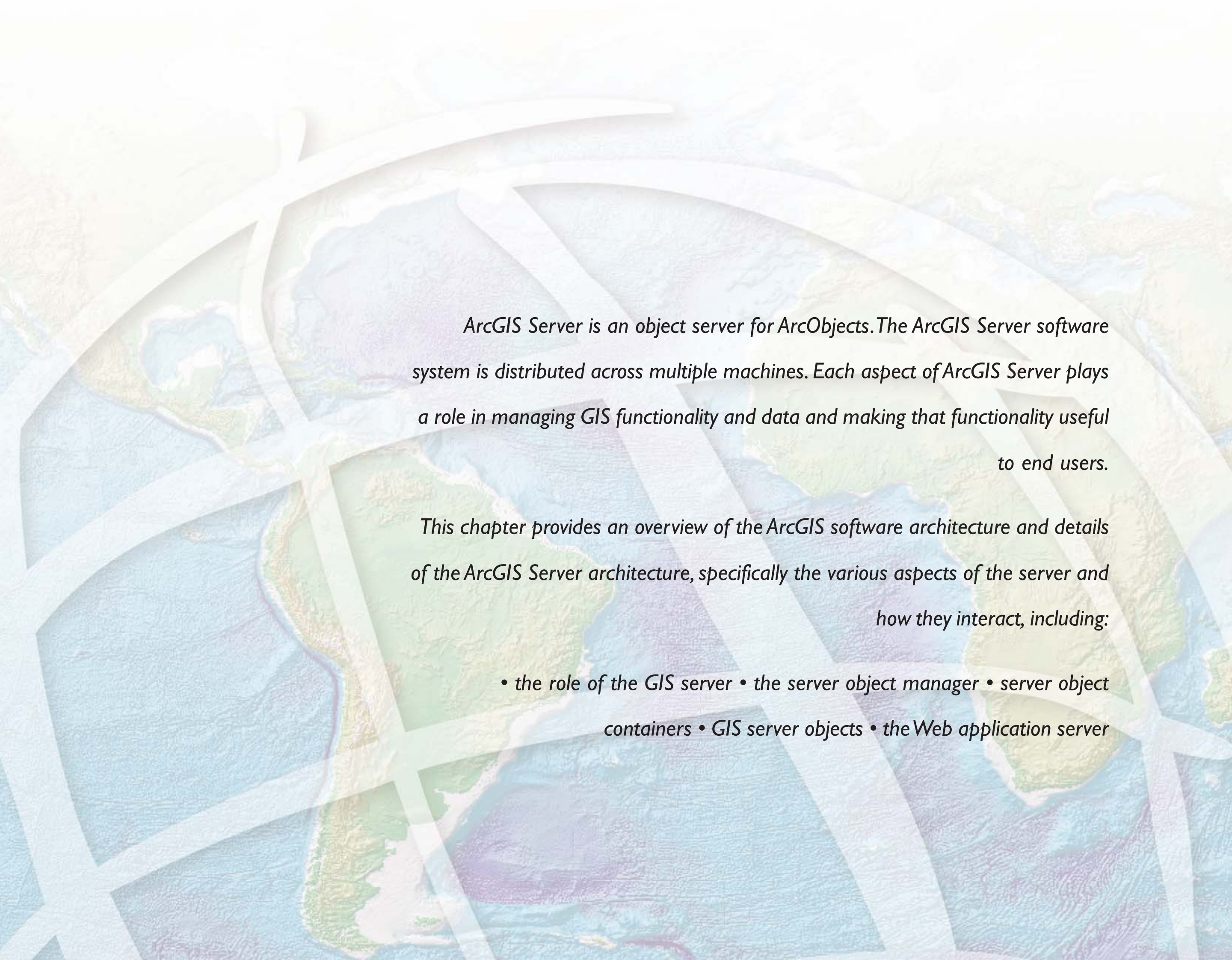
2 The ArcGIS Server architecture



ArcGIS Server is an object server for ArcObjects. The ArcGIS Server software system is distributed across multiple machines. Each aspect of ArcGIS Server plays a role in managing GIS functionality and data and making that functionality useful to end users.

This chapter provides an overview of the ArcGIS software architecture and details of the ArcGIS Server architecture, specifically the various aspects of the server and how they interact, including:

• the role of the GIS server • the server object manager • server object containers • GIS server objects • the Web application server

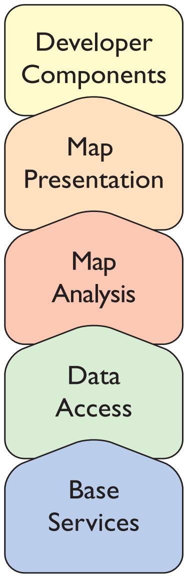
Before discussing the details of the ArcGIS Server architecture, it's important to discuss the ArcGIS system architecture as a whole. The ArcGIS architecture has evolved over several releases of the technology to be a modular, scalable, cross-platform architecture implemented by a set of software components called ArcObjects. This section focuses on the main themes of this evolution at ArcGIS 9 and introduces the reader to the libraries that compose the ArcGIS system.

The ArcGIS software architecture supports a number of products, each with its unique set of requirements. ArcObjects components, which make up ArcGIS, are designed and built to support this. This chapter introduces ArcObjects.

ArcObjects is a set of platform-independent software components, written in C++, that provides services to support GIS applications on the desktop in the form of thick and thin clients and on the server.

For a detailed explanation of COM see the Microsoft COM section of Appendix D, 'Developer environments'.

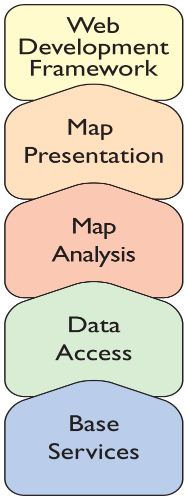
As stated, the language chosen to develop ArcObjects was C++; in addition to this language, ArcObjects makes use of the Microsoft Component Object Model. COM is often thought of as simply specifying how objects are implemented and built in memory and how these objects communicate with one another. While this is true, COM also provides a solid infrastructure at the operating system level to support any system built using COM. On Microsoft Windows operating systems, the COM infrastructure is built directly into the operating system. For operating systems other than Microsoft Windows, this infrastructure must be provided for the ArcObjects system to function.



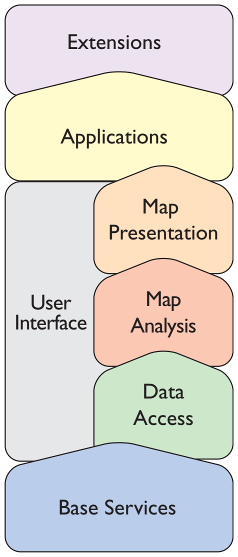
ArcGIS Engine

Not all ArcObjects components are created equally. The requirements of a particular object, in addition to its basic functionality, vary depending on the final end use of the object. This end use broadly falls into one of the three ArcGIS product families:

- ArcGIS Engine—Use of the object is within a custom application. Objects within the Engine must support a variety of uses; simple map dialog boxes, multithreaded servers, and complex Windows desktop applications are all possible uses of Engine objects. The dependencies of the objects within the Engine must be well understood. The impact of adding dependencies external to ArcObjects must be carefully reviewed, since new dependencies may introduce undesirable complexity to the installation of the application built on the Engine.
- ArcGIS Server—The object is used within the server framework, where clients of the object are most often remote. The remoteness of the client can vary from local, possibly on the same machine or network, to distant, where clients can be on the Internet. Objects running within the server must be scalable and thread safe to allow execution in a multithreaded environment.
- ArcGIS Desktop—Use of the object is within one of the ArcGIS Desktop applications. ArcGIS Desktop applications have a rich user experience, with applications containing many dialog boxes and property pages that allow end users to work effectively with the functionality of the object. Objects that contain properties that are to be modified by users of these applications should have property pages created for these properties. Not all objects require property pages.



ArcGIS Server



ArcGIS Desktop

Many of the ArcObjects components that make up ArcGIS are used within all three of the ArcGIS products. The product diagrams on these pages show that the objects within the broad categories of base services, data access, map analysis, and map presentation are contained in all three products. These four categories contain the majority of the GIS functionality exposed to developers and users in ArcGIS.

This commonality of function among all the products is important for developers to understand, since it means that when working in a particular category, much of the development effort can be transferred between the ArcGIS products with little change to the software. After all, this is exactly how the ArcGIS architecture is developed. Code reuse is a major benefit of building a modular architecture, but code reuse does not simply come from creating components in a modular fashion.

The ArcGIS architecture provides rich functionality to the developer, but it is not a closed system. The ArcGIS architecture is extendable by developers external to ESRI. Developers have been extending the architecture for a number of years, and the ArcGIS 9 architecture is no different; it, too, can be extended. However, ArcGIS 9 introduces many new possibilities for the use of objects created by ESRI and you. To realize these possibilities, components must meet additional requirements to ensure that they will operate successfully within this new and significantly enhanced ArcGIS system. Some of the changes from ArcGIS 8 to ArcGIS 9 appear superficial, an example being the breakup of the type libraries into smaller libraries. That, along with the fact that the objects with their methods and properties that were present at 8.3 are still available at 9.0, masks the fact that internally ArcObjects has undergone some significant work.

The main focus of the changes made to the ArcGIS architecture at 9.0 revolves around four key concepts:

- Modularity—A modular system where the dependencies between components are well-defined in a flexible system.
- Scalability—ArcObjects must perform well in all intended operating environments, from single user desktop applications to multiuser/multithreaded server applications.
- Multiple Platform Support—ArcObjects for the Engine and Server should be capable of running on multiple computing platforms.
- Compatibility—ArcObjects 9 should remain equivalent, both functionally and programmatically, to ArcObjects 8.3.

MODULARITY

The esriCore object library, shipped as part of ArcGIS 8.3, effectively packaged all of ArcObjects into one large block of GIS functionality; there was no distinction between components. The ArcObjects components were divided into smaller groups of components, these groups being packaged in Dynamic Link Libraries (DLLs). The one large library, while simplifying the task of development for external developers, prevented the software from being modular. Adding the type information to all the DLLs, while possible, would have greatly increased the burden on external developers and, hence, was not an option. In addition, the DLL structure did not always reflect the best modular breakup of software components based on functionality and dependency.

ESRI has developed a modular architecture for ArcGIS 9 by a process of analyzing features and functions and matching those with end user requirements and deployment options based on the three ArcGIS product families. Developers who have extended the ArcGIS 8 architecture with custom components are encouraged to go through the same process to restructure their source code into similar modular structures.

There is always a trade-off in performance and manageability when considering architecture modularity. For each criterion, thought is given to the end use and the modularity required for support. For example, the system could be divided into many small DLLs with only a few objects in each. Although this provides a flexible system for deployment options, at minimum memory requirements, it would affect performance due to the large number of DLLs being loaded and unloaded. Conversely, one large DLL containing all objects is not a suitable solution either. Knowing the requirements of the components allows them to be effectively packaged into DLLs.

An obvious functionality split to make is user interface and nonuser interface code. UI libraries tend to be included only with the ArcGIS Desktop products.

The ArcGIS 9 architecture is divided into a number of libraries. It is possible for a library to have any number of DLLs and executables (EXEs) within it. The requirements that components must meet to be within a library are well-defined. For instance, a library, such as Geometry (from the base services set of modules), has the requirements of being thread safe, scalable, without user interface (UI) components, and deployable on a number of computing platforms. These requirements are different from libraries, such as ArcMap (from the applications category), which does have user interface components and is a Windows-only library.

All the components in the library will share the same set of requirements placed on the library. It is not possible to subdivide a library into smaller pieces for distribution. The library defines the namespace for all components within it and is seen in a form suitable for your chosen ArcObjects API.

- Type Library—COM
- .NET Interop Assembly—.NET
- Java Package—Java
- Header File—C++

SCALABILITY

The ArcObjects components within ArcGIS Engine and ArcGIS Server must be scalable. Engine objects are scalable because they can be used in many different types of applications; some require scalability, while others do not. Server objects are required to be scalable to ensure that the server can handle many users connecting to it, and as the configuration of the server grows, so does the performance of the ArcObjects components running on the server.

The scalability of a system is achieved using a number of variables involving the hardware and software of the system. In this regard, ArcObjects supports scalability with the effective use of memory within the objects and the ability to execute the objects within multithreaded processes.

Thread safety refers to concurrent object access from multiple threads.

There are two considerations when multithreaded applications are discussed: thread safety and scalability. It is important for all objects to be thread safe, but simply having thread-safe objects does not automatically mean that creating multithreaded applications is straightforward or that the resulting application will provide vastly improved performance.

The ArcObjects components contained in the base services, data access, map analysis, and map presentation categories are all thread safe. This means that application developers can use them in multithreaded applications; however, programmers must still write multithreaded code in such a way as to avoid application failures due to deadlock situations and so forth.

In addition to the ArcObjects components being thread safe for ArcGIS 9, the apartment threading model used by ArcObjects was analyzed to ensure that ArcObjects could be run efficiently in a multithreaded process. A model referred to as "Threads in Isolation" was used to ensure that the ArcObjects architecture is used efficiently.

The classic singleton per process model means that all threads of an application will still access the main thread hosting the singleton objects. This effectively reduces the application to a single-threaded application.

This model works by reducing cross-thread communication to an absolute minimum or, better still, removing it entirely. For this to work, the singleton objects at ArcGIS 9 were changed to be singletons per thread and not singletons per process. The resource overhead of hosting multiple singletons in a process was outweighed by the performance gain of stopping cross-thread communication where the singleton object is created in one thread, normally the Main single-threaded apartment (STA), and the accessing object is in another thread.

ArcGIS is an extensible system, and for the Threads in Isolation model to work, all singleton objects must adhere to this rule. If you are creating singleton objects as part of your development, you must ensure that these objects adhere to the rule.

MULTIPLE PLATFORM SUPPORT

As stated earlier, ArcObjects components are C++ objects, meaning that any computing platform with a C++ compiler can potentially be a platform for ArcObjects. In addition to the C++ compiler, the platform must also support some basic services required by ArcObjects.

Microsoft Windows is a little endian platform, while Sun Solaris is a big endian platform.

Although many of the platform differences do not affect the way in which ArcObjects components are developed, there are areas where differences do affect the way code is developed. The byte order of different computing architectures varies between little endian and big endian. This is most readily seen when objects read and write data to disk. Data written using one computing platform will not be compatible if read using another platform, unless some decoding work is performed. All the ArcGIS Engine and ArcGIS Server objects support this multiple platform persistence model. ArcObjects components always persist themselves using the little endian model; when the objects read persisted data, it is converted to the appropriate native byte order. In addition to the byte order differences, there are other areas of functionality that differ between platforms; the directory structure, for example, uses different separators for Windows and UNIX—'\' and '/', respectively. Another example is the platform-specific areas of functionality, such as Object Linking and Embedding Database (OLE DB).

COMPATIBILITY

While the aim of ArcGIS releases is to limit the change in the APIs, developers should still test their software thoroughly with later releases.

Maintaining compatibility of the ArcGIS system between releases is important to ensure that external developers are not burdened with changing their code to work with the latest release of the technology. Maintaining compatibility at the object level was a primary goal of the ArcGIS 9 development effort. Although this object-level compatibility has been maintained, there are some changes between the ArcGIS 8 and ArcGIS 9 architectures that will affect developers, mainly related to the compilation of the software.

Although the changes required for software created for use with ArcGIS 8 to work with ArcGIS 9 are minimal, it is important to understand that to realize any existing investment in the ArcObjects architecture at ArcGIS 9, you must review your developments with respect to ArcGIS Engine, ArcGIS Server, and ArcGIS Desktop.

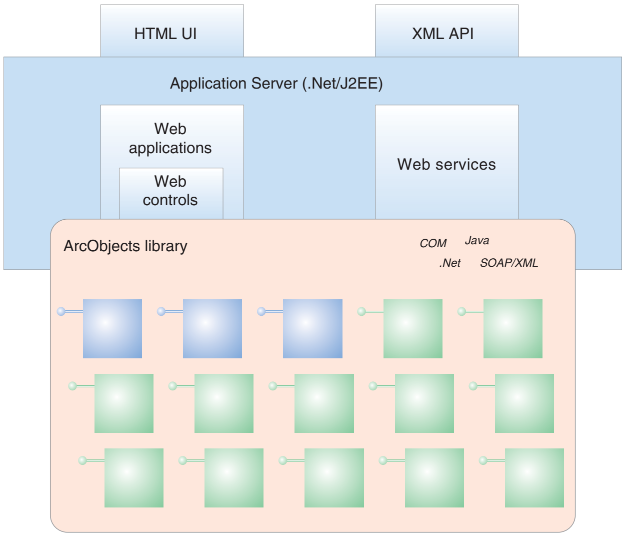
ESRI understands the importance of a unified software architecture and has made numerous changes for ArcGIS 9 so the investment in ArcObjects can be realized on multiple products. If you have been involved in creating extensions to the ArcGIS architecture for ArcGIS 8, you should think about how the new ArcGIS 9 architecture affects the way your components are implemented.

The remainder of this chapter focuses on the components of ArcGIS Server that make it possible to run ArcObjects in a server environment. For more information about developing with ArcGIS Engine and ArcGIS Desktop, refer to *ArcGIS Engine Developer Guide* and *ArcGIS Desktop Developer Guide*, respectively.

ArcGIS Server is fundamentally an object server that manages a set of GIS server objects. These server objects are software objects that serve a GIS resource such as a map or a locator. Developers make use of server objects in their custom applications. Server objects are ArcObjects. ArcObjects is a collection of software objects that make up the foundation of ArcGIS.

ArcObjects components have multiple developer application programming interfaces. These include COM, .NET, Java, and C++. Developers can use these APIs to build applications that make use of ArcObjects functionality.

ArcObjects is at the core of all the ArcGIS products: ArcGIS Desktop, ArcGIS Engine, and ArcGIS Server. ArcGIS Server adds the framework for running ArcObjects in a server. ArcGIS Server also provides a framework for developers to build advanced GIS Web services and Web applications using ArcObjects in standard application server frameworks such as .NET and J2EE.



As described above, at the core of ArcGIS Server is a rich ArcObjects library that can be exploited in Web applications and Web services to deliver advanced GIS functionality to a wide range of users who interact with the server through Web browsers and other thin client applications. Web applications that run in the Web server use distributed object technology to communicate with ArcObjects components, which are themselves COM objects, running in the GIS server over a local area network or wide area network (WAN).

Distributed object technology allows applications running in one process to use COM objects that are running in another process seamlessly using local interprocess communication. Distributed object technology also allows applications to use COM objects running on other machines by using network protocols to provide the communication between the client and COM server. This allows the Web application or Web service developer to make use of remote ArcObjects running in the server using the same programming interfaces a developer writing a desktop application with ArcObjects would use.

A proxy object is a local representation of a remote object. The proxy object controls access to the remote object by forcing all interaction with the remote object to be via the proxy object. The supported interfaces and methods on a proxy object are the same as those supported by the remote object. You can make method calls on and get and set properties of a proxy object as if you were working directly with the remote object.

ArcObjects include APIs for both .NET and Java. These APIs allow developers to write Web applications using both .NET and Java and to host those Web applications in standard server frameworks such as .NET and J2EE. To facilitate the development of such applications, the ArcGIS Server ADF includes a set of Web controls and ArcObjects proxies for both .NET and Java around which to build GIS Web applications. These Web applications can deliver advanced GIS functionality to the end user through browsers.

ArcGIS Server also includes a Simple Object Access Protocol (SOAP) toolkit that contains ArcObjects components that support SOAP request handling via an extensible markup language (XML) API. Once an ArcObjects component is exposed as a Web service, this XML API allows applications to remotely use that object running in the server over standard Internet protocols.

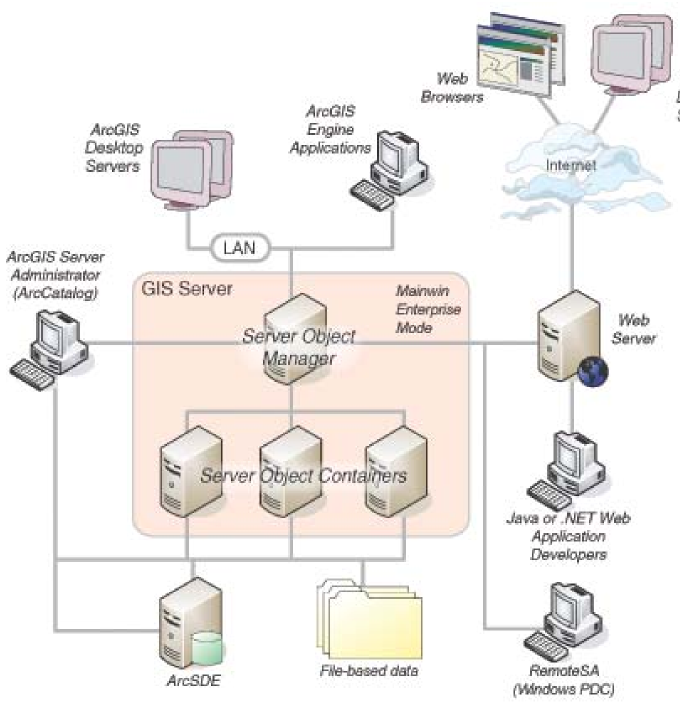
The remainder of this chapter explores how the different pieces of the ArcGIS Server work together to make the use of ArcObjects in server applications possible. Detail is given to the ArcGIS Server programming model in Chapter 4, ‘Developing ArcGIS Server applications’, and the application developer frameworks are discussed in detail in Chapters 5, ‘Developing Web applications with .NET’, and 6, ‘Developing Web applications with Java’.

ArcGIS Server is a distributed system consisting of several components that can be distributed across multiple machines. Each component in the ArcGIS Server system plays a specific role in the process of managing, activating, deactivating, and load balancing the resources that are allocated to a given server object or set of server objects.

The components of ArcGIS Server can be summarized as:

- GIS server—Hosts and runs server objects. The GIS server consists of a server object manager (SOM) and one or more server object containers (SOCs).
- Web server—Hosts Web applications and Web services that use the objects running in the GIS server.
- Web browsers—Used to connect to Web applications running in the Web server.
- Desktop applications—Connect over HyperText Transfer Protocol (HTTP) to ArcGIS Web services running in the Web server or connect directly to GIS servers over a LAN or WAN.

The ArcGIS Server is a distributed system that consists of a server object manager, server containers, and clients to the server, such as desktop and Web applications.



When running the GIS server on UNIX, Visual Mainwin in Enterprise mode provides Distributed Component Object Model (DCOM) support for the server on UNIX.

When running the GIS server on UNIX, Visual Mainwin's Remote Security Authority (RemoteSA), a Windows service, needs to be set up on a Windows machine to provide authentication services for the GIS Server and all machines connecting to it. This service and related documentation are available on the ArcGIS Server CD in the support\remotesa directory.

THE GIS SERVER

The GIS server, responsible for hosting and managing server objects, is the set of objects, applications, and services that makes it possible to run ArcObjects components on a server. Before describing the various aspects of the GIS server, server objects and the role of ArcObjects in ArcGIS Server will be defined.

A server object is a software object that manages and serves a GIS resource such as a map or a locator. For example, a server object named RedlandsMap may serve a map document of data for the city of Redlands, while the server object RedlandsGeocode may serve an address locator for geocoding addresses. ArcGIS Server objects are themselves ArcObject components.

Server objects are managed and run within the GIS server. A server object may be preconfigured and preloaded in the server and can be shared between applications. Server applications make use of server objects and may also use other ArcObjects that are installed on the GIS server.

The server object manager

The GIS server is composed of a SOM, which is a Windows service or UNIX daemon running on a single machine, and SOCs, which run on one or more machines (container machines). 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, so the parameter that is provided for the connection to be made is the name or Internet Protocol (IP) address of the SOM machine.

The server object containers

The container machine or machines actually host the server objects that are managed by the SOM. Each container machine is capable of hosting multiple container processes. A container process is a process in which one or more server objects is running. Container processes are started and shut down by the SOM. The objects hosted within the container processes are ArcObjects components that are installed on the container machine as part of the installation of ArcGIS Server.

All server objects run on all container machines and are balanced equally across all container machines. So, it's important that all container machines have access to the resources and data necessary to run each server object. It's also important to note that the GIS server assumes that all container machines are configured equally, such that they are all capable of hosting the same number of server objects. Server object resources and data are discussed in more detail in the next section.

The server directories

A server directory is a location on a file system. The GIS server is configured to clean up any files it writes to a server directory. By definition, a server directory can be written to by all container machines.

The GIS server hosts and manages server objects and other ArcObjects components for use in applications. In many cases, the use of those objects requires writing output to files. For example, when a map server object draws a map, it writes images to disk on the server machine. Other applications may write their own data; for example, an application that checks out data from a geodatabase may write the checkout personal geodatabase to disk on the server.

Typically, these files are transient and need only be available to the application for a short time—for example, the time for the application to draw the map or the time required to download the checkout database. As applications do their work and write out data, these files can accumulate quickly. The GIS server will automatically clean up its output if that output is written to a server directory.

A server directory can be configured such that files created by the GIS server in it are cleaned based on either file age or time since they were last accessed. The maximum file age is a property of a server directory. All files created by the GIS server that are older than or have not been accessed for the time defined by the maximum age are automatically cleaned up by the GIS server.

The Remote Security Authority (GIS Server on UNIX only)

A Windows machine is required to act as Primary Domain Controller (PDC) to authenticate users of a GIS Server on UNIX.

Visual Mainwin's Remote Security Authority, a Windows service, needs to be set up on a Windows machine to provide authentication services for the GIS Server and all machines connecting to it. This service and related documentation are available on the ArcGIS Server CD in the support\remotesa directory.

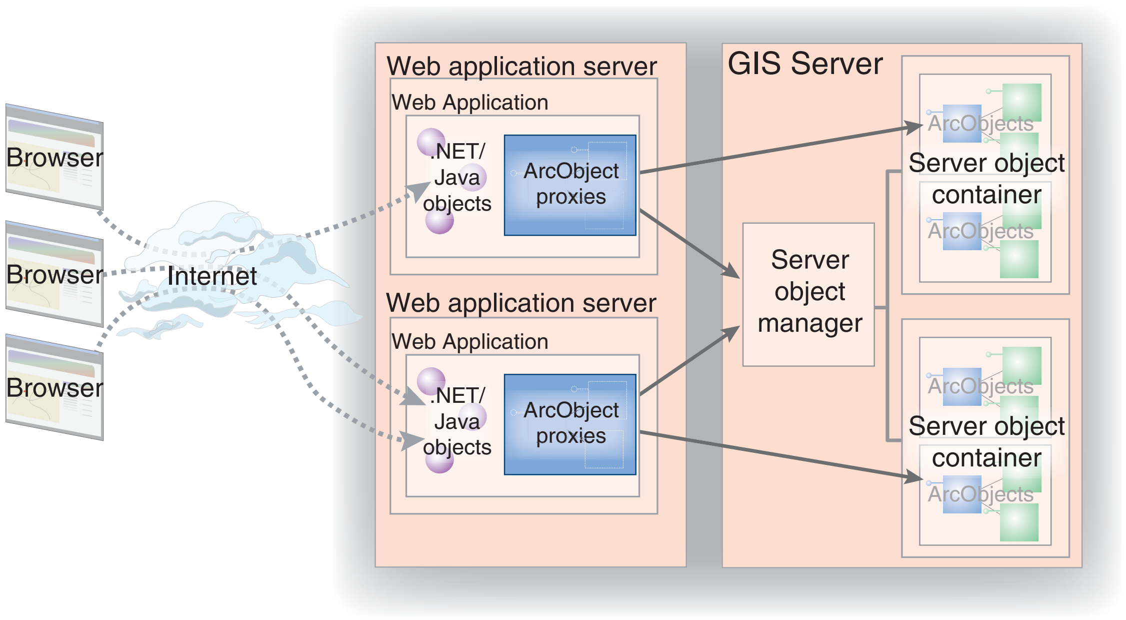
Visual Mainwin in Enterprise mode (GIS Server on UNIX only)

Visual Mainwin in Enterprise mode provides DCOM support for the GIS Server on UNIX. This download and related documentation are available on the ArcGIS Server CD in the support\msc directory.

THE WEB SERVER (UNIX-SPECIFIC INFORMATION)

The Web server hosts server applications and Web services written using the ArcGIS Server API. These server applications use the ArcGIS Server API to connect to a SOM, make use of server objects, and create other ArcObjects for use in their applications.

These Web services and Web applications can be written using the ArcGIS Server Application Developer Framework, which is available for both .NET and Java developers. Examples of Web applications include mapping applications, disconnected editing applications, and any other application that makes use of ArcObjects and is appropriate for Web browsers.



Web applications running in the Web server connect to the SOM and work with proxies to objects running within containers in the GIS server.

Examples of Web services include Web services for exposing map and geocode server objects that desktop GIS users can connect to and consume over the Internet. It is possible to create your own native .NET or Java Web services whose parameters are not ArcObjects types, but do perform a specific GIS function. For example, it is possible to write a Web service called FindNearestHospital that accepts x,y coordinates as input and returns an application-defined Hospital object that has properties such as the address, name, and number of beds.

A more detailed description of the application developer's framework and how it is used to create Web services and Web applications is given later in this book.

Web applications connect to GIS servers within their organization over the LAN. In this sense, the Web application or Web service is a client of the GIS server. Users connect to Web applications and Web services over the Internet or Intranet, but all of the Web application's logic runs in the Web server and sends hypertext markup language (HTML) to the browser client. The Web application itself makes use of objects and functionality running within the GIS server. This allows the development of Web applications to make use of ArcObjects in the server as would a desktop application connecting to the GIS server in client/server mode over the LAN or WAN.

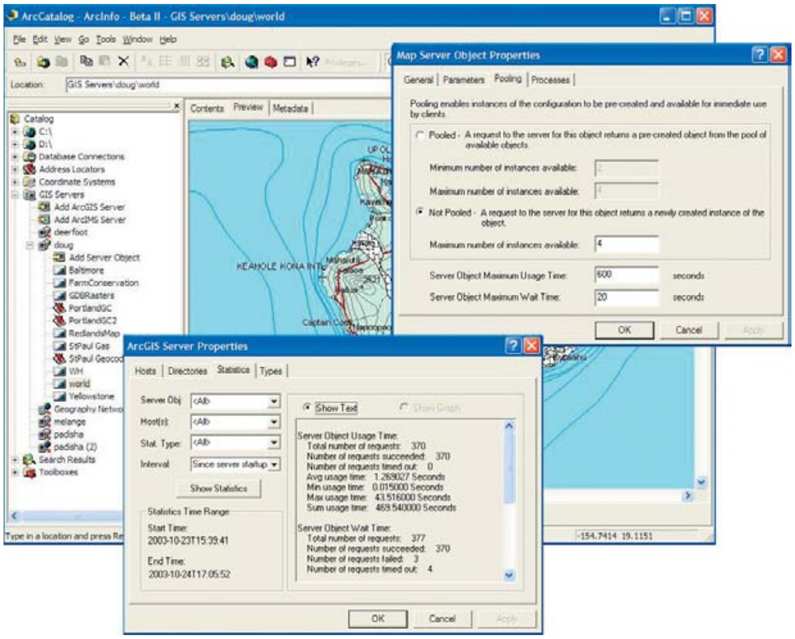
As users interact with their browser, it makes requests to the Web application, which in turn makes requests on the SOM. The SOM hands back a proxy to a server object or server objects that are running within the GIS server. The Web application uses the proxy to work with the object as if it existed in the Web application's process, but all execution happens on the GIS server.

Web and desktop server application development will be discussed in greater detail in later chapters of this book.

ARCGIS DESKTOP APPLICATIONS

Users can connect to ArcGIS Server using ArcGIS Desktop applications to make use of map and geocode server objects running in the server. Users can use ArcCatalog to connect to a GIS server directly on the LAN or WAN. They 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.

Additionally, the set of server objects and their properties are managed by the GIS server administrator using ArcCatalog. Administrators can connect to the GIS server over the LAN/WAN and use ArcCatalog to add and remove map and geocode server objects as well as configure how server objects should be run, including the set of container machines that are available for the server and the directories on the server they can use to write any output.



Map Server Object Properties

General | Parameters | Pooling | Processes

Pooling enables instances of the configuration to be pre-created and available for immediate use by clients

( ) Pooled - A request to the server for this object returns a pre-created object from the pool of available objects

Minimum number of instances available: 2

Maximum number of instances available: 4

(●) Not Pooled - A request to the server for this object returns a newly created instance of the object.

Maximum number of instances available: 4

Server Object Maximum Usage Time: 600 seconds

Server Object Maximum Wait Time: 20 seconds

OK Cancel Apply

ArcGIS Server Properties

Hosts | Directories | Statistics | Types

Server Obj: <All>
Host(s): <All>
Stat. Type: <All>
Interval: Since server startup

Show Statistics

Statistics Time Range
Start Time:
2003-10-23T15:39:41
End Time:
2003-10-24T17:05:52

(●) Show Text ( ) Show Graph

```
Server Object Usage Time:
   Total number of requests:   370
   Number of requests succeeded:   370
   Number of requests timed out:   0
   Avg usage time:   1.269027 Seconds
   Min usage time:   0.015000 Seconds
   Max usage time:   43.516000 Seconds
   Sum usage time:   469.540000 Seconds

Server Object Wait Time:
   Total number of requests:   377
   Number of requests succeeded:   370
   Number of requests failed:   3
   Number of requests timed out:   4
```

OK Cancel Apply

Users can connect to ArcGIS Server using ArcGIS Desktop applications to consume and administer server objects.

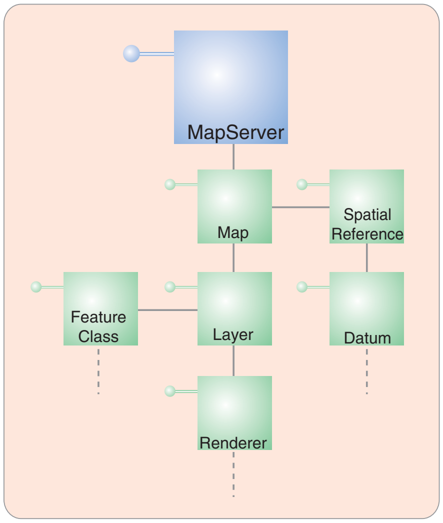
As described earlier in this chapter, a server object is a software object that manages and serves a GIS resource such as a map or a locator. Server objects are managed by the GIS server and run within processes on container machines.

A server object is simply a coarse-grained ArcObjects component, that is, a high-level object that simplifies the programming model for doing certain operations and hides the fine-grained ArcObjects that do the work. These coarse-grained objects allow clients to perform large units of work, such as drawing a map or geocoding a table of addresses, using a single method call. These coarse-grained objects use the finer-grained ArcObjects components on the server to draw the map and geocode the addresses.

Server objects also have SOAP interfaces for handling SOAP requests to execute methods and returning results as SOAP responses. This support for SOAP request handling makes it possible to expose server objects as Web services that can be consumed by clients across the Internet.

ArcGIS Server for 9.0 includes two coarse-grained server objects: the GeocodeServer and the MapServer.

Note that a server object also has other associated objects that a developer can get to and make use of; for example, a developer working with a MapServer object can get to the Map and Layer objects associated with that map. These are the same Map and Layer objects that a desktop or engine developer would work with, except they live in the server. This is discussed in much greater detail in Chapter 4, 'Developing ArcGIS Server applications'. For now, the concentration will be on how the MapServer and GeocodeServer objects are managed.



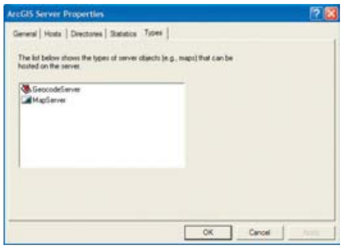
A server object is a coarse-grained ArcObjects component that has other ArcObjects components associated with it.

A server object, unlike other ArcObjects components, can be preconfigured by a GIS server administrator. Once these server objects are preconfigured, they can be used by developers and end users who can connect to the server through ArcGIS Desktop applications.

When a server object is configured by the server administrator to run in a GIS server, the following configuration aspects of the server object need to be specified:

- Name of the server object
- Type of server object
- The initialization data and parameters for the server object
- Whether or not the object is pooled and the minimum and maximum number of instances that can be running
- How long a client can wait for and use a server object
- The isolation level of the server object
- Whether the object is recycled

Each of these aspects of server objects will be described in more detail.



The list below shows the types of server objects (e.g., maps) that can be hosted on the server.

GeocodeServer
MapServer

The GIS server includes the MapServer and GeocodeServer server object types.

SERVER OBJECT TYPE

All server objects have a type, which dictates what its initialization parameters are and what methods and properties it exposes to developers. At ArcGIS 9, there are two server object types: the MapServer, which is in the Carto library, and the GeocodeServer, which is in the Location library.

INITIALIZATION DATA AND PARAMETERS

As already stated, a GIS server object manages and serves a GIS resource. When a server object is configured, this resource and other required parameters associated with the server object must be specified such that when a server object is initialized, it knows what resource to bind to.

The MapServer server object's initialization data is the map document (.mxd) or published map document (.pmf) that it's going to serve. When instances of a particular MapServer object are initialized on the server, the map document is loaded.

The GeocodeServer server object's initialization data is the address locator that it will use to perform address matching against. The address locator may be a locator file (.loc), an ArcView 3 address locator (.mxs), or an ArcSDE address locator. In addition to the address locator, the GeocodeServer also has a batch size parameter that indicates the number of records it will process at a time when doing batch geocoding.

Note that when configuring the initialization data for a server object, both the resource (that is, map document or locator) and the data that the resource references are accessible by the GIS server's container machines. This becomes an especially important consideration when the GIS server has multiple container machines. In these cases, the location of both the resource and the data it uses must be on a shared file system (in the case of file-based information) or on an ArcSDE server that all the container machines can connect to.

For example, a map document that acts as a resource to a MapServer object hosted on multiple container machines must be located on a shared network drive. All of the data for the various layers in the map that reference file-based data must also be on a shared network drive, and any layers that reference data in an ArcSDE geodatabase must be able to connect to that database.

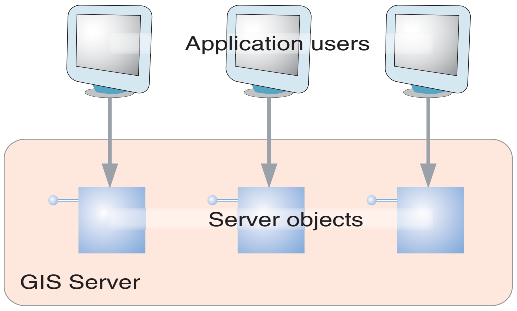
Note that server objects can run on any of the container machines configured in the GIS server. All container machines must have access to the data and resources needed for a particular server object.

SERVER OBJECT POOLING

A server object that is running on the GIS server is available for use by users who connect to the server through ArcGIS Desktop or by developers who create applications that connect to and make use of the server. Any server object may be used by a number of different users. For example, a MapServer object containing features from a land records database may be used in an application that users query for information about their land parcel. The same MapServer may also be used by editors who update the database in a disconnected editing application.

A server object may be configured to be pooled or non-pooled. Non-pooled server objects are created new for each application use and destroyed when released by the application to the server. The creation of the object includes creating the object and loading up any initialization data, such as the map document associated with a MapServer server object. Each user of a server application that makes use of a non-pooled server object requires an instance of that object dedicated to the user's application.

The number of users on the system at any one time has a 1:1 correlation with the number of running server object instances, so the number of concurrent users the GIS server can support is equal to the number of server objects that it can support effectively at any one time. When configuring a server object to be non-pooled, the maximum number of instances can be limited by specifying it as a property of the configuration. Once the maximum number of instances has been reached (that is, the maximum number of concurrent users of that server object), additional users will be queued until the number of users drops below the maximum.



The number of non-pooled server objects running in the GIS server is 1:1 with the number of application users making use of that server object.

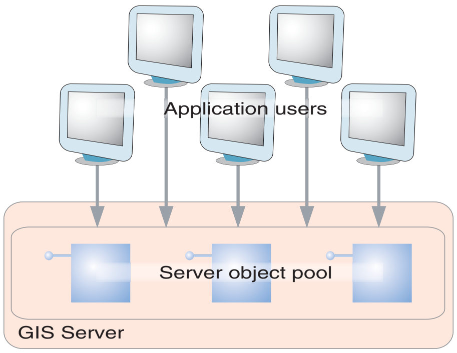
ArcGIS Server allows you to pool instances of server objects so they can be shared between multiple application sessions at the per request level. This allows you to support more users with fewer resources allocated to a particular server object. When a server object is configured to be pooled, the minimum and maximum number of instances must be specified as properties of the server object configuration. When the server object is started, the GIS server will precreate and initialize the minimum number of server objects. When an application asks the server object manager for an instance of that server object, it will get a reference to one of the preloaded server objects in the pool.

The advantages of pooling server objects include:

- Separate the potentially expensive initialization of a server object from the actual work that the object performs for each client.
- Share the cost of expensive initialization and acquisition of resources, such as database connections and so on, across all clients.
- Precreate objects at server object manager startup, before any client requests come in.
- Administratively configure pooling to take best advantage of available hardware resources.

Applications that use pooled server objects keep their reference on that object for the duration of their request (for example, draw map, identify feature, geocode address), then return the object to the server. When the object is returned to the server, it's available for use by another user's request. Users of such an application may be working with a number of different instances of a server object in the pool as they interact with the application, all of which is transparent to the users.

If there are more simultaneous requests on a pooled server object than the minimum, new instances of the server object will be created until the maximum number of instances is reached, at which point the user is queued until objects in the pool become free.



Pooled server objects can support more users because application sessions share a collection of objects in the pool.

Pooling server objects allows the GIS server to support more users. Because applications can share a pool of server objects, the number of concurrent users on the system is greater than the number of server objects that the GIS server can effectively support at one time.

Non-pooled and pooled server objects support different types of server applications. Applications are expected to make stateless use of pooled server objects, meaning they do not make changes to the server object when they are using it, and they are released back to the pool in a timely manner when the request has been processed.

For example, a stateless mapping application that wants to draw a certain extent of a MapServer's map will get a reference to an instance of a MapServer server object from the pool, execute a method on the MapServer to draw the map, then release it back to the pool. The next time the application needs to draw the map, this is repeated. Each draw of the map may use a different instance of the pooled server object; therefore, each pooled object must be the same (have the same set of layers, the same renderer for each layer, and so on). If the state of one of the pooled objects is changed (for example, a new layer is added, a layer's renderer is changed), then as a user pans and zooms around the map, inconsistent results will be seen.

Non-pooled server objects are created for each application session that uses it. Since server objects can take time to initialize, the application that makes use of a non-pooled object typically holds a reference to the server object for the duration of the application's session. Since the server object is destroyed when it's returned to the server, the application is free to change any aspect of the server object's state.

Stateful versus stateless use of server objects, as well as the use of application session state in ArcGIS Server applications, will be discussed in greater detail in Chapter 4, 'Developing ArcGIS Server applications'.

CREATION TIME, WAIT TIME, AND USAGE TIME

When server objects are created in the GIS server, either as a result of the server starting or in response to a request for a server by a client, the time it takes to initialize the server object is referred to as its creation time. The GIS server maintains a maximum creation time-out that dictates the amount of time a server object has to start before the GIS server will assume its startup is hanging and cancel the creation of the server object.

When the maximum number of instances of a pooled or non-pooled server object is in use, a client requesting a server object will be queued until another client releases one of the server objects. The amount of time it takes between a client requesting a server object and getting a server object is called the wait time. A server object can be configured to have a maximum wait time. If a client's wait time exceeds the maximum wait time for a server object, then the request will time out.

Once a client gets a reference to a server object, it can hold onto that server object for as long as it wants before releasing it. The amount of time between when a client gets a reference to a server object and when it releases it is called the usage time. To ensure that clients don't hold references to server objects for too long (that is, they don't correctly release server objects), each server object can also be configured with a maximum usage time. If a client holds on to a server object for longer than the maximum usage time, then the server object is automatically released and the client will lose its reference to the server object.

Maximum usage time also protects server objects from being used to do larger volumes of work than the administrator intended. For example, a server object that is used by an application to perform geodatabase checkouts may have a maximum usage time of 10 minutes. In contrast, a server object that is used by applications that only draw maps may have a maximum usage time of one minute.

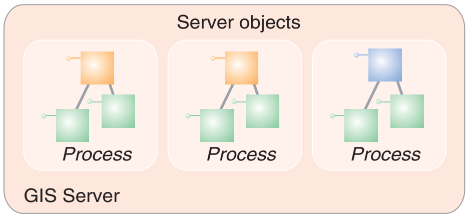
The GIS server maintains statistics both in memory and in its log files about wait time, usage time, and other events that occur within the server. The server administrator can use these statistics to determine if, for example, the wait time for a server object is high, which may indicate a need to increase the maximum number of instances for that server object.

For more information about how to view these statistics, see Chapter 3, 'Administering an ArcGIS Server', and Appendix B, 'Configuration and log files'.

Server object isolation

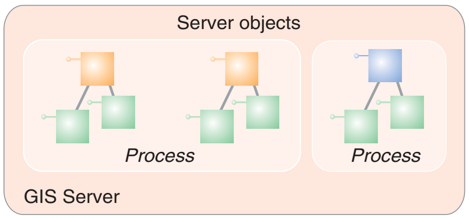
Server objects run within processes on the container machines. Server objects can be configured such that they run in a dedicated process on the server, or they can be configured to run in processes they share with other server objects. How they share processes is referred to as their isolation level.

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.



Server objects with high isolation run in dedicated processes on the GIS server.

Up to four server objects can share the same process. When more than four server objects of a particular type (for example, 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.



Server objects with low isolation can share processes with other server objects of the same type.

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. If a server object is sharing its process with other server objects, however, and the process is shut down or crashes, all the server objects in that process will be destroyed.

Server object recycling

Recycling allows server objects that have become unusable to be destroyed and replaced with fresh server objects; recycling also reclaims resources taken up by stale server objects. This process allows you to keep the pool of server objects fresh and cycle out stale or unusable server objects.

Pooled server objects are typically shared between multiple applications and users of those applications. Through reuse, a number of things can happen to a server object to make it unavailable for use by applications. For example, an application may incorrectly modify a server object's state, or an application may incorrectly hold a reference to a server object, making it unavailable to other applications or sessions. In some cases, server objects may become corrupted and unusable.

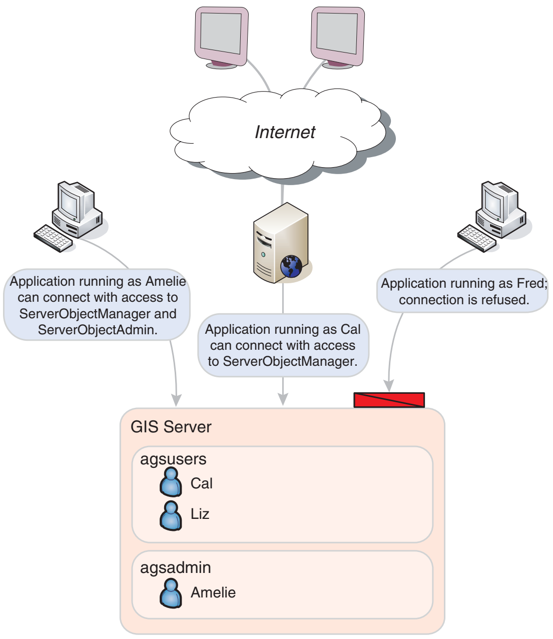
Non-pooled server objects whose isolation level is low can also be recycled. This recycling will shut down and restart the processes in which non-pooled objects are started and run.

In each case, recycling occurs as a background process on the server. The time between recycling events is called the recycling interval. A server object's recycling interval can be configured by the administrator. During recycling, instances of server objects in use by clients are not recycled until released, so recycling occurs without interrupting the user of a server object.

The ArcGIS Server is a secure server and grants connections only to those users who are authorized to connect by the GIS server administrator. There are two levels at which security can be configured for an ArcGIS Server: at the GIS server level itself and at the level of a Web application or Web service that runs in the Web server. Each of these will be discussed separately.

GIS SERVER SECURITY

ArcGIS Server security is based on authenticating operating system user accounts. Connections will be granted to the server for those users who are members of the ArcGIS Server users group (agsusers). The agsusers group is an operating system group, created by the ArcGIS Server install on the SOM machine and all container machines. When a user runs an application that connects to the server, that user's login is authenticated against the users group. If the user is a member of that group, then access is granted to the server; if not, the connection is rejected.


Internet
Application running as Amelie can connect with access to ServerObjectManager and ServerObjectAdmin.
Application running as Cal can connect with access to ServerObjectManager.
Application running as Fred; connection is refused.
GIS Server
agsusers
Cal
Liz
agsadmin
Amelie

When applications make connections to the GIS server, they are authenticated against the agsusers and agsadmin users groups on the GIS server.

Once connected to the server, the user or application can make use of the server objects running in the server and create new objects in the server for the application's use.

Members of the users group have consumer-level privileges on the GIS server. Consumers may not perform administrative tasks, such as add, remove, or modify preconfigured server objects, or modify properties of the server itself, such as adding and removing machines and so on.

Users may also connect to the server if they are a member of the ArcGIS Server administrators group (agsadmin). These users are granted administrator privileges on the GIS server. Once connected to the server as an administrator, the user or application can administer aspects of the server, such as:

• Add or remove container machines.

• Add, remove, or modify server directories.

• Add, delete, or modify server objects.

• Start, stop, or pause server objects.

• View statistical information.

The users (agsusers) and administrators (agsadmin) groups are created as local user groups by the ArcGIS Server install on both the SOM machine and container machines. When adding a user account to one of these groups, it's important to make sure that the account is added to the group on each machine (the SOM machine and all container machines). One strategy for doing this is to create a domain user group for server users and another for server administrators, then add those domain groups to the users and administrators groups, respectively, on all machines. Then your task of adding and removing users privileges to the ArcGIS Server can be managed by adding and removing them from those domain user groups.

If your login user is not a member of the users or administrators groups, you can use the "runas" command to run ArcCatalog as another user who is in the appropriate group.

A desktop client application to ArcGIS Server will run as the user account that started the application. For example, if you are logged into a desktop computer as the domain user ANDY on the domain AVWORLD, and you are running an application such as ArcCatalog, the identity of the application is AVWORLD\ANDY. When you connect to an ArcGIS Server in that ArcCatalog session, you are connecting as AVWORLD\ANDY. As long as AVWORLD\ANDY is a member of the users group (agsusers) on the SOM, you will be able to connect. If AVWORLD\ANDY is a member of the administrators group (agsadmin), you will have administrator privileges on the server through that connection.

When you connect to ArcGIS Server with ArcCatalog, you will have more commands when connected as an administrator than you would as a consumer—that is, the commands necessary for administering the GIS server. For more details on how to use ArcCatalog to administer ArcGIS Server, see Chapter 3, 'Administering an ArcGIS Server'.

IMPERSONATION

Web applications running in the Web server must connect to the ArcGIS Server as a valid GIS server user (that is, a member of the users group). The Web application must use impersonation to connect to the server as a user account in the users group. For more information on impersonation in Java and .NET, see Chapters 5, 'Developing Web applications with .NET', and 6, 'Developing Web applications with Java'.

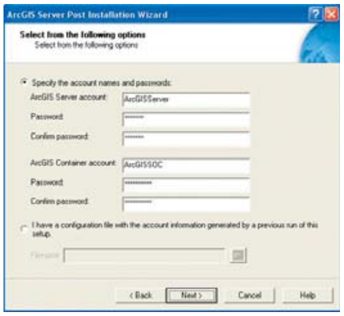
At the application level, Web applications and Web services define their own security model based on ASP.NET, the .NET implementation of Active Server Pages, and Java 2 Platform Enterprise Edition (J2EE). Based on this standard security infrastructure, you can build anonymous applications and Web services that are open to all who know the URL. You can also build secure applications with their own users, authentication, and authorization independent of the GIS server. Using these security infrastructures, you can limit the accessibility of your Web application or Web service and deny unauthorized users from having access to your GIS server through the Internet.

These technologies are beyond the scope of this book. For more information on securing ASP.NET and J2EE Web applications, consult the literature on those technologies.

IDENTITY

The GIS server itself runs as two distinct operating system accounts: the server account and the container account. These accounts can have any name and can be assigned to any account already existing in your organization. However, it is important to understand why these accounts are necessary before deciding whether existing accounts in your domain should be used or if you should let the ArcGIS Server installation create these accounts for you.

The server account is the account that runs the Server Object Manager Windows service or UNIX daemon and process. This process manages the container processes on the container machines as well as the GIS server's configuration information and log files. So, the server account has privileges to write to the locations


ArcGIS Server Post Installation Wizard

Select from the following options
Select from the following options

(●) Specify the account names and passwords:

ArcGIS Server account:	ArcGISServer
Password:	********
Confirm password:	********
ArcGIS Container account:	ArcGISSOC
Password:	**********
Confirm password:	**********

( ) I have a configuration file with the account information generated by a previous run of this setup.

Filename:

The GIS server postinstallation application allows you to specify the server and container accounts.

where the server configuration information and log files are stored. It also has privileges to start container processes on the container machines.

The container process actually hosts the server objects and does the work. Container processes are started by the server object manager but run as the container account. Therefore, the container account must have read access to any GIS resources (maps, locators, data) that preconfigured and application-specific server objects require to do their work. In addition, the container account must have write access to the server directories of the GIS server so that server objects running in container processes can write their output. These aspects of the container account are important for administering your site, especially when considering privileges on shared network drives and so on.

One important aspect of the container account is that, since the container processes runs as that account, a user who connects to the GIS server can do anything that the container account can do. Because developers are free to create their own objects on the server, they have access to a wide range of functionality, including the ability to read data that the container account has read privileges on. More important, developers can edit, delete, and otherwise affect files that the container account has write privileges to.

It can be dangerous to use a domain account with many privileges as the container account for your GIS server. The container account should only have enough privileges to access necessary data and perform the task of running server objects. The ArcGIS Server installation can create the server container account with the following minimum privileges on each container machine:

- Ability to launch container processes
- Write access to the system temp directory

It is up to the GIS server administrator to grant this account access to any necessary data and write privileges to the server's output directories.