 388766 SEN= 861941 NEE= 388858 NEN= 868375 NWE= 379207 NWN= 868513	The storage of location defined by quadrilateral areas in BGS tables has mainly been used in tables related to County Series maps. The reason for this is that County Series maps were created prior to the British National Grid and due to the projection and sheetlines used they are not aligned to the National Grid. They are best represented by non-grid aligned quadrilaterals.
There are some examples of Line and Polygon data being held in standard Oracle tables as a list of x-y coordinates identified by an Id and kept in the correct order by a Sequence identifier (SEQ).	ID,SEQ, ID,X,Y 1,1,365709,456384 1,2,456890,468345 1,3,469670,476758 2,1,256789,156734 2,1,345873,244567 …	This method of storing line data in Oracle is used for example to store Offshore tracks.

Examples of code that was created during the project to spatially enable some of these location types are listed in Appendix 5.

3.7 RECOMMENDATIONS FOR USE OF ORACLE SPATIAL

- BGS should proceed to spatially enable Oracle tables using Oracle Spatial/Locator.
- Spatial columns should be held in related tables if there are existing applications that might be compromised by adding a spatial column directly to an existing table.
- For new tables, spatial columns should be added directly for simplicity and performance.
- Triggers should be used in preference to batch updates to keep geometry synchronised with the locational attributes (e.g. easting and northing fields).
- Related spatial tables should be kept in the same schema as the tables to which they relate to make management easier.

4 Research into disconnected editing and versioning

4.1 DISCONNECTED EDITING

Disconnected editing is designed to allow a selected set of spatial features to be extracted from a master ESRI SDE geodatabase (Check-out), edited offline on a laptop or other portable device, and returned to the master database at a later date (Check-in). The editing is carried out in an environment that tracks the changes being made to the offline database (the Check-out database) thus allowing the changes to be applied back to the master database when offline editing and capture has been completed (long transaction).

Disconnected editing is based on ESRI's versioning technology and uses an optimistic approach to support long transactions. When data is extracted from the master database a version is created representing the data that has been checked out (this version should not be edited). The features and rows in the master geodatabase are not locked and can therefore still be edited by other users via the DEFAULT version. When previously checked-out features are checked back into the master database they are checked-in to the version in the master database that was created when the Check-out database was created. This version can then be integrated with the master version (normally the DEFAULT version). Conflicts where features have been edited both in the master database and in the checked out database need to be resolved at this stage. The operator has to decide which of the feature or attribute edits are correct. This process is called reconciling and posting.

Disconnected editing allows the editing and capture of spatial data in locations where it is not practical to maintain a live link back to a corporate database, for example when working in the field or in locations where network connections are poor or do not exist.

During testing disconnected editing worked well and provided an easy mechanism for the upload of data collected in personal geodatabases into a corporate geodatabase.

Potential uses of this technology in BGS are:

a) Field data collection
b) Geological Spatial Database (GSD)
c) AEGIS

Potential issues are:

a) Any changes to the master geodatabase schema, invalidates any Check-out databases that contain the feature classes or tables whose structure has changed, making them useless. All existing check-outs would have to be checked in before the master geodatabase schema could be altered.
b) Master database needs to be versioned. This is not necessarily a problem but it does mean extra management of versions and resolution of any conflicts. Also precludes any editing of the data using Oracle clients.

c) For new data capture, users cannot just take an empty copy of a personal geodatabase from a corporate drive and start populating it. A named empty check-out database needs to be created via the ArcMap Disconnected Editing toolbar or by using the Toolbox Check Out tool. A customised function could be created to automate the process.

4.2 VERSIONING

Versioning, allows multiple versions of spatial data to be held without replication, only the changes are stored. Users can edit the same features and rows without the need for traditional locking mechanisms. Edits are stored in so called delta tables. A particular version is constructed by merging the data held in the base tables along with data pertaining to that version from the delta tables. Thus a version is a bit like an Oracle View in that it gives the user a customised view of the data held within the database.

4.3 RECOMMENDATIONS FOR THE USE OF DISCONNECTED EDITING AND VERSIONING

- The best use of disconnected editing and versioning in BGS is in the area of field data capture. However since the current field data capture system uses a hybrid ArcGIS and Access solution, disconnected editing and versioning is not appropriate. It is therefore recommended that further research on the use of disconnected editing and versioning is not pursued unless the existing field capture system is going to be redesigned in a pure ESRI environment.

5 Remote Data Transfer

5.1 SECURE TOKEN AND EXTRANET

To allow users of the Mobile Integrated Data Acquisition System (MIDAS) to download their data back to the BGS in the evening it would be possible to either provide users with a Secure Token / digital certificate with which to access the BGS servers or allow them to download data via an extranet connection.

The use of a secure token would enable users to log on directly to the corporate BGS servers and copy their field database across to BGS every evening. This would then provide the extra security of a second backup of each field database and mean that the data would be stored within BGS should we wish to upload this data to ArcIMS / ArcGIS Server for other field geologists to view whilst out in the field. A secure token can be used with dial-up, broadband or GRPS Internet connection. Each secure token would cost £37.50 + VAT and would need a licence, which costs £73 +VAT. Each token lasts for 3 years but a licence is permanent. The cost of these tokens and licences would be borne by SNS.

Another option, that could replace the need for a secure token, could be the use of digital certificates. Digital certificates would allow users to log onto our secure servers by proving the identity of the client computer i.e. the tablet PC. This would then mean that it would be possible to log into our network using the normal windows id and password.

The Extranet option would allow users to access a customised application via an extranet connection. Whilst at present the exact function such an application could take has not been decided it is reasonable to assume that should we use an ArcGIS server application for viewing and accessing data whilst out in the field (see below) an extranet connection could also allow access to this application and provide the means of uploading and viewing field data in an ArcGIS server application.

It should be noted that there will be a time factor involved in any upload / download of data. The field database can range in size between approx 10 and 20 Mb. Working with a dial up connection of approx 30kb/s it could take roughly 1 - 2 hours to upload the data. Whilst this could be considered to be a long time, it is feasible that a geologist could leave this upload running whilst they do any other work they have to do in the evening.

5.2 ARCIMS AND ARCGIS SERVER

Once the field data has been transmitted back to the BGS it would then be possible to use this data within an ArcIMS or ArcGIS Server application to allow other users to view it whilst in the field. The purpose of this application would be to allow users to view data collected by other geologists such as their Field Observation Points, Geolines, Geopolys and Map Face Annotation. Geologists would be able to view the attributes of the features displayed on the map and this could potentially help them with their own interpretations whilst working in the field.

An ArcIMS application would allow users to view the basic feature classes populated by the MIDAS system (Field Observation Point, Geolines, Geopolys), however, the main drawback of this method of viewing data collected in the field is that ArcIMS can not handle annotation feature classes or related data very well. Within the MIDAS system the map face notes are stored as an annotation feature class and are an important part of the data collected in the field. It is therefore essential that they can be viewed in any application designed to allow geologists to view each others data. ArcIMS does not support annotation classes and the only way to display them in ArcIMS would be to convert them to a shapefile. This creates a polyline shapefile, containing an attribute with the annotation, but it does not export the line work associated to the annotation which shows the exact location the text refers to. In the MIDAS system users can move their annotation to the best location on their map but the annotation still maintains an arrow pointing to the exact location the text refers to. The fact that this location arrow is not maintained when the annotation is exported means this is not a practical solution for the display of the Map face notes. The ability to relate data is also an important feature for any application designed to allow users to view the data collected in the field. The majority of the detailed data collected in the field is held in tables related to the field observation point, so the inability to handle relates means that it would prove difficult to allow users in the field to view all the data collected at a field observation point. Whilst it is possible to program ArcIMS to handle one to many and many to many relationships it is highly likely that this would stretch ArcIMS to the limit and it might eventually prove impossible to replicate all the one to many relationships that would be needed.

An alternative to using ArcIMS would be to design an ArcGIS server application. An ArcGIS Server application could be set up to display the features collected in the field in the same way as would be possible in ArcIMS. However, ArcGIS server would also be able to display the map face notes annotation feature class in the same way as ArcMap does (i.e. preserving the link to the location the text refers to). Another advantage to using ArcGIS Server is that it is programmed using ArcObjects. This would mean that the application could easily be programmed to handle and display the related data that is held in the Field Database, therefore allowing geologists to view all the data collected at a particular field observation point. However, at present BGS staff have limited experience of developing ArcGIS server applications, so these would take slightly longer to develop. Another issue concerns the fact that ArcGIS server is currently not configured for external web use, it would be necessary to address this matter and make the server available across the internet before an ArcGIS server application could be used outside of BGS.

5.3 DOWNLOADING DATA WHILST IN THE FIELD

The ability to download, as opposed to viewing, data whilst the field geologist is out of the office would be of great use in situations where a geologist realises there is a need for a certain piece of data once they have left the office. A download data system could then be used to extract the required data from an internet based application. This system would be used to download existing BGS data and not copies of another geologists field data, as it is expected that the field geologists will only need to view each others data. Additionally due to the fact the Field Database has been developed primarily as an MS Access system and not as a geodatabase, it is currently not possible to extract data from this database using the data delivery extensions mentioned below. These data delivery extensions are designed to work with spatial data such as shapefiles or geodatabases.

5.3.1 ArcIMS – Data Delivery Extension

ArcIMS has a data delivery extension that allows users to zoom to their area of interest and download the data from the ArcIMS server. Once this extension is loaded onto our server it is simple to implement the functionality and the interface can be customized to meet our needs. The data is extracted from the main data and zipped up ready for download. The zip file can them be downloaded from the server.

5.3.2 ArcGIS Server Data Delivery

ArcGIS server can be customized to provide users with a similar data delivery system, however this is programmed into the system rather than a software extension provided by ESRI. As a result it is highly customizable and can be developed to provide exactly the required interface, download options and delivery options. The system in essence works in a similar way to the ArcIMS data Delivery extension with the user zooming to the area of interest then the applications will clip the data, create a geodatabase containing the data and zip up the geodatabase ready for downloading. (See section 2.14.5 for more information)

5.4 DATA TRANSFER VIA GPRS TO AND FROM TO BGS

To allow the transfer of data from a remote field location back to the BGS, and vice versa, it would be necessary to use a GPRS connection. GPRS works very well in urban areas where the GPRS signal is good. This is borne out by Westminster council's use of an ESRI developed system which transmits point data via GPRS. However, many of our field geologists currently work in areas where the GPRS signal is poor or non-existent (e.g the Highlands) and this would therefore make the transfer of data in these locations using this method impossible. Another consideration must also be the length of time and cost of transmitting data via GPRS. ESRI currently only transmit point data via this method as it is simple data and therefore quick and as a result cheap. The field data that is collected is a combination of point, line and polygon data. Due to the fact that the line and polygon data is more complex this would lead to a longer and more expensive upload / download time. It is unlikely that a field geologist will be prepared to wait for data to upload or download while they are out mapping. Time is of the essence to them and as a result if there is a time factor involved they would be more likely to wait until they were back at their field accommodation to transfer data. This would then make the use of GPRS unnecessary as once back in the field accommodation dial-up or broadband internet connections could be used.

5.5 RECOMMENDATIONS

It is definitely possible and practical to develop a system that would allow users to upload their data from the field accommodation for use in a viewing application. A viewing application would be of use to field geologists, allowing them to view data collected in other field areas that may be relevant to the area they are working in. With regard to developing a system that will allow users to download data whilst they are out of the office it is, again, possible to develop such a system and this would no doubt prove useful in situations where geologists require additional datasets once they are out of the office. However, more investigation should be done as to whether it is necessary to allow users to download another geologists actual field data. It is unlikely that this would be practical at present.

5.5.1 Transmitting data whilst in field accommodation

Any of the methods mentioned would be suitable for this purpose. Initially it would be very simple to provide each geologist with a secure token, which would immediately give them the

capability of transmitting their data back to the office. In the long run if an ArcGIS server application is built to view and receive data from the field then the extranet option would provide access to this application

5.5.2 Viewing data

Due to the capabilities of ArcGIS server and the fact that it can handle all the data we would need to allow users to view, this would prove the best option for a viewing application despite the fact it is not currently configured for external use. It would also be far more easily expanded in the future should more functionality be required. Whilst an ArcIMS application would provide some access to the data, and is already configured for external use, it would not allow access to all the data and this would reduce the usefulness of the application. The fact that the ArcGIS server is not configured for external use should not be considered as a problem. To date there has not been a need to have external access to the server, therefore, whilst there will be matters to be addressed it is not foreseen that these will prevent the server going external.

5.5.3 Downloading of data

Both ArcIMS and ArcGIS server have the functionality to allow users to download data from an area of interest. ESRI state that the ArcGIS server functionality is better than the ArcIMS functionality. Taking into consideration the fact that the most practical solution to viewing the data would be to develop an ArcGIS server application it would also therefore dictate that the ArcGIS server download capabilities should be used.

5.5.4 Downloading and uploading of data whilst away from field accommodation

Due to the current situation with regards to GPRS coverage in a number of areas where BGS carries out field mapping it is not currently practical to recommend to geologists that they transmit data away from their field accommodation. Many geologists will encounter situations when they are unable to transmit or receive data due to a lack of GPRS coverage and will therefore be unable to use the system. There is also the fact that those geologists who are in areas of GPRS coverage will encounter waiting times whilst data transmits. A system which allows them to upload and download data in their field accommodation is far more practical and more likely to be used than a system which, when it is available, requires a waiting time whilst out in the field.

Glossary

ADF	Application Development Framework
API	Application Programming Interface
ASP	Active Server Pages
COM	Component Object Model
DLL	Dynamic Link Library
JSP	Java Server Pages
GDI	GeoSpatial Data Index
GSD	Geological Spatial Database
IASC	Information Architecture Steering Committee
IIS	Internet Information Services
SDE	Spatial Data Engine
SOAP	Simple Object Access Protocol
SOBI	Single Onshore Borehole Index
SOC	Server Object Container
SOM	Server Object Manager
VB	Visual Basic
VBA	Visual Basic for Applications
VB .NET	Visual Basic .NET
WSDL	Web Services Description Language
XML	Extensible Markup Language

References

Bader, E., Cameron, E., Davies, C., Gill, S., Jones, S., MacDonald, A., Meister, G., Minami, M., O'Neill, D., Reuland, A., Singh, R., Van Esch, S., Yu, Z. 2004. *ArcGIS Server Administrator and Developer Guide)*. (Redlands: ESRI.) ArcGIS 9.0

Halvorson, M. 2003. *Microsoft Visual Basic .NET Step by Step: Version 2003*. Microsoft Press

Murray, C. 2002. *Oracle Spatial User's Guide and Reference Release 9.2*. (Oracle Corporation.) Part No. A96630-01

Russell, J. 2003. *PL/SQL User's Guide and Reference 10g Release1(10.1)*. (Oracle Corporation.) Part No. B10807-01

Walther, S. 2003. *ASP.NET Kick Start*. Sams Publishing

ColdFusion MX 7 CFML Reference. Macromedia (2005)

ColdFusion MX 7 Developers Guide. Macromedia (2005)

Web sites

ESRI (www.esri.com)

W3C Consortium (www.w3c.org)

Web services architect (www.webservicesarchitect.com)

Appendix 1 ArcGIS Server Example Web Service

In order to gain familiarity with the implementation of Web services within ArcGIS Server, the example scenario outlined in chapter 7 of the ArcGIS Server Administrator and Developer Guide was implemented. This Web service finds all toxic waste sites within a certain distance of an address. It expects three input parameters (an address, a zip code and a distance) and returns an array of application defined toxic waste site objects. The example was implemented in .NET, but could just as easily have been implemented using Java.

CONSUMING THE EXAMPLE WEB SERVICE

Having created the Web service, the next step was to see how easy it was to consume its functionality within development environments used by the BGS. This example .NET Web service was consumed in ColdFusion. (Later in this report we will describe the generation of a BGS specific Web service within ArcGIS Server and its consumption in both ColdFusion and Java.)

The first step in consuming the Web Service in ColdFusion was to analyse the Web Service Description Language (WSDL) of the Web service. This describes the methods and types exposed by the Web service to the consumer. The WSDL for the example Web service is available at http://kwntsarcgis/ToxicLocations/ToxicLocations.asmx?WSDL and is given below.

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:s="http://www.w3.org/2001/XMLSchema"
     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:tns="http://kwntsarcgis/Webservices/"
     xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
     xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
     targetNamespace="http://kwntsarcgis/Webservices/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
   - <wsdl:types>
     - <s:schema elementFormDefault="qualified"
          targetNamespace="http://kwntsarcgis/Webservices/">
        - <s:element name="FindToxicLocations">
          - <s:complexType>
            - <s:sequence>
                 <s:element minOccurs="0" maxOccurs="1" name="Address"
                    type="s:string" />
                 <s:element minOccurs="0" maxOccurs="1" name="ZipCode"
                    type="s:string" />
                 <s:element minOccurs="1" maxOccurs="1" name="Distance"
                    type="s:double" />
              </s:sequence>
            </s:complexType>
          </s:element>
        - <s:element name="FindToxicLocationsResponse">
          - <s:complexType>
            - <s:sequence>
                 <s:element minOccurs="0" maxOccurs="1"
                    name="FindToxicLocationsResult"
                    type="tns:ArrayOfToxicSite" />
              </s:sequence>
            </s:complexType>
          </s:element>
        - <s:complexType name="ArrayOfToxicSite">
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="unbounded"
                 name="ToxicSite" nillable="true" type="tns:ToxicSite" />
```

```
            </s:sequence>
          </s:complexType>
        - <s:complexType name="ToxicSite">
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="Name"
                 type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="Type"
                 type="s:string" />
              <s:element minOccurs="1" maxOccurs="1" name="X"
                 type="s:double" />
              <s:element minOccurs="1" maxOccurs="1" name="Y"
                 type="s:double" />
            </s:sequence>
          </s:complexType>
        </s:schema>
    </wsdl:types>
  - <wsdl:message name="FindToxicLocationsSoapIn">
      <wsdl:part name="parameters" element="tns:FindToxicLocations" />
    </wsdl:message>
  - <wsdl:message name="FindToxicLocationsSoapOut">
      <wsdl:part name="parameters" element="tns:FindToxicLocationsResponse" />
    </wsdl:message>
  - <wsdl:portType name="ToxicSiteLocatorSoap">
    - <wsdl:operation name="FindToxicLocations">
        <wsdl:input message="tns:FindToxicLocationsSoapIn" />
        <wsdl:output message="tns:FindToxicLocationsSoapOut" />
      </wsdl:operation>
    </wsdl:portType>
  - <wsdl:binding name="ToxicSiteLocatorSoap" type="tns:ToxicSiteLocatorSoap">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
         style="document" />
    - <wsdl:operation name="FindToxicLocations">
        <soap:operation
           soapAction="http://kwntsarcgis/Webservices/FindToxicLocations"
           style="document" />
      - <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
      - <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
  - <wsdl:service name="ToxicSiteLocator">
    - <wsdl:port name="ToxicSiteLocatorSoap" binding="tns:ToxicSiteLocatorSoap">
        <soap:address
           location="http://kwntsarcgis/ToxicLocations/ToxicLocations.asmx" />
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

A detailed examination of the WSDL is beyond the scope of the report, but the important parts relevant to our needs are summarised. The main method of the Web service is FindToxicLocations. This expects an input of a complex data type called FindToxicLocations, which is made up of a string representing an address, a string representing the zip code and a double representing the distance to search from the address. The FindToxicLocations method returns the complex data type FindToxicLocationsResult, which is an array of toxicSites, made up of a name string, a type string and an X and Y coordinate (both doubles).

After examining the WSDL, our next task was to create a simple Web form in which a user can enter details of an address, zip code and distance in order to test the Web service. The code for this simple form is given below with example inputs.

```
<form method="post" action="consume.cfm">
Address: <input type="text" size="20" name="address"> 2111 Division St<br /><br />
ZipCode: <input type="text" size="20" name="zipcode"> 97202<br /><br />
Distance: <input type="text" size="20" name="distance"> 10000<br /><br />
<input type="submit" value="Get toxic sites">
</form>
```

The ColdFusion page consume.cfm then takes this input and communicates with the Web service. This input need not have been entered by a user via a Web form, but could have been generated programmatically by another part of the system.

There are a number of ways to interact with Web services in ColdFusion. The page consume.cfm uses cfscript and is given below.

```
<cfscript>
   ws = CreateObject("Webservice",
            "http://kwntsarcgis/ToxicLocations/ToxicLocations.asmx?WSDL");
   ToxicSites = ws.FindToxicLocations("#form.address#", "#form.zipcode#", #form.distance#);
</cfscript>

<h3>Toxic site locations</h3>

<cfoutput>
<table border="1" cellpadding="4" cellspacing="0"><tr>
<td><b>Name</b></td>
<td><b>Type</b></td>
<td><b>X</b></td>
<td><b>Y</b></td>
</tr>
<CFLOOP index="i" from="0" to="#evaluate(ArrayLen(ToxicSites.getToxicSite())-1)#">
      <tr>
      <td>#evaluate("ToxicSites.getToxicSite(#i#).Name")#</td>
      <td>#evaluate("ToxicSites.getToxicSite(#i#).Type")#</td>
      <td>#evaluate("ToxicSites.getToxicSite(#i#).X")#</td>
      <td>#evaluate("ToxicSites.getToxicSite(#i#).Y")#</td>
      </tr>
</CFLOOP>
</table>
</cfoutput>
```

Firstly referencing the Web service's WSDL creates a Web service object. The FindToxicLocations function is then called, passing the values entered into the form by the user.

This gives us our array of toxic sites objects as shown below.

object of kwntsarcgis.Webservices.ArrayOfToxicSite	
Methods	hashCode (returns int) equals (returns boolean) getSerializer (returns interface org.apache.axis.encoding.Serializer) getDeserializer (returns interface org.apache.axis.encoding.Deserializer) getTypeDesc (returns org.apache.axis.description.TypeDesc) getToxicSite (returns kwntsarcgis.Webservices.ToxicSite) getToxicSite (returns [Lkwntsarcgis.Webservices.ToxicSite;) setToxicSite (returns void) setToxicSite (returns void) getClass (returns java.lang.Class) wait (returns void) wait (returns void) wait (returns void) notify (returns void) notifyAll (returns void) toString (returns java.lang.String)

We then use the getToxicSite function to iterate through the array object in order to extract the details of the sites.

The functionality for obtaining the toxic sites is done entirely in ArcGIS server. ColdFusion simply accepts the results returned from ArcGIS Server and presents them to the user.

A screen shot of the initial form and the results extracted by ArcGIS Server are given below.

Address: 2111 Division St 2111 Division St

ZipCode: 97202 97202

Distance: 10000 10000

Get toxic sites

Toxic site locations

Name	Type	X	Y
EAST SIDE PLATING INC PLANT 4	Hazardous waste generator	7647860.94569	679162.182541
Portland Office of Transportation	Brownfield Pilot	7646057.61595	684318.663502
Portland Office of Transportation	Brownfield Pilot	7646057.61595	684318.663502
TRI MET CENTER STREET GARAGE	Hazardous waste generator	7651285.44055	672416.229791

Appendix 2 Building a BGS GeoSure Web Service using ArcGIS Server

Experiments with the sample ArcGIS Server Web service proved the concept of integrating GIS functionality into ColdFusion applications. The next stage was to develop a BGS example. GeoSure seemed an ideal candidate and a Web service was developed in ArcGIS Server that expected an input of an easting and a northing and returned a value of A-E for each of the six GeoSure hazards for that location. As part of the investigations into ArcGIS Server, an application had already been developed that performed this same functionality. This provided an opportunity to see how quickly an existing piece of functionality could be converted into a Web service and consumed by an external application.

It took approximately one hour to convert the existing GeoSure application into a Web service and to write a ColdFusion application to consume its functionality. It is appreciated that the structure of the Web service generated might not be the most elegant, but the concept was successfully proven for the purposes of this research and development exercise.

WSDL

The WSDL describing the Web service is available at:

http://kwntsarcgis/GeoSureWebService/GeoSureWebService.asmx?WSDL and is given below:

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://tempuri.org/GeoSureWebService/GetGeoSureValues"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://tempuri.org/GeoSureWebService/GetGeoSureValues"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  - <wsdl:types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://tempuri.org/GeoSureWebService/GetGeoSureValues">
      - <s:element name="GetGeoSureValues">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="X"
                 type="s:double" />
              <s:element minOccurs="1" maxOccurs="1" name="Y"
                 type="s:double" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="GetGeoSureValuesResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1"
                 name="GetGeoSureValuesResult" type="tns:GeoSureScores"
                 />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:complexType name="GeoSureScores">
        - <s:sequence>
```

```
            <s:element minOccurs="0" maxOccurs="1" name="ShrinkSwell"
               type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="Compressible"
               type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="Collapsible"
               type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="RunningSand"
               type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="Dissolution"
               type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="SlopeInstability"
               type="s:string" />
         </s:sequence>
      </s:complexType>
   </s:schema>
 </wsdl:types>
- <wsdl:message name="GetGeoSureValuesSoapIn">
   <wsdl:part name="parameters" element="tns:GetGeoSureValues" />
 </wsdl:message>
- <wsdl:message name="GetGeoSureValuesSoapOut">
   <wsdl:part name="parameters" element="tns:GetGeoSureValuesResponse" />
 </wsdl:message>
- <wsdl:portType name="GetGeoSureValuesSoap">
  - <wsdl:operation name="GetGeoSureValues">
      <wsdl:input message="tns:GetGeoSureValuesSoapIn" />
      <wsdl:output message="tns:GetGeoSureValuesSoapOut" />
   </wsdl:operation>
 </wsdl:portType>
- <wsdl:binding name="GetGeoSureValuesSoap" type="tns:GetGeoSureValuesSoap">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
      style="document" />
  - <wsdl:operation name="GetGeoSureValues">
      <soap:operation
         soapAction="http://tempuri.org/GeoSureWebService/GetGeoSureValues/Get
         GeoSureValues" style="document" />
     - <wsdl:input>
         <soap:body use="literal" />
      </wsdl:input>
     - <wsdl:output>
         <soap:body use="literal" />
      </wsdl:output>
   </wsdl:operation>
 </wsdl:binding>
- <wsdl:service name="GetGeoSureValues">
  - <wsdl:port name="GetGeoSureValuesSoap" binding="tns:GetGeoSureValuesSoap">
      <soap:address
         location="http://kwntsarcgis/GeoSureWebService/GeoSureWebService.asmx
         " />
   </wsdl:port>
 </wsdl:service>
          </wsdl:definitions>
```

The WSDL explains that the Web service has a method called getGeosureValues that expects to be passed an easting and a northing. This method then returns a value for each of the six GeoSure hazards.

COLDFUSION CONSUMER

Two ColdFusion pages were again written to test the Web service. The first is a simple form that allows the user to enter an easting and a northing. The code for this is shown below:

```
<form method="post" action="usegeosure_arcserver.cfm">
Eastings: <input type="text" size="20" name="eastings"><br /><br />
Northings: <input type="text" size="20" name="northings"><br /><br />
<input type="submit" value="Get geosure report">
</form>
```

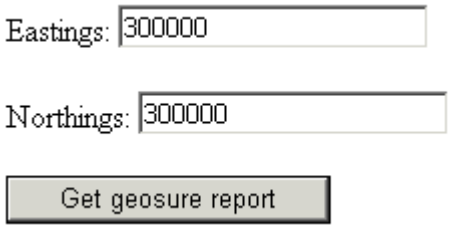
This form then posts to a second ColdFusion script that consumes the Web service and displays the GeoSure data for that location. This script is shown below.

```
<cfscript>
   ws = CreateObject("Webservice",
"http://kwntsarcgis/GeoSureWebService/GeoSureWebService.asmx?WSDL");
   hazards = ws.getGeosureValues(form.eastings,form.northings);
</cfscript>

<cfoutput>
<h4>Geosure Hazards</h4>

Shrink/Swell - #hazards.getShrinkSwell()#<br />
<br />
Dissolution - #hazards.getDissolution()#<br />
<br />
Slope - #hazards.getSlopeInstability()#<br />
<br />
Compressible - #hazards.getCompressible()#<br />
<br />
Running sand - #hazards.getRunningSand()#<br />
<br />
Collapsibles - #hazards.getCollapsible()#<br />
<br />
```

Once again, cfscript is used to make a Web service object by referencing the WSDL. The getGeosureValues method is then passed the easting and northing entered by the user. The object returned by the Web service is shown below.

object of org.tempuri.GeoSureWebService.GetGeoSureValues.GeoSureScores	
Methods	hashCode (returns int) equals (returns boolean) getSerializer (returns interface org.apache.axis.encoding.Serializer) getDeserializer (returns interface org.apache.axis.encoding.Deserializer) getTypeDesc (returns org.apache.axis.description.TypeDesc) getShrinkSwell (returns java.lang.String) getDissolution (returns java.lang.String) getSlopeInstability (returns java.lang.String) getCompressible (returns java.lang.String) getRunningSand (returns java.lang.String) getCollapsible (returns java.lang.String) setShrinkSwell (returns void) setCompressible (returns void) setCollapsible (returns void) setRunningSand (returns void) setDissolution (returns void) setSlopeInstability (returns void) getClass (returns java.lang.Class) wait (returns void) wait (returns void) wait (returns void) notify (returns void) notifyAll (returns void) toString (returns java.lang.String)

The various get methods for each Geosure hazard indicated in the object are then used to obtain the various hazard scores. Screen shots of the two ColdFusion pages are shown below.



JAVA CONSUMER

An example client was also developed in Java to demonstrate consumption of ArcGIS Server Web services in another BGS development environment. As ColdFusion is in fact built on top of Java, the two methods are very similar behind the scenes, though the work required from the developer is different.

Consuming Web services within Java applications relies on the Apache Axis libraries. The WSDL2Java tool can then be used to help generate the code required for your Web service consumer. Assuming the Apache Axis libraries have been downloaded and installed, invoke the WSDL2Java tool (org.apache.axis.wsdl.WSDL2Java) using a command similar to the following in order to generate the Java code necessary to wrap the objects sent back and expected by the Web service:

```
java org.apache.axis.wsdl.WSDL2Java wsdlurl
```

Verbose example:

```
java -classpath <path>\axis.jar;<path>\commons-logging-
1.0.4.jar;<path>\common
s-discovery-0.2.jar;<path>\jaxrpc.jar;<path>\saaj.jar;<path>\wsdl4j-
1.5.1.jar org.apache.axis.wsdl.WSDL2Java http://wsdl-url?wsdl
```

In order for this command to work, all of the above libraries must be on the system classpath; in the case of the command above they have been added explicitly to the command using the –

classpath option. Note that the only necessary command line argument to the WSDL2Java tool is the URL of the WSDL.

The WSDL2Java tool generates the Java code you need to easily build a Java client for the Web service. Before you can use this code you need to compile it. For a .NET service the usual package namespace begins org.tempuri. The remainder of the package namespace will depend on the Web service WSDL itself. In the ArcGIS Server example at:

http://kwntsarcgis/GeoSureWebService/GeoSureWebService.asmx?WSDL

The package created was as follows (note how this is the same as the ColdFusion object returned): org.tempuri.GeoSureWebService.GetGeoSureValues

and the following classes were generated within that package:

GeoSureScores.java

GetGeoSureValues_Service.java

GetGeoSureValues_ServiceLocator.java

GetGeoSureValuesSoap.java

GetGeoSureValuesSoapStub.java

The WSDL2Java tool will generate this as a directory structure within the directory from which you invoked the command. To compile the package, issue a javac command similar to the following:

```
javac *.java
```

Verbose example:

```
<path>\org\tempuri\GeoSureWebService\GetGeoSureValues>javac -class
path <path>\axis.jar;<path>\common
s-logging-1.0.4.jar;<path>\commons-discovery-
0.2.jar;<path>\jaxrpc.jar;<path>\saaj.jar;<path>\wsdl4j-1.5.1.jar *.java
```

Note again the need for the Axis libraries on the classpath.

Once compiled, the GetGeoSureValues package is ready to use in a Java Web service client. Below is a complete example of the code used to test the ArcServer GeoSure Web service:

```
package uk.ac.bgs.Webservices;

// axis.jar
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;

// jaxrpc.jar
import javax.xml.namespace.QName;

// .NET stubs/objects package as generated by WSDL2Java command
import org.tempuri.GeoSureWebService.GetGeoSureValues.*;

/**
 * Example ArcServer "GeoSure" Web service client for .NET WSDL URL...
 * http://kwntsarcgis/GeoSureWebService/GeoSureWebService.asmx?WSDL
 * @author Ben Wood, BGS (NERC), 2005
 */
public class ArcServerTest
  {

  public static void main(String args[])
    {
    double easting = Double.parseDouble(args[0]);
    double northing = Double.parseDouble(args[1]);

    ArcServerTest test = new ArcServerTest();
    test.queryGeoSure(easting, northing);
    }

  // Test method to query GeoSure for a particular [easting, northing]
  private void queryGeoSure(double easting, double northing)
    {
    try
      {
      // Make a service
      GetGeoSureValues_Service service = new GetGeoSureValues_ServiceLocator();
      // Use service to get a stub
      GetGeoSureValuesSoap hazardousSoap = service.getGetGeoSureValuesSoap();
      // Call business method on Web service to obtain data object
      GeoSureScores hazards = hazardousSoap.getGeoSureValues(easting, northing);

      // Interrogate data object to obtain real values and print to validate
      System.out.println("Shrink swell: " + hazards.getShrinkSwell());
      System.out.println("Compressible: " + hazards.getCompressible());
      System.out.println("Collapsible: " + hazards.getCollapsible());
      System.out.println("Running sand: " + hazards.getRunningSand());
      System.out.println("Dissolution: " + hazards.getDissolution());
      System.out.println("Slope instability: " + hazards.getSlopeInstability());

      }
    catch(Exception e)
      {
      System.err.println("Problem in queryGeoSure(): " + e);
      }
    }


  }
```

As you would hope, the ColdFusion and the Java client return the same GeoSure values. It is evident that there is more work to do in generating the Java client, but it is still in the order of just 30 minutes for a Java developer to create an application that consumes the ArcGIS Server Web service.

Appendix 3 GeoSure Web service using ColdFusion and ArcIMS

As with ArcGIS Server, a ColdFusion application already existed that took an easting and a northing as input and, by referencing ArcIMS, gave back a hazard value for each of the GeoSure hazards. Again it took little more than an hour to convert the existing code to a Web service and to write a simple application to consume it. Web services are published in ColdFusion by repackaging the existing code as components. For more details on publishing Web services using ColdFusion, please refer to the ColdFusion documentation.

The resulting WSDL is available at http://kwp188202/scripts/Webservices/geosure.cfc?wsdl and is given below.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://Webservices.scripts"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://Webservices.scripts" xmlns:intf="http://Webservices.scripts"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns1="http://rpc.xml.coldfusion"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <!--
  WSDL created by Macromedia ColdFusion MX version 7,0,0,91690
    -->
  - <wsdl:types>
    - <schema targetNamespace="http://xml.apache.org/xml-soap"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://rpc.xml.coldfusion" />
        <import namespace="http://Webservices.scripts" />
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      - <complexType name="mapItem">
        - <sequence>
            <element name="key" nillable="true" type="xsd:anyType" />
            <element name="value" nillable="true" type="xsd:anyType" />
          </sequence>
        </complexType>
      - <complexType name="Map">
        - <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="item"
              type="apachesoap:mapItem" />
          </sequence>
        </complexType>
      </schema>
    - <schema targetNamespace="http://rpc.xml.coldfusion"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://xml.apache.org/xml-soap" />
        <import namespace="http://Webservices.scripts" />
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      - <complexType name="CFCInvocationException">
          <sequence />
        </complexType>
      - <complexType name="QueryBean">
        - <sequence>
            <element name="columnList" nillable="true"
              type="impl:ArrayOf_xsd_string" />
            <element name="data" nillable="true"
              type="impl:ArrayOfArrayOf_xsd_anyType" />
          </sequence>
        </complexType>
      </schema>
    - <schema targetNamespace="http://Webservices.scripts"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://rpc.xml.coldfusion" />
```

```
        <import namespace="http://xml.apache.org/xml-soap" />
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      - <complexType name="ArrayOf_xsd_string">
       - <complexContent>
          - <restriction base="soapenc:Array">
              <attribute ref="soapenc:arrayType"
                 wsdl:arrayType="xsd:string[]" />
            </restriction>
          </complexContent>
        </complexType>
      - <complexType name="ArrayOfArrayOf_xsd_anyType">
       - <complexContent>
          - <restriction base="soapenc:Array">
              <attribute ref="soapenc:arrayType"
                 wsdl:arrayType="xsd:anyType[][]" />
            </restriction>
          </complexContent>
        </complexType>
      </schema>
    </wsdl:types>
  - <wsdl:message name="CFCInvocationException">
      <wsdl:part name="fault" type="tns1:CFCInvocationException" />
    </wsdl:message>
  - <wsdl:message name="getGeosureValueResponse">
      <wsdl:part name="getGeosureValueReturn" type="apachesoap:Map" />
    </wsdl:message>
  - <wsdl:message name="getGeosureValueRequest">
      <wsdl:part name="eastings" type="xsd:double" />
      <wsdl:part name="northings" type="xsd:double" />
    </wsdl:message>
  - <wsdl:portType name="geosure">
     - <wsdl:operation name="getGeosureValue" parameterOrder="eastings
         northings">
        <wsdl:input message="impl:getGeosureValueRequest"
           name="getGeosureValueRequest" />
        <wsdl:output message="impl:getGeosureValueResponse"
           name="getGeosureValueResponse" />
        <wsdl:fault message="impl:CFCInvocationException"
           name="CFCInvocationException" />
      </wsdl:operation>
    </wsdl:portType>
  - <wsdl:binding name="geosure.cfcSoapBinding" type="impl:geosure">
      <wsdlsoap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http" />
     - <wsdl:operation name="getGeosureValue">
        <wsdlsoap:operation soapAction="" />
       - <wsdl:input name="getGeosureValueRequest">
          <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://Webservices.scripts" use="encoded" />
        </wsdl:input>
       - <wsdl:output name="getGeosureValueResponse">
          <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://Webservices.scripts" use="encoded" />
        </wsdl:output>
       - <wsdl:fault name="CFCInvocationException">
          <wsdlsoap:fault
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             name="CFCInvocationException"
             namespace="http://Webservices.scripts" use="encoded" />
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>
  - <wsdl:service name="geosureService">
     - <wsdl:port binding="impl:geosure.cfcSoapBinding" name="geosure.cfc">
        <wsdlsoap:address
           location="http://kwp188202/scripts/Webservices/geosure.cfc" />
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

Again a simple form based front end and a simple ColdFusion application were developed to consume the Web service. The code and screen shots are given below.

```
<form method="post" action="usegeosure_arcims2.cfm">

Eastings: <input type="text" size="20" name="eastings"><br /><br />
Northings: <input type="text" size="20" name="northings"><br /><br />
<input type="submit" value="Get geosure report">

</form>

<cfscript>
   ws = CreateObject("Webservice", "http://kwp188202/scripts/Webservices/geosure.cfc?wsdl");
   hazards = ws.getGeosureValue(form.eastings,form.northings);
</cfscript>

<cfoutput>
<h4>Geosure Hazards</h4>
Shrink/Swell - #hazards.shrink#<br /><br />
Dissolution - #hazards.diss#<br /><br />
Slope - #hazards.slope#<br /><br />
Compressibles - #hazards.comp#<br /><br />
Running sand - #hazards.run#<br /><br />
Collapsibles - #hazards.coll#<br /><br />
</cfoutput>
```

Eastings: 300000

Northings: 300000

Get geosure report

Geosure Hazards

Shrink/Swell - A

Dissolution - No Score

Slope - B

Compressible - A

Running sand - A

Collapsibles - No Score

Appendix 4 Geoprocessing in ArcGIS Server

This Appendix contains sample code for performing GeoProcessing in an ArcGIS Server Web application using the fine-grained ArcObjects class IBasicGeoprocessor. The code below performs a Union operation.

```
Private Sub doUnion()

Dim webmap As WebMap = Map1.CreateWebMap
Try
   Dim mapServer As IMapServer = webmap.MapServer
   Dim mapDescription As IMapDescription = WebMap.MapDescription
   Dim mapName As String = mapDescription.Name
   Dim mso As IMapServerObjects = CType(mapServer, IMapServerObjects)
   Dim sc As ServerContext = webmap.ServerContext

   Dim pLayer As IFeatureLayer = mso.Layer(mapName, 0)

   ' Get the first layer's table
   ' Use the Itable interface from the Layer (not from the FeatureClass)
   ' This table defines which fields are to be used in the output
   Dim pFirstTable As ITable
   pFirstTable = pLayer
   pLayer = mso.Layer(mapName, 1)
   Dim pSecondTable As ITable
   pSecondTable = pLayer
   ' Error checking
   If pFirstTable Is Nothing Then Exit Sub
   If pSecondTable Is Nothing Then Exit Sub

   ' Define the output feature class name and shape type
   Dim pFeatClassName As IFeatureClassName
   pFeatClassName = sc.CreateObject("esriGeoDatabase.FeatureClassName")
   With pFeatClassName
      .FeatureType = esriFeatureType.esriFTSimple
      .ShapeFieldName = "Shape"
      .ShapeType = esriGeometryType.esriGeometryPolygon
   End With

   'Set the output location and feature class name
   Dim pNewWSName As IWorkspaceName
   pNewWSName = sc.CreateObject("esriGeoDatabase.WorkspaceName")
   With pNewWSName
     .WorkspaceFactoryProgID = esriDataSourcesFile.ShapefileWorkspaceFactory.1"
     .PathName = "D:\apma\ArcGISServer\GeoProcessing\"
   End With

   Dim pDatasetName As IDatasetName
   pDatasetName = pFeatClassName
   pDatasetName.Name = "Output3"
   pDatasetName.WorkspaceName = pNewWSName

   ' Perform the merge
   Dim pBGP As IBasicGeoprocessor
   pBGP = sc.CreateObject("esriCarto.BasicGeoprocessor")
   Dim pOutputFeatClass As IFeatureClass
   pOutputFeatClass = pBGP.Union(pFirstTable, False, pSecondTable, False, 0, pFeatClassName)
   addLayer(sc, pOutputFeatClass, "Union_output")
   webmap.Refresh()
   Toc1.Draw(True)
Finally
   webMap.Dispose()
End Try

End Sub
```

Appendix 5 Spatially Enable Location Data

EXAMPLE CODE TO SPATIALLY ENABLE TABLES WITH X-Y COORDINATES

```
-------------------------------------------------------
-- Name: Points_Related
-- Purpose: To Spatially enable an existing Oracle Table
--          containing X and Y coordinates using a related
--          spatial table.
-- Written: K. Adlam, 20/5/05, updated 24/5/05, 1/9/05
-------------------------------------------------------
--
-------------------------------------------------------
-- Clear variables
-------------------------------------------------------
undefine existing_table;
undefine east;
undefine north;
undefine keyfield;
-------------------------------------------------------
-- CREATE GEOMETRY TABLE
-------------------------------------------------------
create table &&existing_table._SP(
fid number(38) primary key,
shape mdsys.sdo_geometry)
TABLESPACE TS3A
/
-------------------------------------------------------
-- CREATE VIEW
-------------------------------------------------------
CREATE VIEW &&existing_table._SV AS SELECT b.*,a.*
FROM &&existing_table._sp A, &&existing_table B
WHERE A.FID=B.&&keyfield
/
-------------------------------------------------------
-- Initial insert
-------------------------------------------------------
INSERT INTO &&existing_table._sp SELECT &&keyfield,
DECODE(
DECODE(&&east,null,'A','B')||DECODE(&&north,null,'A','B')
,'BB',
mdsys.sdo_geometry(2001,81989,mdsys.sdo_POINT_TYPE(&&east,&&north,Null),Null,Null)
,NULL)
FROM &&existing_table
/
--INSERT INTO &&existing_table._sp SELECT &&keyfield, mdsys.sdo_geometry
--(2001,81989,mdsys.sdo_POINT_TYPE(&&east,&&north,Null),Null,Null)
--FROM &&existing_table
--WHERE &&east IS NOT NULL and &&north IS NOT NULL
--/
--INSERT INTO &&existing_table._sp SELECT &&keyfield, NULL
--FROM &&existing_table
--WHERE &&east IS NULL and &&north IS NULL
--/
COMMIT
/
-------------------------------------------------------
-- CREATE TRIGGER
-------------------------------------------------------
create or replace trigger &&existing_table._SPT
after insert or update or delete on &&existing_table
/*
 Trigger to maintain related geometry table
 Inserts, updates and deletes are tracked and
 the spatial table is modified appropriately.

 Created by Keith AM Adlam 20/10/2005
*/
```

```
for each row

DECLARE
  ddpoint mdsys.sdo_geometry;

BEGIN

   IF DELETING THEN
     DELETE FROM KAMA.&&existing_table._SP WHERE &&existing_table._SP.fid = :old.&&keyfield;
   END IF;

   IF INSERTING THEN
     IF (:new.&&east IS NULL OR :new.&&north IS NULL)THEN
       INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,Null);
     ELSE
       ddpoint:=mdsys.sdo_geometry
       (2001,81989,mdsys.sdo_POINT_TYPE(:new.&&east,:new.&&north,Null),Null,Null);
       INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,ddpoint);
     END IF;
   END IF;

   IF UPDATING THEN
     IF (:new.&&east != :old.&&east or :new.&&north != :old.&&north)THEN
       IF (:new.&&east IS NULL OR :new.&&north IS NULL)THEN
         UPDATE KAMA.&&existing_table._SP Set SHAPE = NULL where FID = :new.&&keyfield;
       ELSE
         ddpoint:=mdsys.sdo_geometry
         (2001,81989,mdsys.sdo_POINT_TYPE(:new.&&east,:new.&&north,Null),Null,Null);
         UPDATE KAMA.&&existing_table._SP Set SHAPE = ddpoint where FID = :new.&&keyfield;
       END IF;
     END IF;

     --DELETE FROM KAMA.&&existing_table._SP WHERE &&existing_table._SP.fid = :new.&&keyfield;
     --IF (:new.&&east IS NULL OR :new.&&north IS NULL)THEN
     --  INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,Null);
     --ELSE
     --  ddpoint:=mdsys.sdo_geometry
     --  (2001,81989,mdsys.sdo_POINT_TYPE(:new.&&east,:new.&&north,Null),Null,Null);
     --  INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,ddpoint);
     --END IF;
   END IF;

END &&existing_table._SPT;
/
-------------------------------------------------------------
-- CREATE METADATA FOR SPATIAL TABLE
-------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SP', 'SHAPE',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);

-------------------------------------------------------------
-- CREATE METADATA FOR VIEW
-------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SV', 'SHAPE',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);

-------------------------------------------------------------
-- CREATE SPATIAL INDEX
-------------------------------------------------------------
create index &&existing_table.SP_SI on &&existing_table._SP(shape)
indextype is mdsys.spatial_index
PARAMETERS ('SDO_INDX_DIMS = 2 TABLESPACE = TS3B')
/
-------------------------------------------------------------
-- GRANTS on TABLES and VIEW
-------------------------------------------------------------
```

```
grant select on &&existing_table._SP to public
/
grant select on &&existing_table._SV to public
/
-------------------------------------------------------------
-- Clear variables
-------------------------------------------------------------
undefine existing_table;
undefine east;
undefine north;
undefine keyfield;
-------------------------------------------------------------
```

EXAMPLE CODE TO SPATIALLY ENABLE TABLES WITH RECTANGLES

```
-------------------------------------------------------
-- Name: Rectangles_Related
-- Purpose: To Spatially enable an existing Oracle Table
--          containing rectangle coordinates as
--          xmin,ymin,xmax,ymax
--          (often SWE,SWN,NEE,NEN in BGS tables)
--          using a related spatial table.
-- Written: K. Adlam, 25/5/05
-------------------------------------------------------
--
-------------------------------------------------------
-- Clear variables
-------------------------------------------------------
undefine existing_table;
undefine swe;
undefine swn;
undefine nee;
undefine nen;
undefine keyfield;
-------------------------------------------------------
-- CREATE GEOMETRY TABLE
-------------------------------------------------------
create table &&existing_table._SP(
fid number(38) primary key,
shape mdsys.sdo_geometry)
TABLESPACE TS3A
/
-------------------------------------------------------
-- CREATE VIEW
-------------------------------------------------------
CREATE VIEW &&existing_table._SV AS SELECT b.*,a.*
FROM &&existing_table._sp A, &&existing_table B
WHERE A.FID=B.&&keyfield
/
-------------------------------------------------------
-- Initial insert
-------------------------------------------------------
INSERT INTO &&existing_table._sp SELECT &&keyfield,
DECODE(
DECODE(&&swe,null,'A','B')||DECODE(&&swn,null,'A','B')||
DECODE(&&nee,null,'A','B')||DECODE(&&nen,null,'A','B')
,'BBBB',
mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
mdsys.sdo_ordinate_array(&&swe,&&swn,&&nee,&&swn,&&nee,&&nen,&&swe,&&nen,&&swe,&&swn ))
,NULL)
FROM &&existing_table
/
--INSERT INTO &&existing_table._sp SELECT &&keyfield,
--mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
--mdsys.sdo_ordinate_array(&&swe,&&swn,&&nee,&&swn,&&nee,&&nen,&&swe,&&nen,&&swe,&&swn ))
--FROM &&existing_table
--WHERE &&east IS NOT NULL and &&north IS NOT NULL
--/
--INSERT INTO &&existing_table._sp SELECT &&keyfield, NULL
--FROM &&existing_table
--WHERE &&east IS NULL and &&north IS NULL
--/
COMMIT
/
-------------------------------------------------------
-- CREATE TRIGGER
-------------------------------------------------------
create or replace trigger &&existing_table._SPT
after insert or update or delete on &&existing_table
/*
 Trigger to maintain related geometry table
 Inserts, updates and deletes are tracked and
 the spatial table is modified appropriately.

 Created by Keith AM Adlam 20/10/2005
*/

for each row

DECLARE
```

```
  aPoly mdsys.sdo_geometry;

BEGIN

  IF DELETING THEN
    DELETE FROM KAMA.&&existing_table._SP WHERE &&existing_table._SP.fid = :old.&&keyfield;
  END IF;

  IF INSERTING THEN

    --INSERT INTO &&existing_table._sp VALUES( :new.&&keyfield,
    --DECODE(
    --DECODE(:new.&&swe,null,'A','B')||DECODE(:new.&&swn,null,'A','B')||
    --DECODE(:new.&&nee,null,'A','B')||DECODE(:new.&&nen,null,'A','B'),'BBBB',
    --mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
    --
mdsys.sdo_ordinate_array(:new.&&swe,:new.&&swn,:new.&&nee,:new.&&swn,:new.&&nee,:new.&&nen,
    --:new.&&swe,:new.&&nen,:new.&&swe,:new.&&swn )),NULL));

    IF (:new.&&swe IS NULL OR :new.&&swn IS NULL OR
        :new.&&nee IS NULL OR :new.&&nen IS NULL)THEN
      INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,Null);
    ELSE
      aPoly :=
      mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
      mdsys.sdo_ordinate_array(:new.swe,:new.swn, :new.nee,:new.swn, :new.nee,:new.nen,
     :new.swe,:new.nen,:new.swe,:new.swn ));
      INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,aPoly);
    END IF;

  END IF;

  IF UPDATING THEN

    --IF(:new.&&swe != :old.&&swe or :new.&&swn != :old.&&swn or :new.&&nee != :old.&&nee or
:new.&&nen != :old.&&nen)THEN
    --  UPDATE &&existing_table._sp SET SHAPE =
    --  DECODE(
    --  DECODE(:new.&&swe,null,'A','B')||DECODE(:new.&&swn,null,'A','B')||
    --  DECODE(:new.&&nee,null,'A','B')||DECODE(:new.&&nen,null,'A','B'),'BBBB',
    --  mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
    --
mdsys.sdo_ordinate_array(:new.&&swe,:new.&&swn,:new.&&nee,:new.&&swn,:new.&&nee,:new.&&nen,
    --  :new.&&swe,:new.&&nen,:new.&&swe,:new.&&swn )),NULL)
    --  WHERE FID = :new.&&keyfield;
    --END IF;

    IF(:new.&&swe != :old.&&swe or :new.&&swn != :old.&&swn or :new.&&nee != :old.&&nee or
:new.&&nen != :old.&&nen)THEN
      IF (:new.&&swe IS NULL OR :new.&&swn IS NULL OR :new.&&nee IS NULL OR :new.&&nen IS
NULL)THEN
        UPDATE KAMA.&&existing_table._SP SET SHAPE = NULL WHERE FID=:new.&&keyfield;
      ELSE
        aPoly :=
        mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
        mdsys.sdo_ordinate_array(:new.&&swe,:new.&&swn, :new.&&nee,:new.&&swn,
:new.&&nee,:new.&&nen,
       :new.&&swe,:new.&&nen,:new.&&swe,:new.&&swn ));
        UPDATE KAMA.&&existing_table._SP SET SHAPE= aPoly WHERE FID =:new.&&keyfield;
      END IF;
    END IF;

  END IF;

END &&existing_table._SPT;
/
-------------------------------------------------------------
-- CREATE METADATA FOR SPATIAL TABLE
-------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SP', 'SHAPE',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);
```

```
------------------------------------------------------------
-- CREATE METADATA FOR VIEW
------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SV', 'SHAPE',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);

------------------------------------------------------------
-- CREATE SPATIAL INDEX
------------------------------------------------------------
create index &&existing_table._SP_SI on &&existing_table._SP(shape)
indextype is mdsys.spatial_index
PARAMETERS ('SDO_INDX_DIMS = 2 TABLESPACE = TS3B')
/
------------------------------------------------------------
-- GRANTS on TABLES and VIEW
------------------------------------------------------------
grant select on &&existing_table._SP to public
/
grant select on &&existing_table._SV to public
/
------------------------------------------------------------
-- Clear variables
------------------------------------------------------------
undefine existing_table;
undefine swe;
undefine swn;
undefine nee;
undefine nen;
undefine keyfield;
------------------------------------------------------------
```

EXAMPLE CODE TO SPATIALLY ENABLE TABLES WITH QUADRILATERALS

```
-------------------------------------------------------
-- Name: Quads_Related
-- Purpose: To Spatially enable an existing Oracle Table
--          containing quadrilaterals as 4 corners
--          (often SWE,SWN,SEE,SEN,NEE,NEN,NWE,NWN in BGS tables)
--          using a related spatial table.
-- Written: K. Adlam, 25/5/05
-------------------------------------------------------
--
-------------------------------------------------------
-- Clear variables
-------------------------------------------------------
undefine existing_table;
undefine swe;
undefine swn;
undefine see;
undefine sen;
undefine nee;
undefine nen;
undefine nwe;
undefine nwn;
undefine keyfield;
-------------------------------------------------------
-- CREATE GEOMETRY TABLE
-------------------------------------------------------
create table &&existing_table._SP(
fid number(38) primary key,
shape mdsys.sdo_geometry)
TABLESPACE TS3A
/
-------------------------------------------------------
-- CREATE VIEW
-------------------------------------------------------
CREATE VIEW &&existing_table._SV AS SELECT b.*,a.SHAPE
FROM &&existing_table._sp A, &&existing_table B
WHERE A.FID=B.&&keyfield
/
-------------------------------------------------------
-- Initial insert
-------------------------------------------------------
INSERT INTO &&existing_table._sp SELECT &&keyfield,
DECODE(
DECODE(&&swe,null,'A','B')||DECODE(&&swn,null,'A','B')||
DECODE(&&see,null,'A','B')||DECODE(&&sen,null,'A','B')||
DECODE(&&nee,null,'A','B')||DECODE(&&nen,null,'A','B')||
DECODE(&&nwe,null,'A','B')||DECODE(&&nwn,null,'A','B')
,'BBBBBBBB',
mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
mdsys.sdo_ordinate_array(swe,swn,see,sen,nee,nen,nwe,nwn,swe,swn ))
,NULL)
FROM &&existing_table
/
--INSERT INTO &&existing_table._sp SELECT &&keyfield,
--mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
--mdsys.sdo_ordinate_array(swe,swn,see,sen,nen,nwe,nwn,swe,swn ))
--FROM &&existing_table
--WHERE &&east IS NOT NULL and &&north IS NOT NULL
--/
--INSERT INTO &&existing_table._sp SELECT &&keyfield, NULL
--FROM &&existing_table
--WHERE &&east IS NULL and &&north IS NULL
--/
COMMIT
/
-------------------------------------------------------
-- CREATE TRIGGER
-------------------------------------------------------
create or replace trigger &&existing_table._SPT
after insert or update or delete on &&existing_table
/*
 Trigger to maintain related geometry table
 Inserts, updates and deletes are tracked and
 the spatial table is modified appropriately.

 Created by Keith AM Adlam 20/10/2005
```

```
*/

for each row

DECLARE
  aPoly mdsys.sdo_geometry;

BEGIN

   IF DELETING THEN
     DELETE FROM KAMA.&&existing_table._SP WHERE &&existing_table._SP.fid = :old.&&keyfield;
   END IF;

   IF INSERTING THEN
     IF (:new.&&swe IS NULL OR :new.&&swn IS NULL OR
         :new.&&see IS NULL OR :new.&&sen IS NULL OR
         :new.&&nee IS NULL OR :new.&&nen IS NULL OR
         :new.&&nwe IS NULL OR :new.&&nwn IS NULL)THEN
       INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,Null);
     ELSE
       aPoly:=
       mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
       mdsys.sdo_ordinate_array(:new.swe,:new.swn, :new.see,:new.sen, :new.nee,:new.nen,
      :new.nwe,:new.nwn,:new.swe,:new.swn ));
       INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,aPoly);
     END IF;
   END IF;

   IF UPDATING THEN

     IF(:new.&&swe != :old.&&swe or :new.&&swn != :old.&&swn or
     :new.&&see != :old.&&see or :new.&&sen != :old.&&sen or
     :new.&&nee != :old.&&nee or :new.&&nen != :old.&&nen or
     :new.&&nwe != :old.&&nwe or :new.&&nwn != :old.&&nwn)THEN

       IF (:new.&&swe IS NULL OR :new.&&swn IS NULL OR
       :new.&&see IS NULL OR :new.&&sen IS NULL OR
       :new.&&nee IS NULL OR :new.&&nen IS NULL OR
       :new.&&nwe IS NULL OR :new.&&nwn IS NULL)THEN
         UPDATE KAMA.&&existing_table._SP SET SHAPE = NULL WHERE FID = :new.&&keyfield;
       ELSE
         aPoly:=
         mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
         mdsys.sdo_ordinate_array(:new.swe,:new.swn, :new.see,:new.sen, :new.nee,:new.nen,
         :new.nwe,:new.nwn,:new.swe,:new.swn ));
         UPDATE KAMA.&&existing_table._SP SET SHAPE = aPoly WHERE FID = :new.&&keyfield;
       END IF;
     END IF;

   END IF;

END &&existing_table._SPT;
/
-------------------------------------------------------------
-- CREATE METADATA FOR SPATIAL TABLE
-------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SP', 'SHAPE',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);

-------------------------------------------------------------
-- CREATE METADATA FOR VIEW
-------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SV', 'SHAPE',
```

```
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);

-------------------------------------------------------
-- CREATE SPATIAL INDEX
-------------------------------------------------------
create index &&existing_table._SP_SI on &&existing_table._SP(shape)
indextype is mdsys.spatial_index
PARAMETERS ('SDO_INDX_DIMS = 2 TABLESPACE = TS3B')
/
-------------------------------------------------------
-- GRANTS on TABLES and VIEW
-------------------------------------------------------
grant select on &&existing_table._SP to public
/
grant select on &&existing_table._SV to public
/
-------------------------------------------------------
-- Clear variables
-------------------------------------------------------
undefine existing_table;
undefine swe;
undefine swn;
undefine see;
undefine sen;
undefine nee;
undefine nen;
undefine nwe;
undefine nwn;
undefine keyfield;
-------------------------------------------------------
```

EXAMPLE CODE TO SPATIALLY ENABLE TABLES WITH OS TILES

```
-------------------------------------------------------
-- Name: OS_Tiles_Related
-- Purpose: To Spatially enable an existing Oracle Table
--          containing a field with OS Tiles
--          using a related spatial table.
-- Written: K. Adlam, 19/9/05
-------------------------------------------------------
--
-------------------------------------------------------
-- Clear variables
-------------------------------------------------------
undefine existing_table;
undefine swe;
undefine swn;
undefine see;
undefine sen;
undefine neee;
undefine nen;
undefine nwe;
undefine nwn;
undefine keyfield;
undefine sheetfield;
-------------------------------------------------------
-- CREATE GEOMETRY TABLE
-------------------------------------------------------
create table &&existing_table._SP(
fid number(38) primary key,
shape mdsys.sdo_geometry)
TABLESPACE TS3A
/
-------------------------------------------------------
-- CREATE VIEW
-------------------------------------------------------
CREATE VIEW &&existing_table._SV AS SELECT b.*,a.SHAPE
FROM &&existing_table._sp A, &&existing_table B
WHERE A.FID=B.&&keyfield
/
-------------------------------------------------------
-- Initial insert
-------------------------------------------------------
INSERT INTO &&existing_table._sp SELECT &&keyfield, ngtiletopoly(&&sheetfield)
FROM &&existing_table
/
COMMIT
/
-------------------------------------------------------
-- CREATE TRIGGER
-------------------------------------------------------
create or replace trigger &&existing_table._SPT
after insert or update or delete on &&existing_table
/*
 Trigger to maintain related geometry table
 Inserts, updates and deletes are tracked and
 the spatial table is modified appropriately.

 Created by Keith AM Adlam 20/10/2005
*/

for each row

DECLARE
  ddpoint mdsys.sdo_geometry;
  x1 number;
  y1 number;
  x2 number;
  y2 number;
  sheet varchar2(10);

BEGIN

   IF DELETING THEN
     DELETE FROM KAMA.&&existing_table._SP WHERE &&existing_table._SP.fid = :old.&&keyfield;
```

```
   END IF;

   IF INSERTING THEN
     sheet:= :new.&&sheetfield;
     ddpoint:= ngtiletopoly(sheet);
     INSERT INTO KAMA.&&existing_table._SP VALUES(:new.&&keyfield,ddpoint);
   END IF;

   IF UPDATING THEN

     IF (:new.&&sheetfield != :old.&&sheetfield)THEN
       sheet:= :new.&&sheetfield;
       ddpoint:= ngtiletopoly(sheet);
       UPDATE KAMA.&&existing_table._SP Set SHAPE = ddpoint where FID = :new.&&keyfield;
     END IF;
  END IF;

END &&existing_table._SPT;
/
------------------------------------------------------------
-- CREATE METADATA FOR SPATIAL TABLE
------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SP', 'SHAPE',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);

------------------------------------------------------------
-- CREATE METADATA FOR VIEW
------------------------------------------------------------
INSERT INTO USER_SDO_GEOM_METADATA
VALUES (
'&&existing_table._SV', 'SHAPE',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', -100000, 800000, 0.5),
MDSYS.SDO_DIM_ELEMENT('Y', -100000, 1400000, 0.5)
), 81989);

------------------------------------------------------------
-- CREATE SPATIAL INDEX
------------------------------------------------------------
create index &&existing_table._SI on &&existing_table._SP(shape)
indextype is mdsys.spatial_index
PARAMETERS ('SDO_INDX_DIMS = 2 TABLESPACE = TS3B')
/
------------------------------------------------------------
-- GRANTS on TABLES and VIEW
------------------------------------------------------------
grant select on &&existing_table._SP to public
/
grant select on &&existing_table._SV to public
/
------------------------------------------------------------
-- Clear variables
------------------------------------------------------------
undefine existing_table;
undefine swe;
undefine swn;
undefine see;
undefine sen;
undefine neee;
undefine nen;
undefine nwe;
undefine nwn;
undefine keyfield;
undefine sheetfield;
------------------------------------------------------------
```

FUNCTIONS CALLED BY OS_TILES_RELATED

```
--------------------------------------------------------------------
--
--  Routine Name: NGTileToPoly
--  Purpose: To return a polygon for specified national grid map sheet
--  Written: K. Adlam, 14/9/05 after VBA Version 2002,
--  after Mapgrd.aml September 1994.
--  Notes: Null is returned if the supplied OS tile
--  is not valid.
--
--------------------------------------------------------------------
CREATE OR REPLACE FUNCTION NGTileToPoly
(OS_TILE varchar2) RETURN mdsys.sdo_geometry AS

--DECLARE
x1 number;
y1 number;
x2 number;
y2 number;
x3 number;
y3 number;
x4 number;
y4 number;
map varchar2(10);
delta number;
gcode varchar2(2);
map34 varchar2(2);
map56 varchar2(2);
map78 varchar2(4);
map58 varchar2(4);
map3 varchar2(1);
map4 varchar2(1);
map5 varchar2(1);
map6 varchar2(1);
east number(2);
north number(2);
ind Integer;
TYPE scode_typ IS VARRAY(91) of VARCHAR2(2);
TYPE easting_typ IS VARRAY(91) of NUMBER(2);
TYPE northing_typ IS VARRAY(91) of NUMBER(2);
scode scode_typ;
easting easting_typ;
northing northing_typ;
numdigits Integer;
digits varchar2(10);
xstr varchar2(10);
ystr varchar2(10);
pow Integer;
dX number;
dY number;
ddpoly mdsys.sdo_geometry;
bad_ng_tile EXCEPTION;

BEGIN

map := Lower(os_tile);
map := Replace(map, ' ', '');
--------------------------------------------------------------------
-- Check for special map sheets
--------------------------------------------------------------------
If (map = '') Then
  Raise bad_ng_tile;

elsif (map = 'uks') Then
  ddpoly:=
  mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
  mdsys.sdo_ordinate_array(50000,10000,670000,10000,670000,500000,50000,500000,50000,10000 ));
  return ddpoly;

elsif (map = 'ukn') Then
  ddpoly:=
  mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
  mdsys.sdo_ordinate_array(50000,500000,530000,500000,530000,990000,50000,990000,50000,500000 ));
  return ddpoly;

elsif (map = 'uk') Then
```

```
  ddpoly:=
  mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
  mdsys.sdo_ordinate_array(48000,0,680000,0,680000,1000000,48000,1000000,48000,0 ));
  return ddpoly;

elsif (map = 'ukl') Then
  ddpoly:=
  mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
  mdsys.sdo_ordinate_array(10000,-120000,700000,-120000,700000,1280000,10000,1280000,10000,-
120000 ));
  return ddpoly;

Else
  --'
  --' Check for unequal number of characters
  --'
  If (Mod(Length(map),2) <> 0) Then
    Raise bad_ng_tile;
  End If;

  ---------------------------------------------------------------------
  -- Split map sheet up into constituent parts
  ---------------------------------------------------------------------
  gcode := Substr(map, 1, 2);
  map34 := Substr(map, 3, 2);
  map56 := Substr(map, 5, 2);
  map78 := Substr(map, 7, 2);
  map58 := Substr(map, 5, 4);
  map3 := Substr(map, 3, 1);
  map4 := Substr(map, 4, 1);
  map5 := Substr(map, 5, 1);
  map6 := Substr(map, 6, 1);

  ---------------------------------------------------------------------
  -- Set initial values for variables
  ---------------------------------------------------------------------
  x3 := 0;
  x4 := 0;
  y3 := 0;
  y4 := 0;
  delta := 0;

  ---------------------------------------------------------------------
  -- Setup arrays
  ---------------------------------------------------------------------
  scode := scode_typ(
  'hl', 'hm', 'hn', 'ho', 'hp', 'jl', 'jm',
  'hq', 'hr', 'hs', 'ht', 'hu', 'jq', 'jr',
  'hv', 'hw', 'hx', 'hy', 'hz', 'jv', 'jw',
  'na', 'nb', 'nc', 'nd', 'ne', 'oa', 'ob',
  'nf', 'ng', 'nh', 'nj', 'nk', 'of', 'og',
  'nl', 'nm', 'nn', 'no', 'np', 'ol', 'om',
  'nq', 'nr', 'ns', 'nt', 'nu', 'oq', 'or',
  'nv', 'nw', 'nx', 'ny', 'nz', 'ov', 'ow',
  'sa', 'sb', 'sc', 'sd', 'se', 'ta', 'tb',
  'sf', 'sg', 'sh', 'sj', 'sk', 'tf', 'tg',
  'sl', 'sm', 'sn', 'so', 'sp', 'tl', 'tm',
  'sq', 'sr', 'ss', 'st', 'su', 'tq', 'tr',
  'sv', 'sw', 'sx', 'sy', 'sz', 'tv', 'tw');

  easting := easting_typ(
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6,
  0, 1, 2, 3, 4, 5, 6);

  northing := northing_typ(
```

```
  12, 12, 12, 12, 12, 12, 12,
  11, 11, 11, 11, 11, 11, 11,
  10, 10, 10, 10, 10, 10, 10,
  9, 9, 9, 9, 9, 9, 9,
  9, 8, 8, 8, 8, 8, 8,
  7, 7, 7, 7, 7, 7, 7,
  6, 6, 6, 6, 6, 6, 6,
  5, 5, 5, 5, 5, 5, 5,
  4, 4, 4, 4, 4, 4, 4,
  3, 3, 3, 3, 3, 3, 3,
  2, 2, 2, 2, 2, 2, 2,
  1, 1, 1, 1, 1, 1, 1,
  0, 0, 0, 0, 0, 0, 0);

  ------------------------------------------------------------------
  -- Convert sheet code to numbers
  ------------------------------------------------------------------
  ind := 0;
  --FOR i IN scode.FIRST .. scode.LAST
  FOR i IN 1 .. 91 LOOP
    if (scode(i) = gcode)then
      ind := i;
    end if;
  END LOOP;
  if(ind > 0)then
    east := easting(ind);
    north := northing(ind);
  else
    Raise bad_ng_tile;
  end if;

  ------------------------------------------------------------------
  -- Check for 100km sheet e.g. SJ
  ------------------------------------------------------------------
  If (Length(map) = 2) Then
    delta := 100000;

  ------------------------------------------------------------------
  -- Check for 100km quarter sheet e.g. SJNW
  ------------------------------------------------------------------
  elsif (Length(map) = 4) Then
    If ((map34 = 'sw') Or (map34 = 'se') Or (map34 = 'ne') Or (map34 = 'nw')) Then --' level 2
100km quarter
       delta := 50000;
      If (map34 = 'se') Then
        x3 := 5;
      End If;
      If (map34 = 'ne') Then
        x3 := 5;
        y3 := 5;
      End If;
      If (map34 = 'nw') Then
        y3 := 5;
      End If;
  ------------------------------------------------------------------
  -- Check for 10km sheet e.g. SJ24
  ------------------------------------------------------------------
    Else
      If (IsNumeric(map3) And IsNumeric(map4)) Then
        delta := 10000;
        x3 := to_number(map3);
        y3 := to_number(map4);
      Else
        Raise bad_ng_tile;
      End If;
    End If;

  ------------------------------------------------------------------
  -- Check for 10km quarter sheet e.g. SJ24NE
  ------------------------------------------------------------------
  elsif (Length(map) = 6 And IsNumeric(map3) And IsNumeric(map4)) Then
    If ((map56 = 'sw') Or (map56 = 'se') Or (map56 = 'ne') Or (map56 = 'nw')) Then
      delta := 5000;
      x3 := to_number(map3);
      y3 := to_number(map4);
      If (map56 = 'se') Then
```

```
      x4 := 5;
    End If;
    If (map56 = 'ne') Then
      x4 := 5;
      y4 := 5;
     End If;
    If (map56 = 'nw') Then
     y4 := 5;
    End If;
--------------------------------------------------------------------
--  Check for 1km sheet e.g. SJ2434
--------------------------------------------------------------------
  Else
    If (IsNumeric(map5) And IsNumeric(map6)) Then
      x3 := to_number(map3);
      y3 := to_number(map5);
      delta := 1000;
      x4 := to_number(map4);
      y4 := to_number(map6);
    Else
      Raise bad_ng_tile;
    End If;
  End If;
--------------------------------------------------------------------
--  Check for 0.1km sheet etc. e.g. SJ243444
--------------------------------------------------------------------
elsif ((Length(map) > 6) And (IsNumeric(Right(map, Length(map) - 2)))) Then
  numdigits := Length(map) - 2;
  digits := SUBSTR(map,(Length(map)-numdigits)+1, numdigits);
  xstr := SUBSTR(digits,1, numdigits / 2);
  ystr := SUBSTR(digits,(Length(map)-(numdigits/2))+1 , numdigits / 2);
  pow := (5 - (numdigits / 2));
  dX := to_number(xstr) * POWER(10, pow);
  dY := to_number(ystr) * POWER(10 , pow);
  x1 := (east * 100000) + dX;
  y1 := (north * 100000) + dY;
  delta := POWER(10, pow);
  x2 := x1 + delta;
  y2 := y1 + delta;
  ddpoly:=
  mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
  mdsys.sdo_ordinate_array(x1,y1,x2,y1,x2,y2,x1,y2,x1,y1 ));
  return ddpoly;

elsif ((Length(map) = 8)) Then
--------------------------------------------------------------------
--  Check for 2.5 km sheet(field slip) e.g. SJ24nwnw
--------------------------------------------------------------------
  If (((map56 = 'sw') Or (map56 = 'se') Or (map56 = 'ne') Or (map56 = 'nw'))
  And ((map78 = 'sw') Or (map78 = 'se') Or (map78 = 'ne') Or (map78 = 'nw'))) Then
    delta := 2500;
    x3 := to_number(map3);
    y3 := to_number(map4);
    If (map58 = 'nwnw') Then
      x4 := 0; y4 := 7.5;
    elsif (map58 = 'nwne') Then
      x4 := 2.5; y4 := 7.5;
    elsif (map58 = 'nenw') Then
      x4 := 5; y4 := 7.5;
    elsif (map58 = 'nene') Then
      x4 := 7.5; y4 := 7.5;
    elsif (map58 = 'nwsw') Then
      x4 := 0; y4 := 5;
    elsif (map58 = 'nwse') Then
      x4 := 2.5; y4 := 5;
    elsif (map58 = 'nesw') Then
      x4 := 5; y4 := 5;
    elsif (map58 = 'nese') Then
      x4 := 7.5; y4 := 5;
    elsif (map58 = 'swnw') Then
      x4 := 0; y4 := 2.5;
    elsif (map58 = 'swne') Then
      x4 := 2.5; y4 := 2.5;
    elsif (map58 = 'senw') Then
      x4 := 5; y4 := 2.5;
    elsif (map58 = 'sene') Then
      x4 := 7.5; y4 := 2.5;
```

```
      elsif (map58 = 'swsw') Then
        x4 := 0; y4 := 0;
      elsif (map58 = 'swse') Then
        x4 := 2.5; y4 := 0;
      elsif (map58 = 'sesw') Then
        x4 := 5; y4 := 0;
      elsif (map58 = 'sese') Then
        x4 := 7.5; y4 := 0;
      End If;
    --CASE map58
    --  WHEN 'nwnw' Then x4 := 0.0; y4 := 7.5;
    --  WHEN 'nwne' Then x4 := 2.5; y4 := 7.5;
    --  WHEN 'nenw' Then x4 := 5.0; y4 := 7.5;
    --  WHEN 'nene' Then x4 := 7.5; y4 := 7.5;
    --  WHEN 'nwsw' Then x4 := 0.0; y4 := 5.0;
    --  WHEN 'nwse' Then x4 := 2.5; y4 := 5.0;
    --  WHEN 'nesw' Then x4 := 5.0; y4 := 5.0;
    --  WHEN 'nese' Then x4 := 7.5; y4 := 5.0;
    --  WHEN 'swnw' Then x4 := 0.0; y4 := 2.5;
    --  WHEN 'swne' Then x4 := 2.5; y4 := 2.5;
    --  WHEN 'senw' Then x4 := 5.0; y4 := 2.5;
    --  WHEN 'sene' Then x4 := 7.5; y4 := 2.5;
    --  WHEN 'swsw' Then x4 := 0.0; y4 := 0.0;
    --  WHEN 'swse' Then x4 := 2.5; y4 := 0.0;
    --  WHEN 'sesw' Then x4 := 5.0; y4 := 0.0;
    --  WHEN 'sese' Then x4 := 7.5; y4 := 0.0;
    --END CASE;
    Else
      Raise bad_ng_tile;
    End If;
  Else
    Raise bad_ng_tile;
  End If;
End If;
-----------------------------------------------------------------------
-- Calculate corner coordinates of map tile
-----------------------------------------------------------------------
x1 := (east * 100000) + (x3 * 10000) + (x4 * 1000);
y1 := (north * 100000) + (y3 * 10000) + (y4 * 1000);
x2 := x1 + delta;
y2 := y1 + delta;

ddpoly:=
mdsys.sdo_geometry(2003,81989,null,mdsys.sdo_elem_info_array(1,1003,1),
mdsys.sdo_ordinate_array(x1,y1,x2,y1,x2,y2,x1,y2,x1,y1 ));
return ddpoly;
--dbms_output.put_line(x1 ||' '||y1||' '||x2||' '||y2);
EXCEPTION
  WHEN bad_ng_tile then
    x1 := 0;
    y1 := 0;
    x2 := 0;
    y2 := 0;
    return null;
    --dbms_output.put_line('No such National grid sheet');
    --dbms_output.put_line(x1 ||' '||y1||' '||x2||' '||y2);
  WHEN OTHERS THEN
    x1 := 0;
    y1 := 0;
    x2 := 0;
    y2 := 0;
    return null;

END;
/

CREATE OR REPLACE FUNCTION ISINTEGER
(IN_STR IN varchar2) RETURN BOOLEAN AS
/*
 Function to determine if a string contains a valid integer.
 The function must be passed a string.
 Created by Keith AM Adlam., 14/9/05
*/
BEGIN
```

```
 Return LTRIM(RTRIM(TRANSLATE(IN_STR,'0123456789','**********'),'*'),' ')IS NULL;
END;
/

CREATE OR REPLACE FUNCTION ISNUMERIC
(IN_STR IN varchar2) RETURN BOOLEAN AS
/*
 Function to determine if a string contains a valid number.
 The function must be passed a string.
 Created by Keith AM Adlam., 14/9/05
*/
num number;
BEGIN
  num := TO_NUMBER(in_Str);
 Return True;
EXCEPTION
  WHEN VALUE_ERROR THEN
    Return False;
END;
/

CREATE OR REPLACE FUNCTION LEFT
(IN_STR IN varchar2, NUM_CHAR IN Integer) RETURN VARCHAR2 AS
/*
 Function to extract the Left Characters from a string.
 The function must be passed original string and the number
 of characters to be extracted.
 Created by Keith AM Adlam., 14/9/05
*/
BEGIN
 Return substr(in_str,1,num_char);
END;
/

CREATE OR REPLACE FUNCTION RIGHT
(IN_STR IN varchar2, NUM_CHAR IN Integer) RETURN VARCHAR2 AS
/*
 Function to extract the Right Characters from a string.
 The function must be passed original string and the number
 of characters to be extracted.
 Created by Keith AM Adlam., 14/9/05
*/
BEGIN
   Return substr(in_str,(Length(in_str)-num_char)+1,num_char);
END;
/
---------------------------------------------------------------------
```

Appendix 6 Example of code To UPDATE a geometry table via a batch job

ALTERNATIVE METHOD OF UPDATING A GEOMETRY TABLE IN BATCH MODE

```
-------------------------------------------------------
-- Name: Update Points
-- Purpose: To update geometry table for base table
--          containing X and Y coordinates.
-- Written: K. Adlam, 10/10/05
-------------------------------------------------------
--
-------------------------------------------------------
-- variables passed as arguments
-------------------------------------------------------
define existing_table = &1;
define keyfield = &2;
define east = &3;
define north = &4;
-------------------------------------------------------
-- Remove records that do no exist in the base table
-------------------------------------------------------
DELETE FROM &&existing_table._SP WHERE FID NOT IN
(SELECT BGS_ID FROM &&existing_table)
/
-------------------------------------------------------
-- Remove records that have been updated in the last 5 days
-------------------------------------------------------
DELETE FROM &&existing_table._SP WHERE FID IN
(SELECT BGS_ID FROM &&existing_table WHERE DATE_UPDATED > (SYSDATE-5))
/
-------------------------------------------------------
-- Insert new records into geometry table
-------------------------------------------------------
INSERT INTO &&existing_table._sp SELECT &&keyfield,
DECODE(
DECODE(&&east,null,'A','B')||DECODE(&&north,null,'A','B')
,'BB',
mdsys.sdo_geometry(2001,81989,mdsys.sdo_POINT_TYPE(&&east,&&north,Null),Null,Null)
,NULL)
FROM &&existing_table WHERE &&keyfield NOT IN (SELECT FID FROM &&existing_table._sp)
/
-------------------------------------------------------
-- Clear variables and COMMIT
-------------------------------------------------------
undefine existing_table;
undefine keyfield;
undefine east;
undefine north;
COMMIT
/
-------------------------------------------------------
```