ArcFM is a powerful extension of ESRI's ArcGIS® 9 platform that provides a complete enterprise utility solution for editing, modeling, maintenance, and management of facility information for electric, gas, and water/wastewater utilities.

**What is ArcFM Server?**  
Let us start with what is ArcFM. ArcFM (Facility Management) has been created on top of ArcGIS platform by [Miner and Miner](http://www.telvent-gis.com/information/) as desktop application for utility users. Since ESRI was not focusing on Utility users, M&M has found a niche and tackled it with a solid product. Made a good money out of it.  Schneider Electric found this gem and bought it.   
  
ArcFM Server is an extension for [ArcGIS Server](http://www.husseinnasser.com/2009/08/6-ways-to-optimize-arcgis-server.html) (now 10.2), allows you to support ArcFM functionalities on your GIS Services. They probably didn't say this in the brochures and flyers but ArcFM Server is won't workwithout ArcFM Desktop Editor. You need the desktop version to configure your geodatabase, upgrade it to support ArcFM etc.  
  
Here is a [live demo](http://server.arcfmsolution.com/ArcFMSilverlight/)of ArcFM Server, it is on 2.4 at the time of writing this article.  
  
**Good thing about ArcFM Server?**  
At my company, we have implemented ArcFM Server for one reason. We didn't want to customize anything. We want a product that we can install, configure and voila!. We have ArcFM desktop fully configured with feeder manager, we implemented ArcFM Server and bang! we have all our electric traces right on ArcFM server page with some basic nice features with no single line of code.  
  
**Disadvantages**  
**Lack of Support and Training**  
If you got error in the installation or configuration ArcFM server (and yes you will) good luck with that, you are on your own. The [forum](http://www.telvent-gis.com/support/forums/viewforum.php?f=7)is filled with unanswered questions. No training yet offered for ArcFM server.  
  
**Silverlight**  
I said this to the guys at schneider electric and I'm gonna say it again. There is an elephant in the room and it is called silverlight. Why are you still supporting this? The only reason we are going Web 2.0 so that we can consume our services from multiple sources. Silverlight is not supported on any of the tablets or phones. HTML5 however is supported. Either build us native applications on mobile to consume these services or support HTML5.  
  
**Trace Tools are so slow**  
ArcFM Server Tracing tools take on average 40 seconds to execute. That is not acceptable in our business so we end up creating our own from scratch. My custom trace achieves the same result with only 0.5 seconds to run. 98% Less!  
  
So I'll give this product a year or two to mature, and until ArcFM have a vivid strategy for this product.

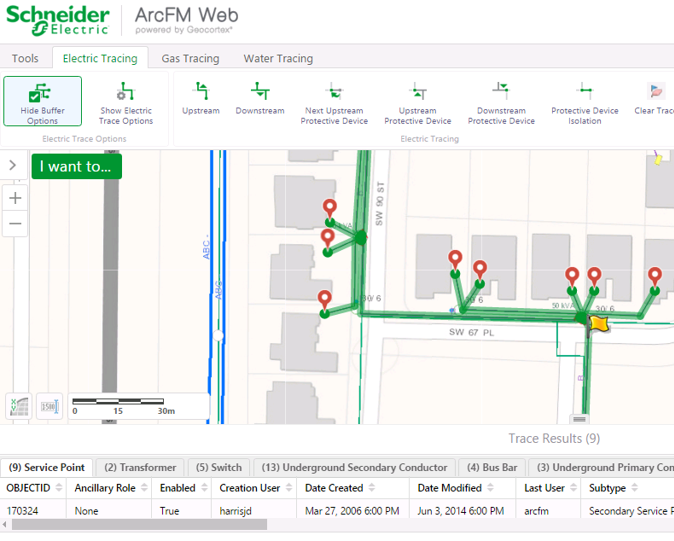
ArcFM Web 4.5

Highlights:

1. Electric, Gas and Water tracing functionality - including Trace Options and Temporary Marks
2. Saving and Sharing of Project Files
3. Searching Non-Spatial Data

With ArcFM Web you can now run traces to do things like identify protective devices, locate valves to isolate an area and find affected customers. These traces can be used in conjunction with other core ArcFM Web functionality (i.e. reports, charts, workflows) to generate the information that's needed for your organization.

Wondering how to share graphics, text or selection sets in ArcFM Web with other people in your organization? Well now you can quickly and easily share information via a Project File so that the application state can be shared with other users.



ArcFM Server 10.1

* ArcFM Server 10.1 is not compatible with Esri's ArcGIS 10.1 SP1

ArcFM Server 10.1.1

* [ArcFM 10.1.1 was released on August 2nd!](http://resources.arcfmsolution.com/emailers/1011/index.html)
* We're seeing roughly a **50 - 100%** performance improvement in most ArcFM Server operations upgrading to ArcGIS for Server 10.1 SP1 w/[Quality Improvement Patch](http://support.esri.com/en/downloads/patches-servicepacks/view/productid/67/metaid/2064) (QIP) and ArcFM Server 10.1.1 w/[QIP](https://infrastructurecommunity.schneider-electric.com/docs/DOC-3020)

ArcFM Server 10.2

* [ArcFM 10.2 was released on September 4th!](https://infrastructurecommunity.schneider-electric.com/community/support/gis/blog/2013/09/04/announcing-the-arcfm-solution-102-release)
* We're seeing roughly a **50 - 100%** performance improvement in most ArcFM Server operations upgrading from 10.1 to 10.2 and applying [Patch #273](https://infrastructurecommunity.schneider-electric.com/docs/DOC-2684) or newer
* ArcFM Server 10.2 will support viewing of Feeder Manager 2.0 info with Esri's hot fixes for NIM093448:  QFE-102-S-266424 (ArcGIS 10.2 for Server) and QFE-102-DT-266424 (ArcGIS 10.2 for Desktop)
* We intend to add support for editing data that causes changes in a Feeder Manager 2.0 network to ArcFM Server 10.2 in the future.

ArcFM for Silverlight

* ArcFM for Silverlight 2.4 was released in April 2013 and works with ArcFM Server 10.0.3 SP1, 10.1, 10.1.1, or 10.2.  If you are using the Address Search functionality, we recommend [changing to the newer geocoding services](https://infrastructurecommunity.schneider-electric.com/thread/23884)**before December 31, 2013**.
* ArcFM for Silverlight 3.2 is currently planned for late Q3 or Q4 of 2014.  Our focus will be on ensuring compatibility with Version 3.2 of Esri's ArcGIS API for Silverlight and high-resolution printing.  ArcFM for Silverlight 3.2 will require the latest version of ArcFM Server available at the time of its release for full functionality.

[REST API](http://resources.arcfmsolution.com/10.1/ServerSDK/RESTAPI1001.html)

* The ArcFM Server REST API provides a simple, open Web interface to services hosted by ArcGIS for Server.

ArcGIS Server Architecture Overview

Based on the ArcObjects framework, a common set of software  components used in ESRI products ß

Is a distributed system, that uses several components to host,  manage, and process GIS or Web services ß

Can be deployed on a single or multiple machines ß Internet / LAN/ WAN access ß

Multiplatform support – Windows and UNIX

Overview of Components

Clients

End users connect to ArcGIS Server with  mobile, web browser, or desktop applications. ArcGIS Desktop applications used to  manage and create data

Web server

Machine with web serving capabilities, is comprised of the web server and  applications server

GIS server

Is comprised of two distinct components  the server object manager and server object container

Data Server

Stores the GIS resources

Server Object Manager (SOM)

The SOM is a Windows service

Manages requests to SOCs

Handles load distribution among available SOCs

Error logging

Server Object Container (SOC)

The SOC can be one or more machines

Each SOC machine can host multiple processes

Data in geodatabases or other file­based data is accessible by the SOC

Receives incoming requests from the SOM

The SOC will process this request

Returns the output result back to the SOM which in turn sends the output to the client

DEPLOYEMENT OF ArcSDE

Prior to 10.1 (10.0, 9.x, 8.x) if you wanted to use a Geodatabase with all of the cool Geodatabase functionality (versioning, archiving, topology, relationship classes, terrains, geometric networks, etc) on a DBMS (Oracle, SQL Server, Postgresql, Informix, DB2) you had to perform a few steps.  
  
First, you needed to install ArcSDE onto your machine, usually this was your DBMS server. This install included the files needed to run ArcSDE command line utilities, the application server, as well as the ArcSDE post installer.  
  
After installing this onto your machine you would need to run the ArcSDE post installer. The main purpose of this post install was to install the Geodatabase into your enterprise database. This includes all the stored procedures, functions, privileges, and schema needed to provide the functionality I mentioned earlier. The post install could also be used to set up your application server.   
  
The application server can be used to connect from a client machine to the DBMS/Geodatabase. It is used more or less to manage the connection requests coming in from clients and provide a way for the clients to 'talk' to the DBMS. For a while this was the only way to connect to a Geodatabase. At some point (I think 9.0) we added the ability to make direct connections (2-tier: client-DBMS) to the DBMS. This made the use of the application server (3-tier: client-app server-dbms) optional. For some people this meant they stopped setting up the application server and started using direct connections. Others continued using the application server.  
  
The ArcSDE command line utilities are a method for the Geodatabase administrator to manage data, users, the application server service.  
  
Fast forward to **10.1** - We have tried to allow you to manage your Geodatabase completely within ArcGIS applications (ArcCatalog, ArcMap, ArcGIS Server, etc). This is done through the use of dialogues in ArcMap/Catalog and the use of geoprocessing.

The first thing we did was to break out the installation of the geodatabase schema tasks into geoprocessing tools.

If you want to create an enterprise geodatabase there are now two options.

Option 1, you can use the '**Create Enterprise Geodatabase** ' geoprocessing tool. This tool will create a new empty geodatabase in an existing instance.

Option 2 is to use the '**Enable Enterprise Geodatabase**' tool will allow you to install the Geodatabase schema in an already existing instance.

The new functionality that Jake mentioned that allows you to now connect to a enterprise database (not a geodatabase) is what allow ArcGIS to then enable geodatabase behavior in your enterprise geodatabase.

This second option would be used where you have already set up a database, have user permissions assigned and maybe have loaded some data (essentially converting your database to a geodatabase).

The first option would be used if you are starting from nothing.  
  
Esri recommends using direct connections for making connections to your geodatabase, it is not mandatory that you install the application server.   
  
Much of the commonly used functionality found in the ArcSDE command line utilities is now available either through ArcGIS applications mentioned earlier or through geoprocessing (disconnecting users, identifying locks, loading data, investigating data, etc). For most users the install of these utilities should not be necessary.  
  
If you determine that you really need the application server or the command line utilities they are available as a separate install.  
  
So, this has been a pretty long winded answer to a pretty straightforward question. Answer is, it's not mandatory to install the application server or command line utilities. If you want to take advantage of Geodatabase behavior you do need to run one of the geoprocessing tools to install the Geodatabase.

The performance of an enterprise level ArcSDE geodatabase is influenced by many factors, such as hardware configuration, network configuration, network traffic, and the number of concurrent users.

The tips in this blog entry are not database platform-specific, but they are general tips that will hopefully enable you to improve the performance of your enterprise geodatabase.

So, five best practices for maintaining an ArcSDE geodatabase are:

* Increase the frequency of updating statistics on tables
* Rebuild indexes on tables
* Plan parent-child version relationships carefully
* Compress the geodatabase often
* Monitor system resources

**1. Increase the frequency of updating statistics on tables**

Statistics in the database describe the column data stored in tables. They help the database Query Optimizer to estimate the selectivity of SQL expressions, and enable it to accurately assess the cost of different query plans. The optimizer then chooses the most efficient execution plan for retrieving and/or updating data in the database. Having poor statistics is a frequent cause of poor performance. Keeping accurate up-to-date statistics will help improve database performance, because this will enable the Query Optimizer to make more accurate assessments of query execution plans.

The frequency of updating statistics will depend on the editing activity in the geodatabase. Typically, more editing activity means you should update statistics more frequently. This is the responsibility of the database administrator, and not the ArcSDE software, which does not maintain statistics. You can update statistics for a table or feature class in ArcCatalog with the Analyze Component’s dialog box. It updates the statistics for the supporting tables that are associated with the selected object.

You should also update statistics on the SDE repository tables. This can be done with database management software. As a general rule of thumb, we suggest you update statistics at least weekly or monthly, and typically before and after a compress, which can be automated at the database level.

There is one exception: in situations where all users are editing just the SDE.DEFAULT version, you should just keep the statistics you collected before the compress. This will ensure that the query optimizer knows the delta tables are still active.

**2. Rebuild indexes on tables**

Indexes are used in a database to help speed up the retrieval of rows from a table, and they are also used by the database Query Optimizer when assessing query plans. As tables are modified by updates, inserts, and deletes of records, the corresponding indexes can become fragmented and unbalanced. This leads to increased I.O processing, which affects performance. This tip works in conjunction with the previous one. If you update statistics frequently, in turn you should consider rebuilding indexes if they are fragmented. Both actions will help improve performance.

In general, accurate statistics help to define a good index. You can assess the usefulness of an index with database management tools by monitoring its usage. Another benefit of rebuilding indexes is that you may reclaim disk space that was caused by its fragmentation. In versioned editing environments (where edits are performed daily), you may want to consider rebuilding indexes at regular intervals (for example, weekly or monthly), to keep performance degradation under control. We recommend you rebuild indexes after a compress. You can rebuild indexes within a database management program, or with ArcSDE commands.

**3. Plan parent-child version relationships carefully**

The versioning environment within an ArcSDE geodatabase enables users to implement and sustain complex business workflows. Typically the number of versions and how they are interrelated will depend on your business workflow. It is important to properly manage versions in the geodatabase, because poor version management will impact performance. You should keep the following in mind: every edit in the geodatabase is adding a state to the state tree. A state tree represents the total number of edits states stored in a geodatabase. Think of it conceptually like a flow chart diagram of circles and lines that flows from top to bottom. Each represents an edit state, and each state is linked by a line showing the edit history in the geodatabase.

A state tree, typically, has a structure similar to an upside-down tree, starting with one circle at the top (let’s say its zero), and flowing down in many branches. For example, a typical ArcSDE geodatabase may have approximately one million edits per day, resulting in hundreds of thousands of edit states in a state tree.

Ideally, you want to keep the state tree as simple and as small as possible. Versions are pointers to an edit state, and they will “pin” the state tree; in other words, they will keep its structure complicated. This can affect performance, because it may take queries longer to execute. Therefore, the more complex the versioning model (in other words, the more versions you have), means more potential records in the delta tables, which means potentially slower performance.

In general, you should try to do the following;

* Reconcile versions to the SDE.DEFAULT version as soon as you can.
* Delete versions when they are no longer needed.
* Avoid creating versions that will never be reconciled with SDE.DEFAULT.

You could also run multiple reconcile services, to reconcile without posting as many older versions as possible each evening. This operation will simplify the state tree, so that when a compress is finally executed, it will trim the state tree. Version management can be performed in the Version Management dialog box in ArcCatalog or ArcMap.

**4. Compress the geodatabase often**

Compressing an ArcSDE geodatabase helps maintain database performance by removing unused data.

Specifically it does two things:

* First, it removes unreferenced dates, and their associated delta table rows.
* Second, it moves entries in the delta tables that are common to all versions into the base tables, thus reducing the amount of data that the database searches through when executing queries. In effect, a compress will improve query performance and system response time by reducing the depth and complexity of the state tree.

When a large volume of uncompressed changes have accumulated in an ArcSDE geodatabase, a compress operation can take hours or even days. This is another very common cause of poor performance. To avoid this, you should compress on a regular basis (daily, weekly, and after periods of high editing activity). Users can stay connected to the geodatabase during a compress, but we suggest that all users be disconnected for the compress operation to be fully effective.

Remember to update statistics before and after a compress, and note the one exception mentioned earlier. The compress command is available in ArcCatalog. You add the command from the Customize dialog box, and you must be connected as the SDE user to execute it, or you could execute a compress with SDE commands.

**5. Monitor system resources**

When experiencing intermittent performance issues, it may be helpful to monitor the memory and CPU usage on both the client and server machines. This may help identify on which machine the performance bottleneck is occurring. For memory, it is important to ensure that the operating system is not running out of available memory and using *swap space* (in other words, *virtual memory*). Enterprise level ArcSDE typically needs at least one gigabyte of free disk space to operate efficiently. For CPU, you want to avoid and reduce how often the system hits a hundred percent CPU usage. Some troubleshooting suggestions to improve server performance include:

* Closing unrelated applications on the server
* Performing a database *trace* to examine and review performance (what’s in the database)
* You could have users switch from application server connections to *direct connects* (this will put more workload on the client and less on the server)

So, just to review, the performance of an ArcSDE geodatabase is influenced by many factors: hardware configuration, network configuration, network traffic, and the number of concurrent users.

Question

How can ArcSDE performance be improved?

Answer

ArcSDE performance can be improved by analyzing the spatial data and rebuilding their indexes. Analysis of tables and rebuilding of indexes are extremely important for regular maintenance of versioned Geodatabases. Performed regularly, these two functions can help keep performance of the versioned Geodatabase at a maximum.  
  
Below are scripts (for Oracle and SQL Server) that perform the following:  
  
1) Rebuild every index by any user that owns SDE data.  
2) Analyzes the schema of each user that owns SDE data.

**ArcGIS** is a [**geographic information system**](https://en.wikipedia.org/wiki/Geographic_information_system) (GIS) for working with maps and geographic information. It is used for: creating and using maps; compiling geographic data; analyzing mapped information; sharing and discovering geographic information; using maps and geographic information in a range of applications; and managing geographic information in a database.

**Asynchronous callbacks with the Web ADF controls.**

The Web ADF provides event handling and standard interfaces server-side controls, such as a Map and Toolbar, as well as a complete set of JavaScript libraries to initiate and process callbacks on the client. The Web ADF are already callback enabled by default. However, custom asynchronous callback solutions can be developed to initiate changes in map display and update other non-Web ADF components on a page. The sample provides the following examples:

 A custom tool in a **toolbar to center the map on a user click**. Custom CallbackResult objects are created to change the extent labels on all the edges of the map. The CallbackResult objects are added to the map control's CallbackResults collection and processed by Web ADF client-side JavaScript.

 A custom HTML client-side button to initiate a callback which centers and zooms in on a user entered point. The Page implements the ICallbackEventHandler interface to handle the callback request (RaiseCallbackEvent()) and send the callback response (GetCallbackResult()). The callback response is handled by the Web ADF JavaScript libraries.

### ASP.NET Callback Framework in ArcGIS Server II

I’m going to give two samples to illustrate how to combine callback with ADF controls.  
Now we know there are two most important things about callback. 1. the returned string, 2. the client javascript function to interpret the string and accordingly, update the client page partially. Sometimes it’s not easy to create a string and the accompanied js fucntion.  
Fortunately ArcGIS ADF provides both. It provides a generic purpose JavaScript function “processcallbackresult” to handle this. It’s on inetpub/wwwroot/aspclient/esri/javascript/dotnetadf.js.  
And every control has a property “callbackresults”, which could be converted into string and is understandable to the “processcallbackresult” function.  
So you don’t need to construct the returned string yourself. All you have to do is to send a callback request, do server side work related to the ADF web controls, send the callbackresults back and assign processcallbackresults as the JavaScript function to handle it.

Sample I.**Zoom to a selected state by a drop down list.**  
Please follow the steps:

1. Drag and drop a MapResourceManager control, a Map control and a HTML Select control on the page.
2. download the data from [here](http://www.utdallas.edu/~yxh038100/jian/USA.zip) , which includes a US states and a US cities featureclass.
3. Create an mxd file using ArcMap9.2. and publish it as a map service.
4. Set up the MapResourceManager control, pointing the MapResourceItem to the map service you just created. Set up the ArcGIS Identify in Web.Config file.

The complete code should look like:

<%@ Page Language="C#" AutoEventWireup="true"CodeFile="Default.aspx.cs" Inherits="\_Default" %>  
<%@ Register Assembly="ESRI.ArcGIS.ADF.Web.UI.WebControls, Version=9.2.3.1380, Culture=neutral, PublicKeyToken=8fc3cc631e44ad86"  
Namespace="ESRI.ArcGIS.ADF.Web.UI.WebControls" TagPrefix="esri" %>  
DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head runat="server">  
<title>Untitled Pagetitle>  
<script language="javascript" type="text/javascript">  
function queryFeature(selectedItem) {  
var message = selectedItem;  
var context = '';  
<%=sCallBackFunctionInvocation%>  
}  
script>  
head>  
<body>  
<form id="form1" runat="server">  
<div>  
<esri:MapResourceManager ID="MapResourceManager1" runat="server">  
<ResourceItems>  
<esri:MapResourceItem Definition="<Definition DataSourceDefinition="jian370" DataSourceType="ArcGIS Server Local" Identity="To set, right-click project and 'Add ArcGIS Identity'" ResourceDefinition="(default)@states" DataSourceShared="True" />"DisplaySettings="visible=True:transparency=0:mime=True:imgFormat=PNG8:height=100:width=100:dpi=96:color=:transbg=False:displayInToc=True"  
Name="MapResourceItem0" />  
ResourceItems>  
esri:MapResourceManager>  
div>  
<esri:Map ID="Map1" runat="server" Height="361px"MapResourceManager="MapResourceManager1" Width="487px">  
esri:Map>  
<select id="Select1" language="javascript"onchange="queryFeature(this.value);" style="width: 169px; height: 65px">  
<option selected="selected" value="California">Californiaoption>  
<option value="Texas">Texasoption>  
<option>option>  
select>  
form>  
body>  
html>

1. The server side code looks like:

using System;  
using System.Data;  
using System.Configuration;  
using System.Web;  
using System.Web.Security;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
using System.Web.UI.WebControls.WebParts;  
using System.Web.UI.HtmlControls;

public partial class \_Default : System.Web.UI.Page, ICallbackEventHandler  
{  
public string sCallBackFunctionInvocation;  
private string mapcallbackresults;  
protected void Page\_Load(object sender, EventArgs e)  
{  
sCallBackFunctionInvocation = Page.ClientScript.GetCallbackEventReference(this, "message","processCallbackResult", "context", "postBackError", true);  
}  
#region ICallbackEventHandler Members  
public string GetCallbackResult()  
{  
mapcallbackresults = Map1.CallbackResults.ToString();  
return mapcallbackresults;  
}  
public void RaiseCallbackEvent(string eventArgument)  
{  
ESRI.ArcGIS.ADF.Web.Geometry.Envelope envelope = newESRI.ArcGIS.ADF.Web.Geometry.Envelope();  
if (eventArgument == "California")  
{  
envelope.XMin = -124;  
envelope.YMin = 31;  
envelope.XMax = -114;  
envelope.YMax = 44;  
}  
else if (eventArgument == "Texas")  
{  
envelope.XMin = -107;  
envelope.YMin = 25;  
envelope.XMax = -93;  
envelope.YMax = 38;  
}  
Map1.Extent = envelope;  
}  
#endregion  
}

1. Run the project, select a state, the map will be zoomed into the area of selected state.

apparently, it has been implemented in the way of callback. Behind the scenes, the message indicates which state is my interest of area, on the server side, if the eventargument is california, create a new envelope around the state. And assign the new envelope as the map extent. Here is the important part. You have changed the map extent, the callbackresults include the information of the new extent. You even don’t need to know the detailed structure of the callback results, all you have to do is let the processcallbackresult do the client side job.

**Sample II. Update Map control and TOC control together.**  
Remember we talked about how to update two controls with one callback result string in part I. Here is a very common scenario, the sample will show how to add a new map resource item into the map control, at the same time, the TOC control will be updated. Please follow the steps:

1. Create two map services, one is US States, another one is US cities.
2. Put a MapResourceManager control, a Map control, a TOC control and an HTML button on the page.
3. Set up all those controls and the AGS Identity.
4. Double click the button and put the code below:

function Button1\_onclick() {  
var message = '';  
var context = '';  
<%=sCallBackFunctionInvocation%>  
}

5. On the server side, the code is:

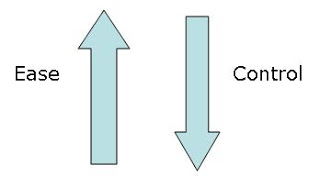
using System;  
using System.Data;  
using System.Configuration;  
using System.Web;  
using System.Web.Security;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
using System.Web.UI.WebControls.WebParts;  
using System.Web.UI.HtmlControls;  
using System.Text;  
public partial class \_Default : System.Web.UI.Page,ICallbackEventHandler  
{  
public string sCallBackFunctionInvocation;  
private string mapcallbackresults;  
protected void Page\_Load(object sender, EventArgs e)  
{  
sCallBackFunctionInvocation = Page.ClientScript.GetCallbackEventReference(this, "message","processCallbackResult", "context", "postBackError", true);  
}  
#region ICallbackEventHandler Members  
public string GetCallbackResult()  
{  
Toc1.Refresh();  
Map1.CallbackResults.AddRange(Toc1.CallbackResults);  
mapcallbackresults = Map1.CallbackResults.ToString();  
return mapcallbackresults;  
}  
public void RaiseCallbackEvent(string eventArgument)  
{  
ESRI.ArcGIS.ADF.Web.UI.WebControls.MapResourceItem mRI = newESRI.ArcGIS.ADF.Web.UI.WebControls.MapResourceItem();  
StringBuilder resDefBuilder = new StringBuilder();  
resDefBuilder.Append(" + "localhost" + "\"");  
resDefBuilder.Append(" DataSourceType=\"ArcGIS Server Local\"");  
resDefBuilder.Append(" Identity=\"\"");  
resDefBuilder.Append(" ResourceDefinition=\"(default)@" + "states" +"\"");  
resDefBuilder.Append(" DataSourceShared=\"True\"/>");  
mRI.Definition = newESRI.ArcGIS.ADF.Web.UI.WebControls.GISResourceItemDefinition(resDefBuilder.ToString());  
mRI.Parent = MapResourceManager1;  
mRI.Name = "USStates";  
mRI.InitializeResource();  
// Add resource info to map resource manager and create resources   
MapResourceManager1.ResourceItems.Add(mRI);  
MapResourceManager1.CreateResource(mRI);  
MapResourceManager1.Refresh();  
Map1.InitializeFunctionality(mRI);  
//refresh map to reflect the new resource  
Map1.Refresh();  
}  
#endregion  
}  
6. Run the project and click the button, you will see a new map resource item will be added into the map.

### ASP.NET Callback Framework in ArcGIS Server III

In the callback event, you don’t need to worry about the code, it just creates a new map resource item and add a map service into map resource manager, and refresh the new item in the map control. The GetCallbackRsult() is the fun part. Refresh the TOC, and you have to add the callbackresult from the TOC into the callbackresult of the map. Remember what we said before, there is only one string allowed to be sent back. So we need to combine them together. If this line of code “Map1.CallbackResults.AddRange(Toc1.CallbackResults);” is commented out, the TOC just doesn’t get updated, simply because the information is not sent back.

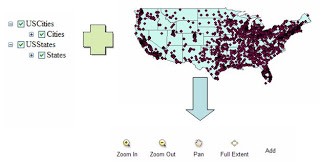
I was asked why don’t just copy the callbackresult from the map control to the TOC control. Yes, you can do that as long as the returned string includes all the information.

We talked about ASP.NET callback framework and how to make it work with ADF controls.  
Even though the ADF provides the ProcessCallbackResult function and the CallbackResults property for each web control, it’s still complicated to implement since we have to deal with the callback manually.  
You can create a simple web mapping application just by dropping Map control, TOC control and Toolbar control without any programming efforts. It makes life really easy, just use it. The controls themselves take care of every thing related to callback, including the request message, the returned string and the client JavaScript.

Now, we know there are three levels of callback. The most basic one, pure callback by implementing icallbackeventhandler for the page. You need to take care of the request message, the return string and the accompanied javascript function.[](http://1.bp.blogspot.com/_M1qv4oAeQaw/RzFd4Sn4NdI/AAAAAAAAAAk/9ncEfMk9rKc/s1600-h/3levels.JPG)  
The second level is to combine with ADF controls, you still need to implement the icallbackeventhandler interface for the page, but ADF provides the client JavaScript function ProcessCallbackResult(), and Those ADF web controls could return the callbackresults. You don’t need to construct it yourself. We showed some samples, such as when clicking a drop down list box, the map is zoomed into the corresponding state. You still need to make the callback request manually, and change the map extent, then the property of map.CallbackResults reflect the change. You need to explicitly send it back to the client and assign processcallbackresults() as the JavaScript.  
The third level is the AJAX enabled server control, such as Map control, Toolbar Control and so on. It wraps everything inside the control. When you click a command in a Toolbar control, it automatically sends the callback request message to the server, then gets the returned string back and updates the component.  
As the level goes up, it's easier to use, but less control to what happens under the hood. Sometimes it’s a nightmare to customize the control since everything is wrapped up, you cannot see the source code behind. I’m going to aim on how to customize those ADF controls.

Let’s start from Tool and command.  
**Custom Command**  
I rewrote the adding map resource item sample in a custom command. Click [here](http://www.utdallas.edu/~yxh038100/jian/customcommand.zip) to download the source code.  
1. In VS.NET 2005, open the unzipped project  
2. Now run the project > click the command > a new map resource item will be added into the map as the sample we showed previously.

You don’t need to send request yourself and it’s not necessary to get the returned string back. You even don’t need to specify the client JavaScript function.

[](http://1.bp.blogspot.com/_M1qv4oAeQaw/RzFehSn4NeI/AAAAAAAAAAs/wQUc_-H4QrI/s1600-h/combinecallback.JPG)But you need to append the TOC callback to the callback of the map control. Because map callback is the default callback result string that will be returned. In fact, the final returned callback string is the callback result of the toolbar. Although the system hide it, behind the scenes, the map callback result has been appended to the toolbar callback. The steps is you append toc callback results to map callback results, and the ADF automatiicaly append map callbackresults to the toolbar callback results.  
It unveils an critical rule: **ALWAYS COPY CALLBACKRESULTS FROM CHANGED CONTROL TO TARGET CONTROL.**We will see and apply this rule again and again.

Custom Tool  
This is an interesting sample. We are going to have some real fun here. The idea is simple though, I just want to have a tool that I can click on the map by this tool and put a text label at the point where I click. At the same time, the label text is from a textbox so that I can input the label. For example, I input "Texas" in the textbox and click on the area of Texas with the tool, a label "Texas" will show on the map. Please download the sample [here](http://www.utdallas.edu/~yxh038100/jian/customtool2.zip).

At first I thought it was going to be easy, all I have to do is to grab the text from a textbox and put it on the map. But that was a wrong idea, do you remember one of our previous samples? I put a textbox and input something, I was trying to get the text from the server but failed. Because the callback request doesn't send it. So, the exactly same logic here. I cannot get the text because when I click on the map, the callback request message is generated by the ADF. I have no control about what to send! Basically, the tool just send the coordinates x and y values.  
so finally, I found a workaround. When you type in something in the textbox and click somewhere else, the textbox just loses the focus, and there is an event can be triggered (this is an event on the client, instead of server side), textchanged. Basically, when you click enter key, tab key or just click other control, the event will be just fired. I create a JavaScript function to respond to the event, and put a callback request in this JavaScript, get the text and send the text as a request message. On the server side, the request message will be stored into session state. It means you can get it out anytime in the future. When I click on the map, I retrieve the text from session variables.  
So we send two rounds of callback at least. The first round, when text changes, it sends the text and saves it into session, the second round happens when click on the map with the tool. And we utilize two callback model here. One is by implementing ICallbackeventHandler, another one is the ESRI built-in callback. You can see that in the JavaScript function the process function is null and return string is null, because nothing visually needs to be udpated. It looks like nothing happens but something really happens behind the scenes.

Versioning in ArcSDE

Versioning allows multiple editors to alter the same data in an enterprise or workgroup geodatabase without applying locks or duplicating data.

To edit feature classes that participate in a topology, network dataset, or geometric network, or edit a parcel fabric, you must register the data as versioned. This is because when you edit a feature in a network, topology, parcel fabric, not all the features lock, which means other editors can edit another part of the network, topology, or parcel fabric in a way that conflicts with your edits.

You always access an enterprise or workgroup geodatabase through a version. When you connect, you specify the version to which you will connect. By default, you connect to the Default version.

## What is a version?

A version is a sort of view of the geodatabase that compartmentalizes edits. Versions allow you and all other users connected to the same version to see your changes. Uers connected to other versions do not see your changes until you reconcile and post them to an ancestor version.

### The Default version

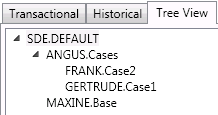
The Default version is the root version and, therefore, the ancestor of all other versions.

Unlike other versions, the Default version always exists and cannot be deleted. In most workflow strategies, it is the published version of the geodatabase, representing the current state of the system being modeled. You maintain and update the Default version over time by posting changes to it from other versions. You can also edit the Default version directly, just like any other version.

### Creating other versions

You create a version by creating children or branches from any existing version. You create the first version by making a child version of the Default version. When the new version is created, it is identical to the Default version. Over time, the versions diverge as changes are made to the Default version and the new version.

A geodatabase can have many versions. The following is the **Version Manager** dialog box, which is accessed through ArcGIS for Desktop. This example shows the tree view of the dialog box, depicting the Default version and four other versions, and how the versions are related. The Cases and Base versions are children of the Default version, and the Case1 and Case2 versions are children of the Cases version.



Creating a version gives you the impression that you are creating a copy of the entire geodatabase, but you are not. Regardless of how many versions you have, each table and feature class is stored only once in the database. ArcGIS leaves each feature class or table in its original format but records any changes in tables referred to as the delta tables.

Versions can be edited simultaneously by multiple users.

## How versions and versioned edits work

Before you can begin performing versioned edits on the data in any version, the datasets must be registered as versioned.

When you register a dataset (a feature class, feature dataset, or table) as versioned, two delta tables are created: the A (or adds) table, which records insertions and updates, and the D (or deletes) table, which stores deletions. Each time you update or delete a record in the dataset, rows are added to one or both of these tables. A versioned dataset, therefore, consists of the original table (referred to as the base or business table) plus any changes in the delta tables. The geodatabase keeps track of which version you were connected to when you made the edits that populated the delta tables. When you query or display a dataset in a version, ArcGIS assembles the relevant rows from the original table and the delta tables to present a seamless view of the data for that version.

All edits to the feature class or table are recorded in the same delta tables, regardless of the version from which the edits were made. This means that any one version references only a subset of rows from the three tables. So how does ArcGIS keep track of which rows in the delta tables belong to which version?

Each row in the A and D tables is marked with an integer identifier called a state ID that references when the row is added to the table. Every time you edit a version, a new state is created and a new row is added to one or both of the delta tables. States can be thought of as being part of a tree structure where each branch records how a version evolves. A sequence of states recording a series of changes from the base table to the current state of a version is called a lineage. When you display or query a version, ArcGIS queries the lineage of a version to get the state IDs, then retrieves the correct records from the A and D tables.

As a geodatabase is edited over time, delta tables increase in size and the number of states increases. The larger the tables and the more states, the more data ArcGIS must process every time you display or query a version. To maintain database performance, the geodatabase administrator must periodically run the [Compress](http://desktop.arcgis.com/en/arcmap/10.3/manage-data/gdbs-in-db2/geodatabase-compress-operation.htm) command to remove unused data

### The option to move edits to base

If you specify the option to move edits to base when you register your data as versioned, changes are recorded in the delta tables. However, when you edit the Default version and save your edits, the changes move from the delta tables to the base table—they do not remain in the delta tables. Note that this is only true when editing the Default version. Edits made to decendent versions are not immediately moved to the base table.

The option to move edits to the base table is useful if the following is true:

* The modifications you are making will take only a few minutes to complete.
* The data does not participate in a network or topology.
* You are connecting to a versioned geodatabase using a third-party application.

Third-party applications are generally set up to query only the base table—they can't access the delta tables. If you use versioning and don't choose to move the edits to the base table, these applications will not incorporate edits made in other versions that have not been reconciled and posted to the Default version. Be aware that as you are editing versions other than the Default, changes are recorded in the same delta tables. When you save, the changes remain in the delta tables. However, when you merge changes into the Default version, changes move from the delta tables to the base tables. Merging changes into versions other than the Default keeps changes in the delta tables, just as if you had not specified to move edits to base.

## Permissions and version edits

The owner of the version (the person who creates it) can set who can access the version. Access permission options are as follows:

* Private: Only the version owner can view and edit the datasets in that version.
* Protected: Any user can view the datasets in the version, but only the owner can edit them.
* Public: Any user can view and edit the datasets, provided he or she has been granted permission on the datasets.

Version access is set when the version is created, but it can also be changed on the **Version Manager** dialog box. See [Creating versions and setting access permissions](http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/creating-versions-and-setting-permissions.htm) and [Working with version properties](http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/version-properties.htm) for more information.

To edit, connect to a specific geodatabase version from ArcMap and add data that has been registered as versioned to the map.

##### **Tip:**

You can also switch the version to which they are connected in ArcMap. See [Changing versions in ArcMap](http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/changing-versions-in-arcmap.htm) for more information.

By default, all edit sessions in ArcMap are versioned edit sessions. Therefore, if you have versioned data in your map, you can start editing as soon as you open an edit session. To open an edit session, click **Start Editing** on the **Editor** drop-down list of the **Editor** toolbar.

Edits made to each version apply only to that version. The exception is schema changes. When you change the schema in a version—for example, add a new field to a table—the change applies to all other versions. Only data owners can alter the schema of a dataset.

When you are done editing, reconcile your changes with and post them to an ancestor version.

## Reconcile and post changes

Reconciling and posting operations integrate your changes into any version that is an ancestor of the version you are working in, such as the parent or Default version. When you reconcile, the changes in the version you are editing are compared with the version into which you want to merge them.

When you modify data in a version, no locks are applied to the data. Two editors working on the same data, either in the same version or a different version, can produce conflicts. A conflict occurs when a row differs in the two versions being compared. The reconciliation process shows you each conflict and allows you to choose which representation of the row to preserve.

In practice, editing conflicts should be rare because the volume of edits is small compared to the amount of geographic data involved. In correctly designed workflows, the cost of reconciling conflicts is minor when compared to the savings from not having to lock or check out features for the duration of the transaction.

Once you finish reconciling, you can post the changes. This applies the modifications you made into the other version. If you no longer need the version you posted from, you can delete it. Alternatively, you can edit it further and reconcile and post changes again.

##### **Tip:**

Instead of manually reconciling, you can use the [Reconcile Versions](http://desktop.arcgis.com/en/arcmap/10.3/tools/data-management-toolbox/reconcile-versions.htm) geoprocessing tool to reconcile multiple versions or [a Python script to batch reconcile and post versions](http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/using-python-scripting-to-batch-reconcile-and-post-versions.htm).

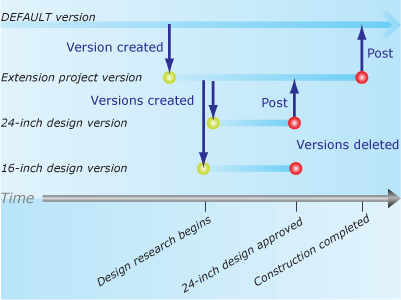
## Versions: An example

To illustrate how versions can be used, follow this scenario from a municipal water utility. The water utility has a geodatabase of features representing the current state of all the water pipes, valves, pumps, and other components of the water system. The utility needs to add a new line extension to the water system.

The utility creates a new version from the Default version called Extension project, which is to contain the design of the new extension. However, the utility staff are unsure whether to go with a 16-inch or a 24-inch pipe design for the new extension. So, from the Extension project version, they create a version to study the 16-inch design and another to study the 24-inch design.

They eventually discover the 24-inch pipe will serve projected water demand for 12 more years and that the greater initial construction cost is justified. The 24-inch design wins approval, is checked for accuracy, and is posted to the Extension project version.

A few months later, the construction of the new line extension is completed. To update the published version of the database, the Extension project version is reviewed for accuracy, reconciled, and posted to the Default version.



Enterprise **ArcSDE** geodatabases provide support for many users creating and maintaining large amounts of GIS data in a central location. In many cases, multiple users need to edit the same data at the same time. In other words, they require concurrent multiuser geodatabase editing.

Datum in Arc GIS

The **datum** is a integral part of the projection, as projected coordinated systems are based on geographic coordinates, which are in turn referenced to a **datum**. It is possible, and even common for datasets to be in the same projection, but be referenced to different **datums**, and therefore have different coordinate values.

Different types of Isolation in ArcGIS Server

Server objects run within processes on the container machines. Server objects can be configured such that they run in a dedicated process on the server, or they can be configured to run in processes they share with other server objects.

How they share processes is referred to as their isolation level. Server objects with **high isolation** do not share a process with other server objects. Each instance of a server object with high isolation has its own dedicated process on the server.

Server objects with **low isolation** can share processes with other server objects of the same type. Up to four server objects can share the same process.

When more than four server objects of a particular type (e.g., four RedlandsMap server objects) are created, an additional process is started for the next four server objects, and so on. As server objects are created and destroyed, they will vacate and fill spaces in these running processes. Instances of server objects whose isolation level is high require more resources on the server to run, as they require dedicated processes. Since instances of server objects with low isolation can share processes, they make more efficient use of server resources. However, isolation does have its benefits: since server objects with high isolation do not share processes, if an error occurs on the object causing its process to shut down or crash, it will not affect other server objects. However, if a server object is sharing its process with other server objects and the process is shut down or crashes, all the server objects in that process will be destroyed.

Pooled Service This means that instances of the service can be shared **between multiple application sessions.**

Tools are nearly the same as commands, except they require you to interact with the map canvas after they have been clicked. For example, the Split tool waits for you to select a point on the selected polyline feature, then breaks it into two features.

Other edit tools include the sketch tools, the Edit tool, and the Rotate tool.

To better illustrate the difference between a command and a tool, look at the Move command versus a tool that moves features. Once clicked, the Move command prompts you with a dialog box for a delta x and a delta y; after these values have been entered, the selected features are moved. Aside from entering values in a dialog box, you don't interact with the ArcMap canvas at all. In contrast, to reposition features with a move tool, such as the Edit tool, you must interactively drag selected features across the display.

Tool

Tools are nearly the same as commands, except they require you to **interact with the map canvas** after they have been clicked. For example, the Split tool waits for you to select a point on the selected polyline feature, then breaks it into two features.

Other edit tools include the sketch tools, the Edit tool, and the Rotate tool.

Command

Commands appear in numerous places in the editor framework, providing a large part of the editors interactive functionality; examples of editor commands include the Buffer, Intersect, and Union commands. As with any commands, an edit command does not require the user to interact with the Map after being clicked.

**Commands do not require you to interact with the map**; instead, commands often rely on the current state of the map.

Projected coordinate systems.

A **projected** coordinate **system** (PCS) is defined on a flat, two-dimensional surface. Unlike a GCS, a PCS has constant lengths, angles, and areas across the two dimensions. A PCS is always based on a GCS that is based on a sphere or spheroid.

**There are three types of Geodatabases:**

* File Geodatabases—Stored as folders in a file system. ...
* Personal Geodatabases—All datasets are stored within a Microsoft Access data file, which is limited in size to 2 GB.
* ArcSDE Geodatabases—Stored in a relational database using Oracle, Microsoft SQL Server, IBM DB2, or IBM Informix.

**Geocoding** is the process of **transforming a description of a location**—such as a pair of coordinates, an address, or a name of a place—to a location on the earth's surface. You can **geocode** by entering one location description at a time or by providing many of them at once in a table.

What is ADF ??

ADF is an internal raster data format used by ESRI products such as ArcGIS, ArcView, and ArcInfo Workstation. It stores spatial data as a binary grid and is one of several ADF files that together comprise the total grid. It is used to represent geographical or other spatial objects such as maps and map features

.

ArcGIS GIS Server service types

| **Service type** | **Required GIS resource** |
| --- | --- |
| Geocode service | Address locator (.loc, .mxs, SDE batch locator) |
| Geodata service | File geodatabase or database connection file (.sde) to a geodatabase |
| Geoprocessing service | Geoprocessing result from the **Results** window in ArcGIS Desktop |
| Image service (cached, dynamic) | Raster dataset or a layer file referencing a raster dataset |
| Map service (cached, dynamic) | Map document (.mxd) |
| Vector tile service | ArcGIS Pro vector tile package (.vtpk |

| **Capability** | **What it does** | **Services that expose this capability** |
| --- | --- | --- |
| Electronic Navigational Charts | Gives access to S-57 ENC, AML, IENC, and encrypted S-63 datasets anywhere for visualization and analysis. | Maritime Chart services |
| Feature Access | Provides access to vector features in a map. | Map services |
| Geocoding | Provides access to an address locator. This capability is always enabled when you publish a geocode service. | Geocode services |
| Geodata | Provides access to the contents of a geodatabase for data query, extraction, and replication. This capability is always enabled when you publish a geodata service. | Geodata services |
| Geoprocessing | Provides access to geoprocessing models. This capability is always enabled when you publish a geoprocessing service. | Geoprocessing services |
| Imaging | Provides access to the contents of a raster dataset or mosaic dataset, including pixel values, properties, metadata, and bands. This capability is always enabled when you publish an image service. | Image services |
| JPIP | Provides JPIP streaming capability when using JPEG 2000 or NITF (with JPEG 2000 compression) files and configured with a JPIP Server from [Exelis VIS](http://links.esri.com/ittvis/" \t "_blank). | Image services |
| KML | Uses a map document to create Keyhole Markup Language (KML) features. | Map services |
| Mapping | Provides access to the contents of a map, such as the layers and their underlying attributes. This capability is always enabled when you publish a map service. | Map services |
| Network Analysis | Solves transportation network analysis problems using the ArcGIS Network Analyst extension. | Map services |
| Schematics | Allows viewing, generating, updating, and editing schematic diagrams. | Map services |
| WCS | Creates a service compliant with the Open Geospatial Consortium, Inc. (OGC), Web Coverage Service (WCS) specification. | Map services, image services, geodata services |
| WFS | Creates a service compliant with the OGC Web Feature Service (WFS) specification. | Map services, geodata services |
| WMS | Creates a service compliant with the OGC Web Map Service (WMS) specification. | Map services, image services |
| WMTS\* | Creates a service compliant with the OGC Web Map Tile Service (WMTS) specification. | Map services, image services |
| WPS | Creates a service compliant with the OGC Web Processing Service (WPS) specification. | Geoprocessing services |

# IFeatureLayer Interface

Provides access to members that control common aspects of a feature layer.

# ILayer Interface

Provides access to members that work with all layers.

# IAGSServerObjectName Interface

Provides access to members that supply server object name information.

# IQueryFilter Interface

Provides access to members that filter data based on attribute values and or relationships.

# ISpatialFilter Interface

Provides access to members that return and modify the type of spatial relationship that the filter will use.

The **Georeferencing** toolbar allows you to **georeference** raster datasets, raster layers (which may have raster functions), image services, and raster products. In general, the steps for**georeferencing** a raster dataset are the following: In **ArcMap**, add the raster that you want to align with your projected data

The first thing you need to do before you attempt to connect to your GIS server is grant yourself administrative access to it. You do this by adding your operating system account to the agsadmin group, which contains a list of users who can administer the GIS server. If there are others in your organization that need to administer the GIS server, you can add their accounts as well.

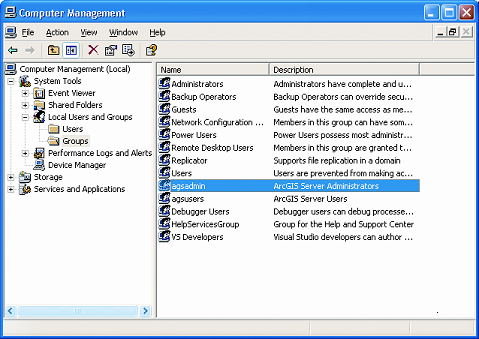
Don't worry if you don't have the complete list of administrators; you can always add and remove accounts later. It's important to note that the accounts you add to this group will be able to add, delete, and modify services running on the GIS server.

If you've added an account to the agsadmin group, you do not need to add it to the agsusers group.

1. On the SOM machine, start the Computer Management application. Computer Management can be found in the Control Panel under Administrative Tools.

2. Expand System Tools, then Local Users and Groups, then Groups.

3. Right-click the ArcGIS Server administrators group, named agsadmin, and click Properties.



4. On the Property page, click Add and, in the dialog box that appears, add your operating system account and any other accounts to which you want to grant administrative access to the GIS server. The account you add will be the account that you typically run ArcCatalog through.

5. Repeat steps 1 through 4 on each server object container machine.

The users of the accounts you add to the ArcGIS Server administrators group may need to log off and back on before the new settings take effect.

A CoClass is a class you can cocreate or instantiate calling the constructor (new.)  A class in ArcObjects lingo is a class that cannot be cocreated.  An inbound interface is a regular interface where calls executes code inside the class.  The call is made from outside the class to execute code inside the class (inbound.)  An outbound interface is an event interface, where events are raised by the class and invoke the event listener code outside of the class (code inside the class triggers code outside the class, outbound).

Objects and Classes  
ArcObjects consists of objects and classes.  
3  
An  
object  
represents a spatial feature such  
as a road or a vegetation stand. In a geodatabase, an object corresponds to a row in a  
table and the object�??s attributes appear in columns. A  
class  
is a set of objects with similar  
attributes. An ArcObjects class can have built-in interfaces, properties, and methods.  
ArcObjects includes three types of classes:  
The most common type is the coclass. A  
coclass  
can be used to create new objects.  
For example,  
FeatureClass  
is a coclass that allows new feature class objects to be  
created as instances of the coclass.  
The second type is the abstract class. An  
abstract  
class cannot be used to create new  
objects, but it exists so that other classes (i.e., subclasses) can use or share the  
properties and methods that the class supports. For example,  
GeoDataset  
is an  
abstract class. The class exists so that geographic datasets such as feature classes  
and raster datasets can all share the properties of extent and spatial reference that  
the  
GeoDataset  
class supports.  
  
The third type is the class. A  
class  
cannot be used directly to create new objects;  
instead, objects of a class can only be created from another class. For example,  
EnumInvalidObject  
is a noncreatable class because an  
EnumInvalidObject  
can only  
be obtained from another object such as a data conversion object. When converting  
a shapefile from one coordinate system to another, for example, a data conversion  
object automatically creates an  
EnumInvalidObject  
to keep track of those objects  
that have failed to be converted