ArcGIS® 9

Managing ArcSDE® Application Servers

Contents

Introducing the ArcSDE application server 1

IN THIS CHAPTER

- **The ArcSDE application server**
- **Properties of an ArcSDE application server**
- **Tips on learning about the ArcSDE application server**

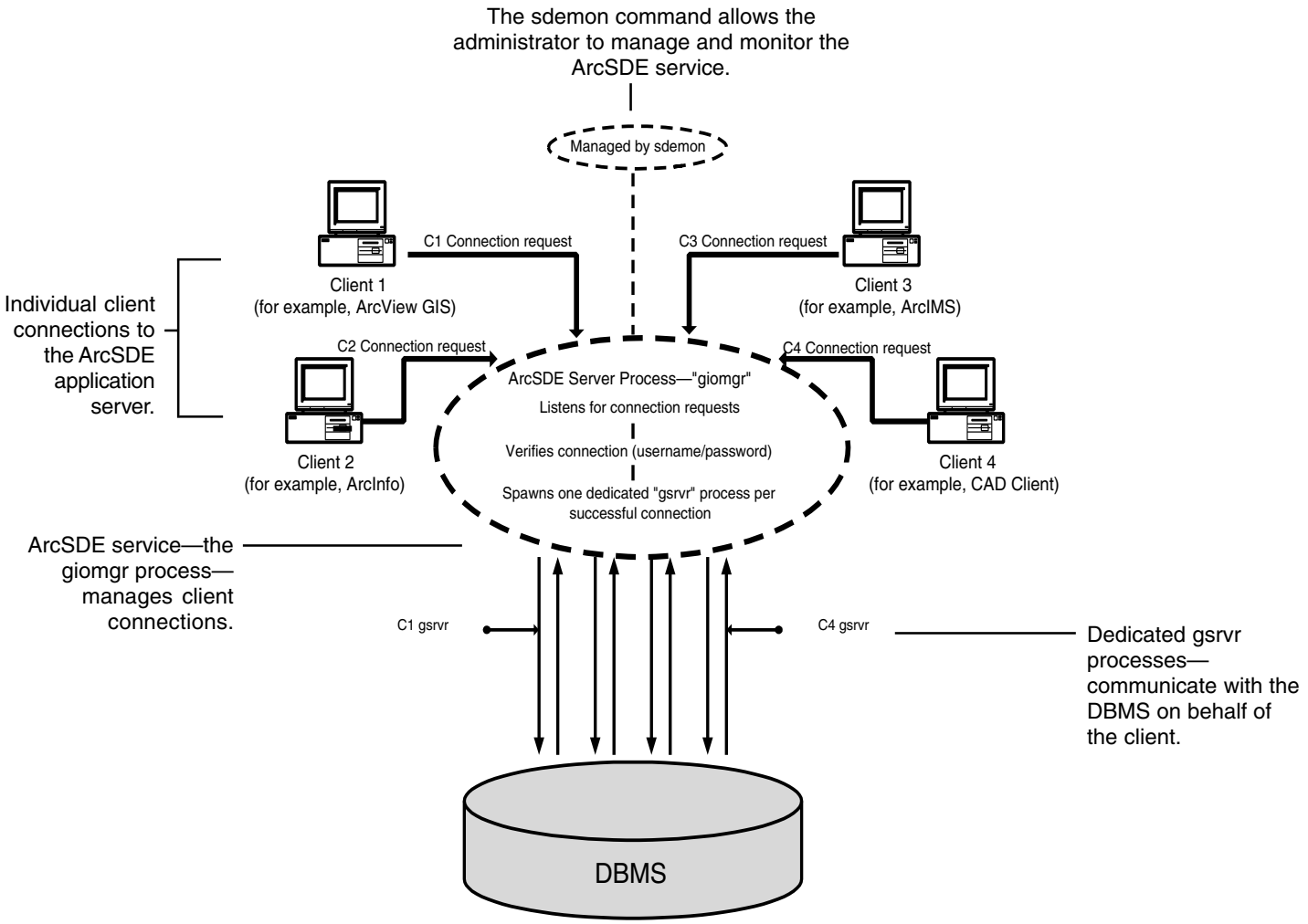
This chapter introduces the key components of the ESRI® ArcSDE® application server and provides an overview of the underlying configuration. Subsequent chapters discuss the creation, configuration, and management of the ArcSDE application server. The appendices contain descriptions of the ArcSDE home directory, the data dictionary, table definitions, listings of the ArcSDE commands, and the ArcSDE initialization parameters.

The ArcSDE application server

An ArcSDE application server conveys spatial data between geographic information system (GIS) applications and a database. The database may be any one of the supported database management systems (DBMSs) such as Oracle®, SQL Server™, Informix®, or DB2®. The database could also be a registered collection of ArcGIS® coverages or shapefiles. The applications that can connect to and access spatial data from an ArcSDE application server include ArcGIS Desktop, ArcIMS®, ArcView® 3, ArcInfo™ Workstation, MapObjects®, and ArcSDE CAD Client, as well as custom-built applications created by either you or an ESRI business partner.

Two-tiered versus three-tiered architecture

ArcSDE provides two alternative methods of connecting to the database instance—direct connection to the database instance in a two-tiered architecture or connecting through the ArcSDE application server in a three-tiered architecture. If it is your intention to adopt the two-tiered approach, then you should consult the Appendix titled 'Making a Direct Connection' in the *ArcSDE Configuration and Tuning Guide for <DBMS>* PDF file, located in the documentation folder on the ArcSDE CD. This guide is primarily concerned with the administration of the ArcSDE application server. However, information concerning such things as the management of server configuration parameters (Chapter 3 and Appendix D) and troubleshooting (Chapter 6) also apply to direct connections.


The sdemon command allows the administrator to manage and monitor the ArcSDE service.
Managed by sdemon
C1 Connection request
C3 Connection request
Client 1
(for example, ArcView GIS)
Client 3
(for example, ArcIMS)
Individual client connections to the ArcSDE application server.
C2 Connection request
C4 Connection request
ArcSDE Server Process—"giomgr"
Listens for connection requests
Verifies connection (username/password)
Spawns one dedicated "gsrvr" process per successful connection
Client 2
(for example, ArcInfo)
Client 4
(for example, CAD Client)
ArcSDE service—the giomgr process—manages client connections.
C1 gsrvr
C4 gsrvr
Dedicated gsrvr processes—communicate with the DBMS on behalf of the client.
DBMS

Properties of an ArcSDE application server

The ArcSDE application server, a client/server configuration, has a number of defining properties; these are introduced in the following sections.

The home directory

Each ArcSDE application server has its own home directory, which is defined by the system variable, SDEHOME. The home directory contains the ArcSDE binary executable files, dynamic shared libraries, configuration files, and internationalization code pages.

The ArcSDE application server monitor—the giomgr process

Each ArcSDE application server has one giomgr process. This process listens for user application connection requests, spawns gsrvr processes, and cleans up disconnected user processes.

The giomgr will not start if a valid server license has not been installed. Contact ESRI customer support if you have not already obtained a valid license.

The gsrvr process

The giomgr process spawns a gsrvr process for each application connected to the ArcSDE application server. The gsrvr process is dedicated to a single-user application connection. It communicates with the database on behalf of the connected application. The gsrvr process responds to the application's query and edit requests to the database.

The Transmission Control Protocol/Internet Protocol service name and port number

The ArcSDE application server, through the giomgr process, listens for application connection requests on a dedicated TCP/IP service name and port number. The gsrvr process, in turn, communicates with the application on the same TCP/IP service and port number. All communication between the applications and the ArcSDE application server take place using the TCP/IP service name and port number.

The configuration parameters

The ArcSDE application server can be configured to optimize the data flow between the connected applications and the database.

The configuration parameters are stored in the SDE.SERVER_CONFIG metadata table, making them accessible to multiple application servers and direct connections. The parameters are managed by editing the configuration files and loading the values into the SDE.SERVER_CONFIG table. The default configuration file, giomgr.defs, is located in the etc folder of SDEHOME.

Tips on learning about the ArcSDE application server

About this book

This book is designed to help you create and maintain an ArcSDE application server. It discusses the ArcSDE application server—how to configure the application server, as well as how to monitor its usage. In addition to showing how to configure and manage your ArcSDE application, it also shows you how to monitor your service and cope with problems that may arise.

The focus of this book is not the creation or administration of a DBMS. This is a separate topic; for information on that topic, see the *ArcSDE Configuration and Tuning Guide for <DBMS>* PDF file, ArcSDE_Config_Gd_<DBMS>.pdf, located in the documentation folder on the ArcSDE CD.

Contacting ESRI

If you need to contact ESRI for technical support, refer to 'ContactingTechnical Support' in the 'Getting more help' section of the ArcGIS® DesktopHelp system.

You can also visit ESRI on the Web at www.esri.com for more information on ArcSDE and ArcInfo.

ESRI education solutions

ESRI provides educational opportunities related to geographic information science, GIS applications, and technology. You can choose among instructor-led courses, Web-based courses, and self-study workbooks to find education solutions that fit your learning style. For more information, go to www.esri.com/education.

Creating ArcSDE application servers 2

IN THIS CHAPTER

- **The ArcSDE home directory**
- **The DBMS database**
- **The ArcSDE DBMS administration account**
- **The ArcSDE service on UNIX and Windows**
- **Windows installs**
- **Accessing an ArcSDE application server through a firewall**

Before you can create an ArcSDE application server, you must install the ArcSDE software. For ArcSDE installation instructions, refer to the *ArcSDE Installation Guide* for your DBMS.

When you install ArcSDE, the ArcSDE software is copied into a directory called the home directory, or SDEHOME.

The DBMS must be prepared by running the sdesetup utility for your DBMS. This utility creates the required ArcSDE and Geodatabase metadata tables under the SDE user's schema.

Before you start an ArcSDE application server, check the following:

- The ArcSDE software has been installed correctly, and the default home directory (SDEHOME) has been established. On Windows®, any installation errors will be reported to the <SDEHOME>\SDE90ORA.LOG file.
- A supported DBMS database has been configured for ArcSDE.
- An ArcSDE user account exists in the DBMS, with enough free space available to store the ArcSDE repository.
- A unique TCP/IP service name and port number is available.

The ArcSDE home directory

The ArcSDE home directory, also known by the system variable that stores its location (either %SDEHOME% on Windows or $SDEHOME on UNIX®), contains all of the executable files and dynamic libraries required to run an ArcSDE application server. This directory structure is automatically set up during the installation process. Each independent ArcSDE application server requires its own home directory.

The ArcSDE application server's configuration files must then be updated to reflect the unique properties of the new application server. Many ArcSDE application servers can run on a single computer, with each application server requiring its own home directory, TCP/IP service name, and port number. The hardware platform must have sufficient resources available to support the processes of the ArcSDE software and the DBMS.

Refer to the *ArcSDE Installation Guide* for more information about creating additional ArcSDE application servers.

The DBMS database

ArcSDE stores data in a DBMS database. This database must exist before the ArcSDE application server can be created. Consult your DBMS documentation for guidance on creating a DBMS database. You should also review the basic advice provided in the *ArcSDE Configuration and Tuning Guide for <DBMS>,* which is located in the documents directory of the CD–ROM installation media and in the ArcSDE home directory (SDEHOME).

Microsoft® SQL Server, IBM® Informix, and IBM DB2 supported servers may contain one or more databases, although be aware that DBMS vendors vary in their definition of a database. An ArcSDE application server can only connect to one database server and to one database.

Configuration parameters that control the manner in which an ArcSDE application server connects to a database may be stored in the dbinit.sde file. This file is located in the %SDEHOME%\etc directory on Windows and in the $SDEHOME/etc directory on UNIX. This avoids the need to rely on environment variables set when the user logs in. The dbinit.sde file is discussed in detail in Chapter 3, ‘Configuring ArcSDE application servers’.

For optimum performance, the database and the ArcSDE application server should coexist on the same host. However, it is possible for ArcSDE to connect to a database hosted on a remote machine. For more information on establishing a remote connection to an ArcSDE service, refer to the *ArcSDE Configuration and Tuning Guide for <DBMS>*.

The ArcSDE DBMS administration account

An ArcSDE application server requires the creation of an ArcSDE DBMS administration account. On Windows, the ArcSDE installation process gave you the opportunity to create an ArcSDE DBMS administration account and the necessary DBMS storage space to store the data dictionary tables owned by the ArcSDE administrator account. If this step was bypassed during the installation, it must be completed before the ArcSDE application server can be started up.

The procedure for creating the ArcSDE DBMS administration account differs for each ArcSDE product. For information on how to create the ArcSDE DBMS administration account, refer to *ArcSDE Installation Guide*.

Under the ArcSDE DBMS administration account, ArcSDE stores its data dictionary tables. The tables are created and initially populated by the ArcSDE sdesetup command line utility. There is a separate utility for each DBMS.

The ArcSDE service on UNIX and Windows

Each ArcSDE application server listens for user connections on a dedicated TCP/IP service name and port number through the giomgr process.

On UNIX systems, edit the $SDEHOME/etc/services.sde file and the /etc/services file to include the service name.

On Windows systems, the ArcSDE application server is created during the ArcSDE software installation as a Windows service. The service name is stored in the registry under HKEY_LOCAL_MACHINE\SOFTWARE\ESRI\ArcInfo\ArcSDE\ArcSDE for <product>\<service name>.

Tip

Port number

Although it is not used at this stage, the port number in the SDEHOME/etc/services.sde file is included to remind the ArcSDE administrator that the service name is assigned to the 5151/TCP port in the operating system services file.

Registering an ArcSDE service on UNIX

1. Edit the $SDEHOME/etc/services.sde file.
2. Add the same service name and port number to the operating system /etc/services file.

```
#
# ESRI ArcSDE service name and port number
#
esri_sde          5151/tcp ———————————— 1
```

$SDEHOME/etc/services.sde file

```
.
.
.
nfsd      2049/udp   # NFS server daemon (clts)
nfsd      2049/tcp   # NFS server daemon (cots)
lockd     4045/udp   # NFS lock daemon/manager
lockd     4045/tcp
esri_sde  5151/tcp   # ArcSDE 9 service ——— 2
dtspc     6112/tcp   # CDE subprocess control
fs        7100/tcp   # Font server
.
.
```

Operating system /etc/services file

Windows installs

The installation program automatically enters the service name and port number for the default service name in both the registry and the Windows services file, C:\winnt\system32\drivers\etc\services.

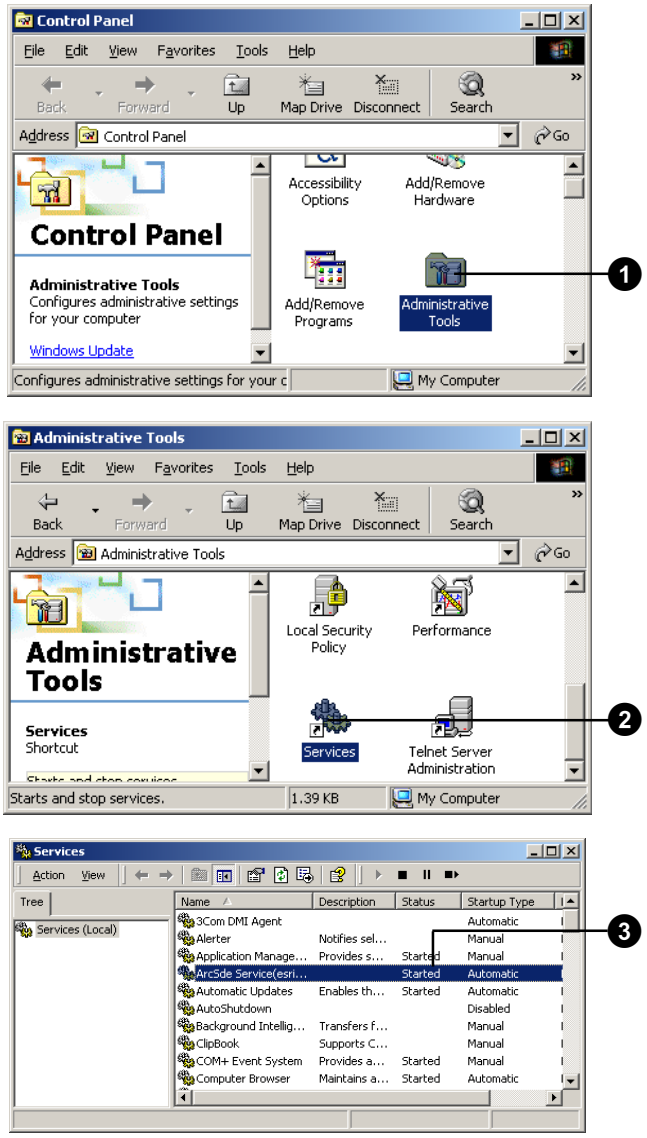
The installation program also creates the Windows service, setting it to automatic—that is, it will automatically start up when the server host is rebooted.

Like all Windows services, the ArcSDE application server is started and stopped from the Windows services menu.

Although it is more convenient to start the ArcSDE application server from the Windows service menu, it is sometimes necessary to use the sdemon command because error messages are printed directly to the MS–DOS command window instead of the event log. Using sdemon to start the service can be useful when trying to diagnose a startup problem. Unlike the Windows service, the sdemon command reads the service name from the %SDEHOME%\etc\services.sde file. Edit this file to ensure that it contains the correct information before using sdemon.

Verifying an ArcSDE service on Windows

1. Open the Control Panel and double-click Administrative Tools.
2. From the Administrative Tools menu, click Services.
3. From Services, make sure the ArcSDE service has been started.



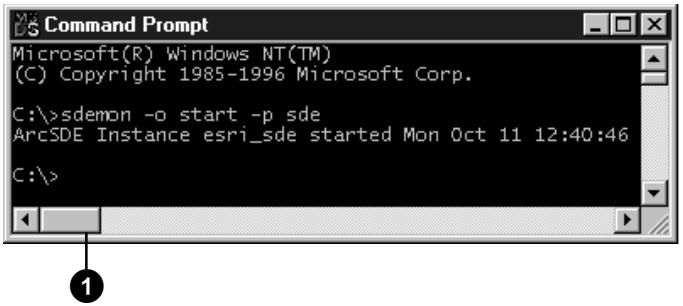
Tip

Adding more ArcSDE services

If additional ArcSDE application servers are required, use the sdeservice command to define the new Windows service. For a complete description of the sdeservice command, see 'Appendix C: ArcSDE application server command references'.

Monitoring the ArcSDE service

1. Use the sdemon command at the Windows MS–DOS command prompt to trap ArcSDE application server error messages.



Accessing an ArcSDE application server through a firewall

To provide access to an ArcSDE application server inside a system security *firewall,* the host computer on which the ArcSDE application server is installed should be listed in your domain name server (DNS) database. The DNS must be registered with your Internet service provider (ISP) or directly with Network Solutions (formerly called InterNIC), the organization that registers Internet domain names.

Your DNS resolves the IP address of your computer to the name, or universal resource locator, you wish to make accessible to the Internet. In most cases, you will have more machines within your local network than you will have Internet IP addresses for. In this case, you would maintain your own set of internal IP addresses known only to your local area network (LAN). Your firewall, or proxy server software, will translate your internal IP addresses to Internet IP addresses when you access computers outside your LAN.

Since ArcSDE application servers listen for connections on a TCP/IP port number that corresponds to your service name, you must also add the TCP/IP port number to the computer's host name when connecting to it.

For ArcSDE applications built with the ArcSDE C application programming interface (API), the host must always be specified using host name.

For ArcSDE Java applications you can specify an ArcSDE host name in two ways. You can either use the DNS name, if it is available, or you can connect to it directly using its Internet IP address.

For example, ESRI's domain name—esri.com—has been registered with Network Solutions, and we identified our DNS as IP address 198.102.62.1. Our DNS has the IP address for the ArcSDE host Toshi in its DNS database. The internal IP address for Toshi is 46.1.2.324, which is translated to the IP address 198.102.62.5 when Toshi sends and receives information through the firewall. The ArcSDE application server running on Toshi is listening for connections on the service name esri_sde3, which corresponds to TCP/IP port number 5165. So, if you wish to connect to that particular ArcSDE application server, you must specify either the host name "toshi.esri.com:5165" or identify Toshi by its IP address "198.102.62.5:5165". In both cases you must also include the service port number, 5165.

The giomgr process bequeaths the port number to the gsrvr process following a successful connection. Therefore, all communication to the ArcSDE application server occurs on the same TCP/IP port number.

If you cannot connect to an ArcSDE application server through a firewall, test the accessibility of the remote ArcSDE host with your Internet browser by specifying either the server name and TCP/IP port number or the IP address and TCP/IP port number as the URL.

The correct syntax is:

```
<server name>:<port number>
<IP address>:<port number>
```

Configuring ArcSDE application servers 3

IN THIS CHAPTER

- **The services.sde file**
- **Operating system services files**
- **The service name on Windows**
- **ArcSDE DBMS environment variables**
- **ArcSDE system environment variables**
- **The dbinit.sde file format**
- **Displaying ArcSDE system environment variables**
- **Adjusting ArcSDE application server initialization parameters**
- **Displaying the ArcSDE initialization parameters**

ArcSDE conveys spatial data between a DBMS and applications. Configuring an ArcSDE application server focuses on maximizing data transfer between the server and the client with limited shared resources. The requirements of the application, the number of users, and the amount of data requested influence the configuration of the ArcSDE service.

The ArcSDE application server can be configured by modifying the parameters in the configuration metadata tables stored in the ArcSDE user's schema or the configuration files services.sde and dbinit.sde, located in the SDEHOME/etc directory.

The services.sde file

Each ArcSDE application server communicates with the database management system and the applications through a TCP/IP port. TCP/IP ports are defined by name in the operating system's services file.

The SDEHOME\etc\services.sde file contains the service name and the unique TCP/IP port number on which the ArcSDE application server accepts connection requests. This port number is also assigned to each user or gsrvr process that the ArcSDE application server initiates. The port number listed in the services.sde file does not designate operating system port use. It is included in the services.sde by convention as a reminder of the port number assigned to the service name in the operating system's services file.

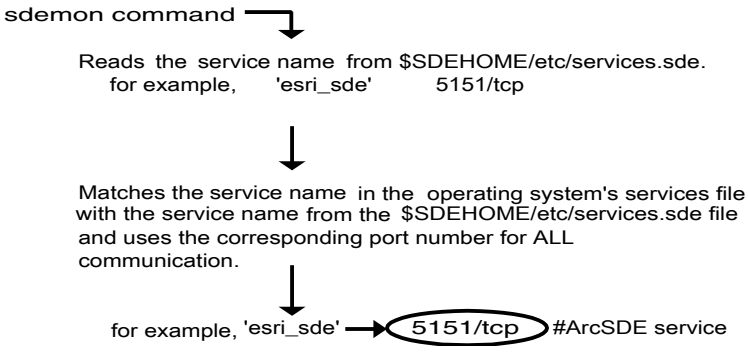
ESRI has registered the default esri_sde service name and 5151 TCP/IP port number with the Information Sciences Institute, Internet Assigned Numbers Authority.

The default services.sde file created during the installation process will contain the following:

```
#
# ESRI ArcSDE Remote Protocol
#
esri_sde              5151/tcp
```

To change the default service name, simply edit the file and restart the service.

On UNIX systems the sdeservice.sde file is always used. On Windows systems, however, the services.sde file is used only when the service is started from the MS–DOS prompt using the sdemon command. If the service was started from the Windows service panel, ArcSDE uses the service name stored in the registry under HKEY_LOCAL_MACHINE\HARDWARE\ESRI\ArcInfo\ArcSDE\ArcSDE for <*dbms*>\esri_sde.



Mapping the SDEHOME/etc/services.sde entries to the system services file entries

When a match is found, ArcSDE starts the giomgr process. It listens for user connection requests on the TCP/IP port number assigned to the service name.

When the ArcSDE application server is started with the sdemon command, the service searches the system services file for a service name that matches the service name in the services.sde file.

If a match is not found, ArcSDE returns the following error on UNIX systems:

```
$ sdemon -o start
Please enter the SDE DBA password: *****
This instance's service name esri_sde not found
in system services file.
```

To diagnose startup problems on a Windows platform, examine the event log. For more information, see Chapter 6, 'Troubleshooting the ArcSDE application server'.

Multiple ArcSDE application servers running on the same host require unique service names and port numbers. If you intend to create another ArcSDE application server on the same host, you must edit one of the services.sde files and amend the service name and port number accordingly. You must also update the system services file to include the new ArcSDE application server.

The services.sde file in each SDEHOME\etc directory should contain only one service name. If you add more than one service name to this file, ArcSDE will use the first one it encounters and ignores the rest.

The operating system services file contains many service names. At least one of them must match your service name, found in the services.sde file. Enter the service name into the operating system services file on the server and client machines.

Here is an example from an operating system services file that contains two ArcSDE services:

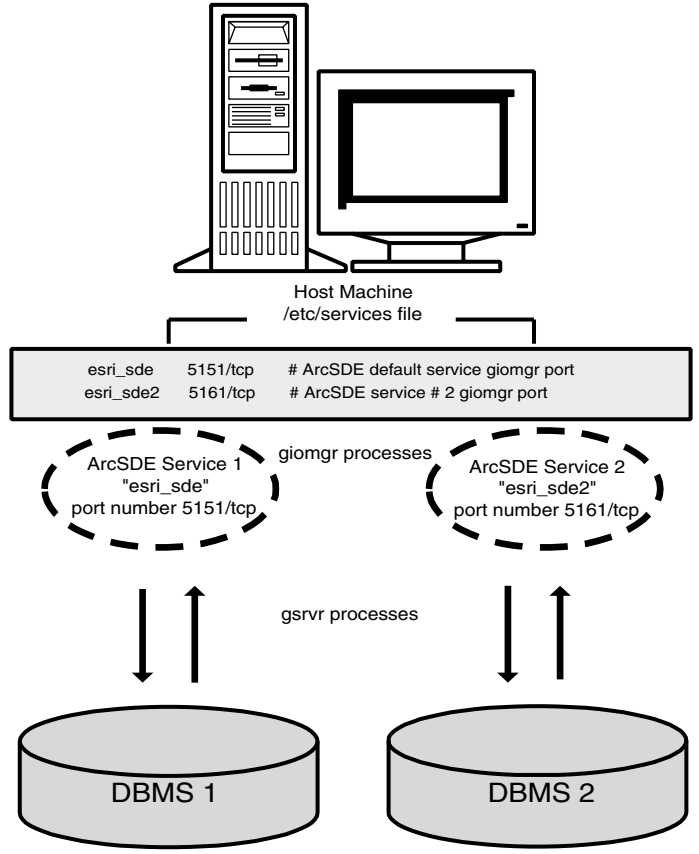
```
esri_sde 5151/tcp       # ArcSDE default service

esri_sde2 5161/tcp      # another ArcSDE service
```

This defines two service names: esri_sde and esri_sde2.

On UNIX systems, you can use the Network Information Service (NIS) services file if you are running NIS to avoid unnecessary duplication of effort when updating the client and server local system services files.

The dbinit.sde file of the new service must also be edited so it contains the connection information of the second DBMS. For more information on setting the DBMS connection variables in the dbinit.sde file, see 'ArcSDE DBMS environment variables' in this chapter.



One host computer supporting two ArcSDE application servers

Operating system services files

The services file on a Windows platform is located under the winnt\system32\drivers\etc directory. Use an editor, such as Notepad, that does not embed format characters to edit the Windows services file.

The services file for UNIX systems is located at /etc/services.

Some UNIX systems direct applications to search the NIS services file rather than the local services file. If you wish to have the operating system search the local services file instead, you must force it to do so.

Forcing a search of local services on HP–UX®

1. Copy the nsswitch.conf file from the /usr/newconfig/etc directory to the /etc directory.
2. Edit the file and change the line 'services: nis files' to 'services: files nis'.

```
# /etc/nsswitch.conf:

#This file uses NIS (YP) in conjunction with
# files.

# "hosts:" and "services:" in this file are used
# only if the /etc/netconfig file has a "-" for
# nametoaddr_libs of "inet" transports.

# the following two lines obviate the "+" entry
# in /etc/passwd and /etc/group.

passwd:       files nis
group:        files nis
# consult /etc "files" only if nis is down.
hosts:        files nis dns
networks:     nis [NOTFOUND=return] files
protocols:    nis [NOTFOUND=return] files
rpc:          nis [NOTFOUND=return] files
ethers:       nis [NOTFOUND=return] files
netmasks:     nis [NOTFOUND=return] files
bootparams:   nis [NOTFOUND=return] files
publickey:    nis [NOTFOUND=return] files
netgroup:     nis
automount:    files nis
aliases:      files nis
# for efficient getservbyname() avoid nis
services:     files nis                    2
sendmailvars: files
```

Forcing a search of the local services on IBM AIX®

1. In the /etc directory, create the file netsvc.conf.
2. Add the line "services=local.nis".

The service name on Windows

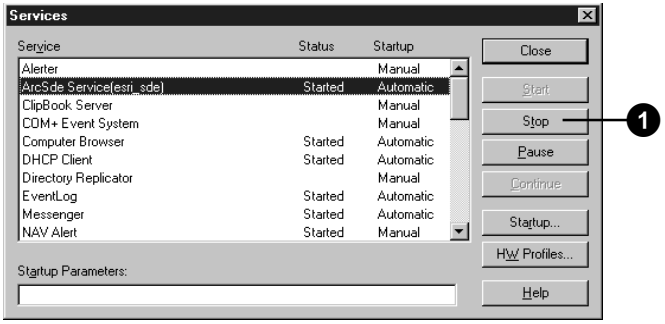
On Windows systems, the services.sde file is only accessed when the ArcSDE application server is started with the sdemon command. When the service is started from the services menu, the service name is read from the system registry.

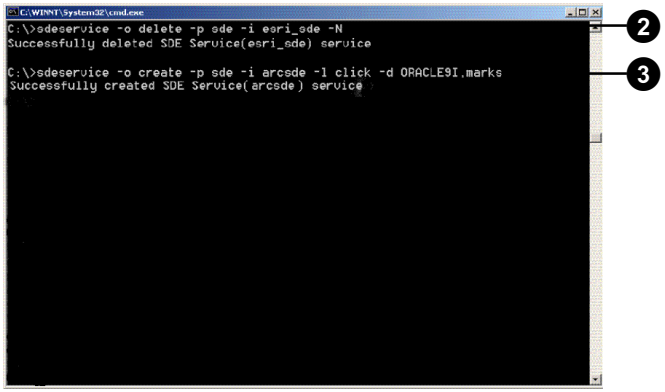
To change the service name in the Windows registry, use the sdeservice administration command.

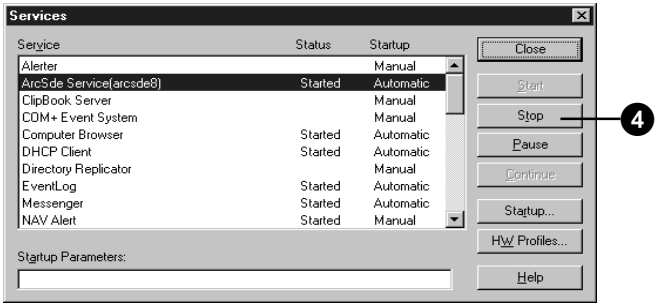
See Also

For more information about the sdeservice command, see 'Appendix C: ArcSDE application server command references'.

1. Shut down the ArcSDE application server from the Windows NT Services dialog box by clicking Stop.
2. From a DOS command prompt, remove the old service name using the sdeservice command with the delete option.
3. From a DOS command prompt, create the new service name with the sdeservice -o create option.
4. Restart the ArcSDE application server from the Services dialog box. Notice the service name has changed.







The dbinit.sde file and the application server

The ArcSDE application server reads system environment variables from the SDEHOME\etc\dbinit.sde file for such things as establishing the DBMS server to connect to and limiting the messages written to the sde error log (sde_<service_name>.log) file. System environment variables set within the dbinit.sde file override those set within the system environment of the user that starts the application server. For variables absent from both the dbinit.sde file and the user environment, default ArcSDE settings are used.

The system environment variables that control the ArcSDE application server's connection to a DBMS depend on each individual DBMS.

For more details on setting environment variables, refer to the *ArcSDE Installation Guide for < DBMS>*.

By default, SDEHOME determines on which ArcSDE application server the sdemon command will operate. This variable can be overridden by the -H option of the sdemon command.

```
setenv SDEHOME d:\arcexe\arcsde
```

SDESERVER determines the host of the ArcSDE application server for the connecting client application. This variable can be overridden by specifying the host in the application. If the host is not specified during connection and the SDESERVER variable is not set, the client application attempts to connect to an ArcSDE application server running on the local host.

```
setenv SDESERVER bruno
```

SDEINSTANCE is set in the environment of the client application and determines the ArcSDE service name to connect to. Specifying the service in the application overrides this variable. If this variable is not set and the service name is not specified in the application, the service name defaults to esri_sde.

```
setenv SDEINSTANCE esri_sde
```

SDEDATABASE can be entered for the DBMS and has multiple database connection possibilities within a single application server. Specifying the database through the application overrides this variable. If the variable is not set and the database is not specified upon connection, the ArcSDE client connects to the default database.

```
setenv SDEDATABASE city_eng
```

SDEUSER specifies the username through which the ArcSDE client application will connect. Specifying the username in the application overrides this variable. If this variable is not set and the username is not specified in the application, an error is returned. A username must be specified.

```
setenv SDEUSER bob
```

SDEPASSWORD specifies the password for the username entered by the ArcSDE client application. Specifying the password on the application connection tool overrides this variable. If the variable is not set and the password is not specified in the application, the application may prompt for the password. If the application does not prompt for the password, an error is returned.

```
setenv SDEPASSWORD fools.gold
```

SDETMP allows you to set the temp directory for the servers using this variable, but it will only be checked if the TEMP keyword is not set in the SDE.SERVER_CONFIG table.

```
setenv SDETMP c:\temp
```

SDEDBECHO echoes the contents of the dbinit.sde file during startup. For ArcSDE application servers started locally on a UNIX system, the output of SDEDBECHO is written to the screen. The SDEDBECHO output for an ArcSDE application server started on a remote UNIX ArcSDE application server is written to its sde.errlog file.

```
setenv SDEDBECHO TRUE
```

SDEVERBOSE reports internal messaging to the screen upon startup and writes gsrvr process startup and shutdown messages to sde.errlog.

```
setenv SDEVERBOSE TRUE
```

Set SDEATTEMPTS to the number of times you want the session to attempt to connect to an ArcSDE application server. Under normal conditions, only one attempt is required; however, should the ArcSDE application server become too busy satisfying the connection requests of many users at the same time, several attempts may be required before a session is able to establish a connection. Each time a session fails to establish a connection, it waits until the IP time out occurs. The IP time out is set by your network administrator but defaults to 75 seconds on most operating systems. By default, SDEATTEMPTS is set to 4. This should be adequate for most environments.

```
setenv SDEATTEMPTS 4
```

Set SDENOIPTEST to true if you do not want ArcSDE to test for the presence of the SERVER name in the HOSTS file. By default SDENOIPTEST is not set. Setting this variable can be useful if you have not set up the HOSTS file. The client will attempt to connect to the server for SDEATTEMPTS (4 times by default).

```
setenvSDENOIPTEST TRUE
```

Set GIOMGRLOGREFRESH to true if you wish to have the giomgr.log file overwritten each time the ArcSDE server is started. If you wish to accumulate the log messages (the default), do not set the variable.

```
setenv GIOMGRLOGREFRESH TRUE
```

Set SDELOGAPPEND to true if you want the sde error logfile to accumulate log messages and be truncated each time the server is restarted. Do not set the variable if you want the sde error logfile to be truncated upon startup.

```
setenv SDELOGAPPEND true
```

ESRI_US_STREETS_DIR specifies the folder that stores the ArcGIS StreetMap™ USA reference data used by the ArcSDE StreetMap USA locators. Set this environment variable to the folder containing the usa.edg file and other .edg StreetMap USA street data files. Note that the value of this environment variable should not contain a trailing slash (“/”) or backslash (“\”).

```
setenv ESRI_US_STREETS_DIR
       d:\streetmap usa\usa\streets
```

LOCATION_ERRLOG defines the file to which ArcSDE location errors are logged. The value of this environment variable should contain the fully qualified name of the logfile. This file does not need to exist before setting the environment variable because the location framework creates it automatically. ArcSDE location errors will be logged to this file whenever the LOCATION_VERBOSE environment variable is set to TRUE.

```
setenv LOCATION_ERRLOG
       c:\temp\location.errlog
```

Set the LOCATION_VERBOSE environment variable to TRUE to log ArcSDE location errors. If the LOCATION_ERRLOG environment variable is also set, location errors are logged to the specified file. If the LOCATION_ERRLOG environment variable is not set, or the specified file cannot be created, location errors are logged to SDEHOME\etc\location.errlog.

```
setenv LOCATION_VERBOSE TRUE
```

The dbinit.sde file format

The dbinit.sde file consists of comments and commands. Comments are any lines preceded by the pound sign (#). For example:

```
# This is the system environment for
# the esri_sde ArcSDE application server
```

The commands in the dbinit.sde file accept two keywords: *set* and *unset*. The set command enables the system variable and assigns it the value following the equal sign. The syntax of the set command is:

```
set <variable>=<value>
```

In this example, the SDEDBECHO variable is set to true, which echoes the variables set in the dbinit.sde file when the ArcSDE application server is started.

```
set SDEDBECHO=TRUE
```

The unset command disables the system variable. It is useful because it ensures that an undesired variable set in the login environment is not set when ArcSDE starts. The syntax of the unset command is:

```
unset <variable>
```

The following example of the unset command ensures that the Oracle TWO_TASK variable is not set:

```
unset TWO_TASK
```

Displaying ArcSDE system environment variables

To display the environment variables for an ArcSDE application server that is currently running, use:

```
$ sdemon -o info -I vars
```

A list of environment variables is returned in the format:

```
<variable>=<value>
```

The variables in the list are a combination of those found in the login system environment file and the dbinit.sde. The variables set in the dbinit.sde file always override the system environment file settings.

On Windows systems the application server is started as a Windows service; therefore, it may not have the same environment setting as the logged in user's environment. Thus, there will probably not be agreement with the MS–DOS set command for the logged in user.

Adjusting ArcSDE application server initialization parameters

The ArcSDE application server initialization parameters stored in the SDE.SERVER_CONFIG table control the configuration of the application server. The parameters are read when the ArcSDE application server starts.

The default values satisfy the needs of most ArcSDE installations. With the exception of adjusting the MINBUFFSIZE and MAXBUFFSIZE parameters to improve data loading performance, the remainder of the parameters should not be altered unless you are certain you need to do so.

ArcSDE applications making direct connects to a DBMS also read the parameters from the SDE.SERVER_CONFIG table. Not all initialization parameters that apply to the ArcSDE application server apply to ArcSDE applications making a direct connection.

See 'Appendix D: ArcSDE initialization parameters' for a full list of initialization parameters. The following sections discuss in more detail some of the individual parameters and their impact on the ArcSDE application server.

READONLY parameter

The READONLY parameter allows you to start the ArcSDE application server in READONLY mode. Set this parameter to TRUE if you want to disable all writes by ArcSDE clients.

When the READONLY parameter is set to FALSE, the default value, the ArcSDE application will allow users to edit the data of the ArcSDE feature classes and tables that they have write permissions for.

Session parameters

The session parameters are CONNECTIONS, TEMP, and TCPKEEPALIVE.

The CONNECTIONS parameter restricts the number of concurrent connections allowed by an ArcSDE application server.

The CONNECTIONS parameter does not restrict the number of direct connections. The default is 64 on UNIX and 48 on Windows.

The TEMP parameter specifies the full path to the directory that the ArcSDE application server uses for temporary things, such as the shared memory file on UNIX systems and temporary storage for the attribute binary large objects (BLOBs), that are larger than specified by the BLOBMEM parameter. For more information on this parameter, please refer to the section 'Managing BLOB data' later in this chapter. A directory with at least 5 MB of available space should be specified. More space may be required for ArcSDE application servers that support applications allowing users to concurrently access binary large objects stored in attribute BLOB columns.

Note that some files created by the ArcSDE application server are always stored in the /tmp directory. For instance, the s.sde<service>.iomgr file is a UNIX protocol socket used for connections to the ArcSDE application server. It must always be created in a known location so that local clients, unaware of the SDEHOME location and the setting of TEMP, can find it. Thus it is always created in /tmp.

Sometimes the computer on which an application is running crashes or a user unexpectedly terminates an application. Unless the TCPKEEPALIVE parameter is set to TRUE, the ArcSDE application server process does not detect the absence of the client process and does not disconnect from the DBMS server. When this happens, the ArcSDE application server process remains, and the administrator must manually terminate it.

The TCP/IP KEEPALIVE interval is a systemwide parameter that affects all application server processes running in the TCP/IP environment.

By setting the ArcSDE TCPKEEPALIVE parameter to TRUE, the system TCP/IP KEEPALIVE settings are used. The default test interval is two hours of idle time—that is, the system checks the connected sessions every two hours.

When this happens, any ArcSDE connections whose client processes have been terminated are disconnected. If the network environment in which the ArcSDE application server operates is reliable, TCPKEEPALIVE may be set to TRUE. However, be aware that a disconnection may be triggered by short-term network outages (~10 minutes) when TCPKEEPALIVE is set to TRUE. By default, TCPKEEPALIVE is set to FALSE.

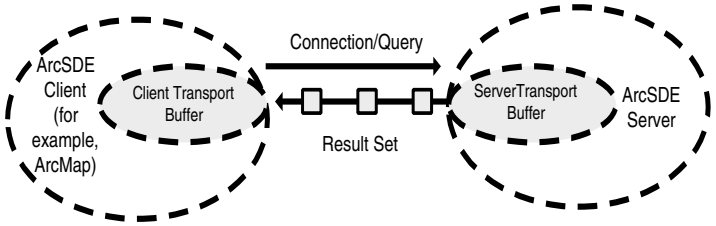
The TCPKEEPALIVE parameter does not affect direct connections.

Transport buffer parameters

After a user connects to an ArcSDE application server from an application, such as ArcMap™ or ArcCatalog™, the feature classes and tables stored in the database can be accessed. Each time any of these items are accessed, an ArcSDE stream is created. An ArcSDE stream is a mechanism that transports data between the database the ArcSDE application server is currently connected to and an application such as ArcMap. For further information on streams and the parameters available to control their operation, see 'Streams' later in this chapter.

When an ArcSDE stream is created, the ArcSDE application server process allocates transport buffers on both the client and the server. Transport buffers reduce I/O and improve performance by accumulating records and sending them across the network in batches rather than as individual records.

The records are collected in the ArcSDE application server process' transport buffer and sent to the ArcSDE client transport buffer when the application is querying the database. Alternatively, the records are collected in the client's transport buffer and sent to the ArcSDE application server process transport buffer when the application is writing data to the database.



Client/Server transport buffers

Three parameters control the transport buffers.

`MAXBUFSIZE` The total amount of memory allocated to each transport buffer

`MINBUFSIZE` The minimum threshold of each transport buffer

`MINBUFOBJECTS` The minimum number of records per transport buffer

The MAXBUFSIZE represents the total amount of memory that is allocated to each transport buffer. The transport buffer stops accumulating records once MAXBUFSIZE is reached and waits for the request to send the records to the other buffer.

The MINBUFSIZE and MINBUFOBJECTS parameters are lower thresholds that prevent the records from being sent until one of them is attained. For example, if the application requests data, the request is deferred until the server transport buffer reaches either MINBUFSIZE or MINBUFOBJECTS.

When creating a geodatabase, you should raise the parameters to increase the transmission speed of data loading. Once loading is complete, reduce the transport buffers.

Before raising the MAXBUFSIZE parameter too high, consider the maximum overall impact to the server's memory budget. The default MAXBUFSIZE adds 64 kilobytes per stream per ArcSDE user connection. For example, if there are 100 users using an application that displays seven feature classes, ArcSDE allocates a total of 44,800 kilobytes (100 * 7 * 64) of memory for the server's transport buffers. Doubling the MAXBUFSIZE in this example would result in 89,600 kilobytes of memory allocated to the server's transport buffers. Excessive paging may result on the server if MAXBUFSIZE is set too high and physical memory is not available to satisfy it.

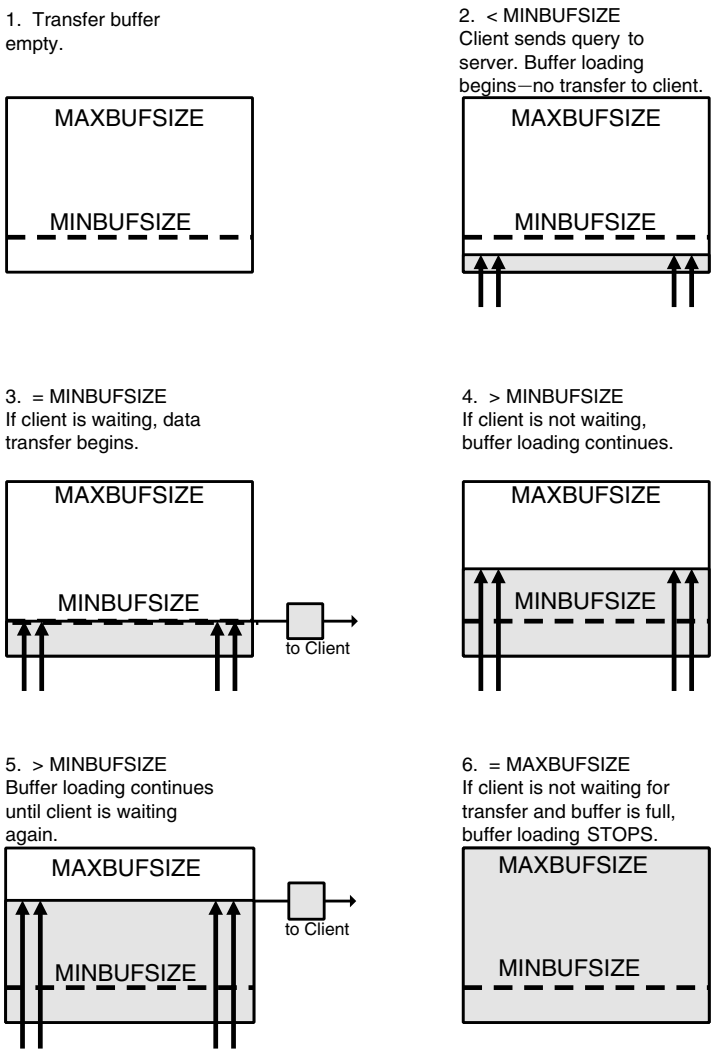
Set MINBUFSIZE to no more than one-half of the MAXBUFSIZE.

Setting MINBUFSIZE too high increases wait time. If MAXBUFSIZE is 64K and MINBUFSIZE is 56K, the client will wait until the 56K threshold is reached before sending the transport buffer.

Reducing the MINBUFSIZE parameter improves the cooperative processing between client and server.

MINBUFOBJECTS depends on the size (bytes) of a row of data. MINBUFOBJECTS is examined first, then MINBUFSIZE. If the threshold of either parameter is breached, the contents of the transport buffer are sent.

ArcSDE application developers can override the giomgr.defs transport buffer parameters with the C API function SE_connection_set_stream_spec.



The data transfer process from an ArcSDE application server transfer buffer to an ArcSDE client

Array buffer parameters

For each ArcSDE stream that is created, ArcSDE allocates an array buffer.

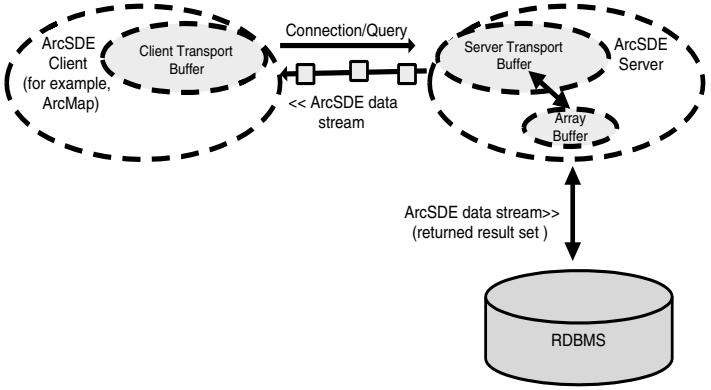
Whenever possible, groups of records are transferred between ArcSDE and the DBMS server. For DBMSs that have implemented array inserts, ArcSDE inserts records into the DBMS server array. All supported DBMSs, except Microsoft Access, have implemented array fetch mechanisms, so whenever ArcSDE issues a select statement on behalf of the user, it fetches or gets the records in an array.

Array buffers reduce the amount of I/O occurring between ArcSDE and the DBMS server by fetching and inserting data in larger chunks. Less I/O results in better performance. However, excessive paging can result if the array buffers are set larger than necessary. As a general rule, approximately 100 records is the optimum number that most applications can handle in the array process.

The array buffer parameters determine the size of the array buffers. Parameters that affect query performance are SHAPEPTSBUFSIZE, ATTRBUFSIZE, MAXARRAYSIZE, and MAXARRAYBYTES.

```
MAXARRAYSIZE       100      #Max.array fetch size
MAXARRAYBYTES    550000     #Max.array bytes
                             allocated per stream
SHAPEPTSBUFSIZE  400000     #Shape POINTS array
                             buffer size
ATTRBUFSIZE       50000     #Attribute array
                             buffer size
```

The array buffer parameters define the number of records transferred together for insert and query operations. During an array insert, ArcSDE fills the array buffer with records from the server transport buffer and transfers the entire array to the DBMS. ArcSDE queries data in an array by using array buffers to receive the data.



ArcSDE server array buffers

Estimating SHAPEPTSBUFSIZE

Array inserts of feature geometry are divided into two parts: feature metadata and point data. Feature metadata, including the feature ID, number of points, and entity type, requires the same amount of space for each feature. The space required to store the point data, on the other hand, varies according to the number of points in the feature.

SHAPEPTSBUFSIZE determines the buffer size for the point data. The default setting is based on an array size of 100. Tuning SHAPEPTSBUFSIZE to the optimal setting is critical to performance. ArcSDE estimates the average size of all features based on the array size (MAXARRAYSIZE) and the size of the points buffer (SHAPEPTSBUFSIZE). If a feature exceeds the average size, it is flagged as truncated and fetched separately.

For example, suppose SHAPEPTSBUFSIZE is 400,000 (400K), and the array size per fetch (MAXARRAYSIZE) is 100 rows (or 100 features). The features in this example do not have annotation, z-values, or measures, so each point requires eight bytes (four bytes for the x-value and four bytes for the y-value).

Dividing SHAPEPTSBUFSIZE by MAXARRAYSIZE returns the maximum space available for each feature's points within the array buffer.

```
400000 bytes / 100 = 4000 bytes
```

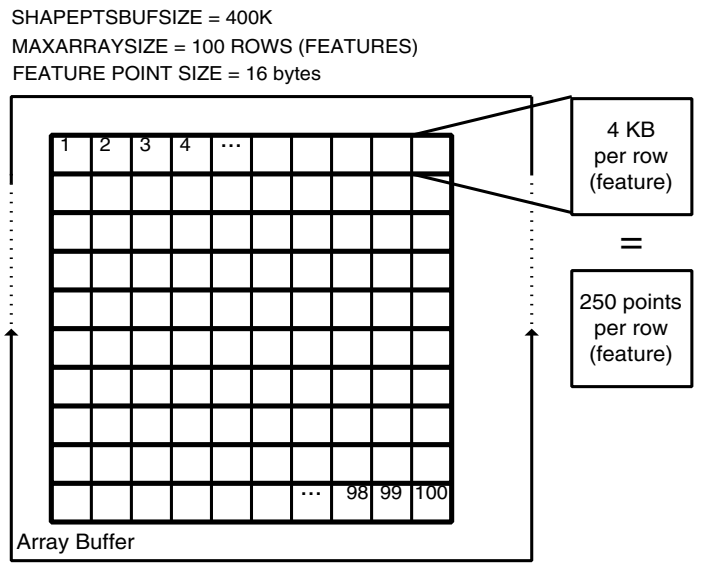
Further dividing the maximum space available to each feature by the number of bytes each point requires returns the total number of points that can be stored in the maximum space available for each feature.

```
4000 / 16 = 250
```

In this case, ArcSDE expects each feature to have no more than 250 points. When ArcSDE encounters features with 251 or more points, the feature is skipped and fetched separately, forcing an additional I/O fetch from the DBMS.

If the shape contains z-values or measures, divide by 24 instead of 16. If the shape contains both z-values and measures, divide by 32.

Setting the SHAPEPTSBUFSIZE parameter too small results in a higher number of features to be fetched individually, which diminishes performance. Setting this parameter too high wastes memory.



The SHAPEPTSBUFSIZE and MAXARRAYSIZE parameters determine the number of points that can be stored in the maximum space available for each feature within an ArcSDE array buffer.

Tip

Point size

The number of bytes required by each point depends on its coordinate type. If the feature class only has x,y coordinates, the number of bytes each point requires is 16. If either z-values or measures are present, 24 bytes are required. If both z-values and measures are present, 32 bytes are required.

Determining 90 percent of the most queried feature classes for an optimum SHAPEPTSBUFSIZE setting

1. Determine the total number of features in the feature class by issuing this SQL query.
2. Determine the largest number of points in the feature class to use as a starting point in the step.
3. Through iteration, determine the number of points 90 percent of the features have. Start by using an x-value that is 0.75 * maximum numofpts. Increase or decrease the x-value until the feature count is 90 percent of total features.
4. Multiply the number of points by the number of rows (MAXARRAYSIZE) the array will hold and the number of bytes required by each point.

   For instance, if MAXARRAYSIZE is set to 100, 90 percent of the features have 710 or fewer points, and each point requires 8 bytes, set the SHAPEPTSBUFSIZE to 568000.

Queries used under the ArcSDE binary model (SqlServer, Oracle LONG RAW and Oracle LOB).

1. `select count(*) “total features” from f<n>;`
2. `select max(numofpts) “maximum numofpts” from f<n>;`
3. `select count(*) 'feature count' from f<n> where numofpts < X;`
4. `SHAPEPTSBUFSIZE = (MAXARRAYSIZE) * (number of points) * (point size)`

   `568000 = 100 * 710 * 16`

Queries used under the object relational model (DB2, Informix and Oracle Spatial)

1. `select count(*) 'total features' from table;`
2. `select max(numpoints<spatial_column>)) 'maximum numofpts' from <table>;`

   `- DB2 and Informix`

   `select max((select count(*) from table(p.shape.SDO_ORDINATES))/2) * points from <table>;`

   `- Oracle Spatial`
3. `select count(*) 'feature count' from table where numpoints(<spatial_column>) < X;`
4. `SHAPEPTSBUFSIZE = (MAXARRAYSIZE) * (number of points) * (point size)`

   `568000 = 100 * 710 * 8`

Estimating ATTRBUFSIZE

ATTRBUFSIZE defines the array buffer size for attribute (non-BLOB) data. Tuning the attribute array buffer is similar to the SHAPEPTSBUFSIZE. The default 50,000 setting allows 100 rows with 500 bytes of attribute data each.

Performance is affected when the number of rows that can be fetched into the attribute buffer does not match the MAXARRAYSIZE parameter setting. For queries involving multiple columns, add the number of bytes per column to get a total row size. The ATTRBUFSIZE divided by row size cannot exceed the number of rows specified by MAXARRAYSIZE. ArcSDE will automatically reduce the size of the attribute buffer to hold MAXARRAYSIZE rows.

Setting MAXARRAYSIZE

As mentioned, MAXARRAYSIZE sets the number of rows that the server will fetch per request. The recommended default value is 100. Optimal values can range between 20 and 150 on different platforms and DBMSs. Once the shape points data (SHAPEPTSBUFSIZE) and attribute buffer (ATTRBUFSIZE) are correctly tuned, try several array sizes to determine the optimal setting for each installation.

Setting MAXARRAYBYTES

Control the maximum number of bytes per stream with the MAXARRAYBYTES parameter. This value represents the total bytes that can be allocated to both ATTRBUFSIZE and SHAPEPTSBUFSIZE for each stream.

MAXARRAYBYTES is simply a way to manage the memory allocations for array buffers on the server. The sum of ATTRBUFSIZE and SHAPEPTSBUFSIZE must be less than or equal to MAXARRAYBYTES. If it isn't, the ArcSDE application server will not start. If this problem occurs, either increase MAXARRAYBYTES or decrease either ATTRBUFSIZE or SHAPEPTSBUFSIZE.

This value cannot be changed with the SE_connection_set_stream_spec function and can only be altered in the sde.server_config table by the ArcSDE administrator.

Managing BLOB data

The BLOB parameters, MAXBLOBSIZE and BLOBMEM, determine the server-side buffer requirements for BLOB data types. MAXBLOBSIZE determines the maximum number of bytes the server will accept. A BLOB is either written to memory or disk, depending on the size. BLOBMEM determines the maximum BLOB size for in-memory storage. That is, the server allocates BLOBMEM bytes to hold the BLOB. A BLOB that exceeds this size is written to disk.

Maximum initial features (Oracle only)

The MAXINITIALFEATS parameter controls the initial features argument that may be submitted to either the sdelayer command or the SE_layer_create function. Based on the value of initial features, sdelayer or SE_layer_create calculates the initial and next extent of the feature and spatial index table. To prevent the possibility of a user inadvertently entering a large initial features value and, thus, acquiring an initial extent beyond what is needed, the administrator can limit the initial features value by setting the MAXINITIALFEATS parameter.

Statistics

The MAXDISTINCT parameter controls the number of distinct values a call to either the SE_table_calculate_stats or the SE_stream_calculate_table_statistics function can return.

Setting this parameter to 0 allows an unlimited number of distinct values to be returned by the applications that call these functions. However, the distinct values are generated in memory on the server and passed to the client's memory when the list is complete. Calculating the statistics of a large table could pose a threat to the client and server resources. A prudent database administrator will set this value high enough to allow most queries to complete, but not so high as to expose the server or the client application to a memory shortage. Should a user receive the error message SE_TOO_MANY_DISTINCTS, the MAXDISTINCT parameter may be raised, but this should be done cautiously as it impacts both client and server memory. It may be advisable to examine the applications to determine whether queries could be performed more efficiently.

Streams

You have read about streams in the section 'Transport buffer parameters' earlier in this chapter. To recap briefly, each time a user accesses a feature class or business table, an ArcSDE stream is created to transfer the information between the client and the server. An ArcSDE stream is a mechanism that transports data between the DBMS database and an application such as ArcMap. The resources allocated to each stream are significant; it is, therefore, important that the ArcSDE administrator understands how they operate and how to control the impact they have on the resources of the ArcSDE host computer.

The administrator controls the behavior with the STREAMPOOLSIZE parameter.

STREAMPOOLSIZE

The process of creating a stream requires the allocation of memory and other resources. These are released when users close the stream, which normally happens when they disconnect from the ArcSDE application server.

If the STREAMPOOLSIZE is set to a value greater than 0, ArcSDE creates a pool of released stream resources for reuse. In this case, when a user releases a stream, the ArcSDE application server first checks the stream pool to determine if it is full. If it is not, the released stream resources are added to the pool; otherwise, the resources are deallocated. When the same user application creates a stream, the ArcSDE application server checks the stream pool for an available stream. If one exists, ArcSDE removes it from the pool and allocates it to the user application.

Raster parameters

ArcSDE stores images similar to the way it stores features, in a DBMS BLOB column. For more information about raster tables, see 'Appendix B: ArcSDE table definitions'.

RASTERBUFSIZE

The RASTERBUFSIZE parameter controls raster data transfer, which operates in a fashion similar to streamed data (see 'Transport buffer parameters' and 'Array buffer parameters' in this chapter).

The RASTERBUFSIZE is specified in bytes and must be large enough to hold the largest raster tile. The default ArcInfo tile size is 128 * 128 pixels. Data that is eight bits per pixel has a tile size of 16,384 bytes (128 * 128).

The raster transfer includes both an array buffer and transport buffers. The raster array buffer is set at two times the

RASTERBUFSIZE parameter, while the raster transport buffers are set to the RASTERBUFSIZE. Therefore, the memory allocated to raster transfer on the server is three times RASTERBUFSIZE. On the client, RASTERBUFSIZE bytes of memory are allocated to the client raster transport buffer. The raster buffers are allocated when raster tiles are accessed by a stream. The raster buffers are not deallocated until the stream is closed (unless the stream is added to the stream pool—see 'STREAMPOOLSIZE'). In other words, a user can elect to display an image and a feature class together in ArcMap. The raster buffers are allocated on demand when a user selects an image stored in the feature class's business table.

If the ArcSDE application server encounters a raster tile that does not fit into the transport buffer, the SE_RASTERBUFFER_TOO_SMALL error is returned. If users report this error, determine the tile size they are using. If they are using a 256 x 256 tile size on an 8-bit image, the RASTERBUFSIZE must be at least 65,536 bytes (256 * 256 * 1). The pixel depth must be taken into account when calculating the number of bytes per pixel. For example, an image with a 64-bit pixel depth and a tile size of 256 * 256 requires a RASTERBUFSIZE of 524,288 bytes (256 * 256 * 8). If memory is at a premium, advise users to specify a smaller tile size rather than raise the RASTERBUFSIZE.

Maximum time difference

The maximum time difference that a client's system clock can deviate from an application server's system clock can be set with the giomgr.defs MAXTIMEDIFF parameter. The parameter is specified in seconds. It prevents an unauthorized entry by individuals who may have captured a network packet with a sniffer software that contains an ArcSDE connection string. Because the encrypted password is time stamped, the packet cannot be resent at a later time if MAXTIMEDIFF is set low enough (for example, 60 seconds).

If legitimate connections receive a "-99 password received was sent 7 MAXTIMEDIFF seconds before" error, reset the client machine's system time to the host's system time.

To disable MAXTIMEDIFF, set it to -1.

The MAXTIMEDIFF parameter does not affect direct connections.

Controlling transactions

The AUTOCOMMIT parameter lets you control the commit rate of large transactions and prevents the DBMS logfiles that store rollback information from filling. The default value of 1,000 specifies that a transaction is committed whenever it reaches 1,000 updates. Setting this parameter to 0 disables automatic commits and requires the application to explicitly commit the transactions.

Logfiles

ArcSDE provides logfiles to store lists of table rows. ArcMap, for example, stores selection sets exceeding 100 rows in a logfile. Physically, the list of row IDs, referred to as the logfile, are stored in a logfile data table.

At ArcSDE version 9, enhancements to the logfile data storage were introduced to solve problems experienced by some applications using the ArcSDE shared logfile method.

ArcSDE 8.3 and prior releases store logfiles in the SDE_LOGFILES and SDE_LOGFILE_DATA tables created in each users schema. By default these logfile tables were created by ArcSDE the first time the user connected and all deletes from the SDE_LOGFILE_DATA table were deferred until the user terminates the session. This type of logfile data storage is referred to as the ArcSDE shared logfile method.

ArcIMS users, and to some extent ArcGIS users, experienced problems with the deferred delete functionality of the ArcSDE

shared logfile method. The number of rows contained within the SDE_LOGFILE_DATA table could grow in size, reaching millions of records, until the user disconnected. The deletion of the rows following the disconnect could take several minutes. Starting at ArcSDE 9, all logfile entries are deleted immediately.

Also, since the logfile tables must be created in the user's schema, DBMS administrators were required to grant privileges to each user that would allow them to create the required data objects.

Another shortcoming of the ArcSDE shared logfile method is that all sessions connecting as the same user wrote to the same table. This contributed to the inflated growth of the SDE_LOGFILE_DATA table.

Enhancements to the logfile system beginning with ArcSDE 9 allow administrators to create standalone and session logfiles, as well as pools of logfiles owned by the sde user that can be checked out by other users.

Therefore, beginning at ArcSDE 9, logfile data can be stored in one of three kinds of tables that include standalone, session, and shared.

Standalone logfile data tables

Standalone logfile tables contain a single logfile. When the logfile is deleted, the table is immediately truncated to remove the records. The truncation avoids the costly logging of changes to the DBMS.

Standalone logfiles are created in the user's schema only when none are available for checkout from the sde user's logfile pool.

Standalone logfiles are used until the user's quota, defined by the MAXSTANDALONELOGS server configuration parameter, is exhausted.

Standalone logfiles are checked out of the logfile pool until the pool is exhausted. The total number of logfiles in the pool is defined by the SESSIONLOGPOOLSIZE server configuration parameter. Set the HOLDLOGPOOLTABLE parameter to TRUE if you want users to retain the checked out logfiles following a logfile delete.

Session logfile data tables

Session logfile data tables are dedicated to a single session and may contain multiple logfiles. When a logfile is deleted, the records are immediately removed from the session logfile data table. The table is truncated if no other logfiles are present; otherwise the logfile records are deleted.

Session logfiles are used if the ALLOWSESSIONLOGFILE parameter is set to TRUE and the user's quota of standalone logfiles has not been exhausted.

Session logfiles are created in the user's schema when they cannot be checked out from the sde user's logfile pool.

Shared logfile data tables

ArcSDE reverts to using the ArcSDE shared logfile data table, the default, if it cannot use a standalone or session logfile data table. This will happen if MAXSTANDALONELOGS is exhausted and ALLOWSESSIONLOGFILE is FALSE. If the shared logfile tables cannot be created, ArcSDE returns an SE_NO_ACCESS (-15).

SESSIONLOGPOOLSIZE

Specifies the number of logfiles that may be borrowed from the sde user's logfile pool. Once the users have exhausted the pool, ArcSDE will attempt to create either the standalone or session logfile in the user's schema. By default the logfile pool is set to 0. If you do not want your users creating logfiles in their own schemas, set this value high enough that all logfiles are borrowed.

ALLOWSESSIONLOGFILE

Allows your users to use a session logfile. By default this parameter is set to FALSE.

HOLDLOGPOOLTABLE

Set to TRUE, this parameter directs the user session to retain a logfile table that it has borrowed from the sde user until the session disconnects from ArcSDE. Setting the parameter to FALSE will result in the sde user's logfiles being shared among a greater number of sessions at the expense of the allocation or deallocation of the logfile tables. By default this parameter is set to TRUE.

MAXSTANDALONELOGS

The maximum number of standalone logfiles a user may use. By default this parameter is set to 0.

States

ArcSDE uses states to manage the chronological order of changes made to the registered tables of the geodatabase. States are added, updated, and deleted as are other objects of the database. As a change is made to a registered table, states are added to the database. States are deleted when versions are removed from the database and the state tree is trimmed. Changes stored in the form of adds and deletes in tables associated with the registered tables are coalesced into the base tables. The intensity with which ArcSDE interacts with the states tree depends on the application and the number of users. Periodically, you need to compress the database using either ArcCatalog or the sdeversion administration command to reduce the size of the state tree. The following parameters help to reduce the interaction with the state tree but should be used to replace or delay necessary maintenance of the state tree.

STATEAUTOLOCKING

By default the autolocking of the states is disabled (set to FALSE) and should not be set to TRUE if you are editing multiversioned tables with applications constructed by ESRI. ESRI's applications are designed not to delete or alter states that are currently opened by other sessions. However, if you are using a non-ESRI application that manages states external to ArcObjects™ you should set this parameter to TRUE. ESRI's applications will then lock states in use to protect them from being deleted by the third party application. Turning STATEAUTOLOCKING on adds overhead to management of the state tree and does adversely affect the performance of the ArcGIS desktop applications attempting to perform edits on the geodatabase.

Collecting session statistics

The PROCSTATS parameter specifies the optional update interval, expressed in seconds, by which ArcSDE updates the SDE.PROCESS_INFORMATION table with each user's session statistics.

By default it is set to -1, which disables the writing of session statistics to the SDE.PROCESS_INFORMATION table.

Set PROCSTATS to 0 if you want a session to update the PROCESS_INFORMATION table whenever the statistics change.

Set PROCSTATS to a value greater than 0 to periodically update the PROCESS_INFORMATION table with the session's statistics. Statistics are written if it has been more than the number of seconds specified since the last update. For example, if PROCSTATS is set to 10, the PROCESS_INFORMATION table is updated at a minimum interval of 10 seconds and only if a change has occurred.

Updating the PROCESS_INFORMATION table adds overhead to the session. Therefore, the PROCSTATS variable should only be altered from its default if you need to collect session statistics for debugging purposes.

The statistics stored in the PROCESS_INFORMATION table can be accessed directly through a SQL query on the table. Alternatively, you can access them using the *sdemon -o info -I stats* administration tool or through the ArcSDE C-API functions *SE_connection_get_instance_statistics* and *SE_instance_get_statistics* or through the ArcSDE Java API function *SeInstance.SeInstanceStats*.

As of ArcSDE 9, a summary statistic is not written to the giomgr.log file when a session disconnects from an application server if PROCSTATS is set to -1, the default value.

Changing the precision of the geometry storage

By default geometry is stored in a 32-bit integer. ArcSDE converts the geometry from world coordinates to integers by applying the offset, multiplying by the system units, and truncating the remainder before storing the value into the integer. The 32-bit integer satisfies the storage requirements of most datasets; however, if your dataset requires a high degree of precision and covers an extensive area, you may find that it will not fit within a 32-bit integer, in which case your should store your spatial data in a 64-bit geometry column. To change the default geometry storage characteristic of your instance, set the DEFAULTPRECISION parameter to 64. The default value is 32.

NULL features and the Oracle8*i* optimizer

If you are using ArcSDE for Oracle8*i* and at least some of your feature classes contain NULL features, you should consider setting the DETECT8XNULLSHAPE parameter to TRUE. Doing so will use a more optimal execution plan for queries that access these feature classes. If you do not have any NULL features or you are not experiencing performance problems with such feature classes, do not set this parameter to TRUE.

Changing the precision to the ArcSDE long integer

Prior to ArcSDE 8.1 for Oracle, the maximum number of digits stored in an ArcSDE long integer was 10. However, this meant that any value exceeding 2,147,483,648 or less than -2,147,483,649 could not be read into a C signed long integer. Therefore, the long integer value was shortened to nine digits and stores a maximum value of 999,999,999 and a minimum value of -999,999,999. However, legacy data created with a prior version of ArcSDE for Oracle may have long integer data exceeding nine digits. If this is the case, you can set the PRECISION10 parameter to TRUE. Doing so will allow ArcSDE to read these long integer values.

Disabling 64-bit integers

Although ArcSDE has added support for 64-bit integers, some applications may not have. If this is the case, you must set the INT64TYPES parameter to false. Doing so will prevent applications from creating fields with the SE_64INT_TYPE and will display existing 64-bit integer fields as double precision.

Setting the time last modified interval

Periodically ArcSDE applications query the ArcSDE object metadata tables for XML columns, spatial columns, and raster columns to search for new objects or changes to existing objects. To avoid actually querying the metadata tables, ArcSDE will first consult the SDE.SDE_TABLES_MODIFIED table to determine if the metadata table itself was modified since the last time the application queried it. If no change has been recorded for the metadata table, the ArcSDE application uses the object's cached information.

The TLMINTERVAL parameter allows you to set the number of seconds that must elapse before the application can consult the SDE.SDE_TABLES_MODIFIED table for changes to an object's metadata table. If the schema of your database is static, you can set the TLMINTERVAL very high, which directs the applications to never consult the SDE.SDE_TABLES_MODIFIED table and always use the object's cached metadata.

Disabling the automatic registration of object relational spatial data types

ArcSDE hosts three object relational spatial data types, Informix DataBlade, DB2 Spatial Extender, and Oracle Spatial. In each implementation it is possible for users to create tables that contain one of these data types outside of an ArcSDE application. Unless the DISABLEAUTOREG parameter is set to TRUE, during startup the ArcSDE application server scans the DBMS metadata for any new spatial columns that are not included in the ArcSDE registry. If it finds one, it adds references to the table and the new spatial column in the ArcSDE metadata tables. If you prefer to manually control the addition of these spatial columns to the ArcSDE metadata, set DISABLEAUTOREG to TRUE. By default this parameter is FALSE.

Enabling layer autolocking

As of ArcSDE 9, layer autolocking has been disabled by default. Layer autolocking can be enabled for the ArcSDE server by setting the server configuration parameter LAYERAUTOLOCKING to TRUE, in which case layers that have their autolocking property enabled will autolock shapes when they are edited in NORMAL_IO mode. By default, a layer is created with its autolocking property enabled. To disable autolocking for a particular layer, use the sdelayer administration command's alter operation.

Layer autolocking occurs when the shape of a qualified layer is edited. The area within the shape's envelope is automatically locked, prohibiting other sessions from editing shapes whose envelope intersects the locked area.

Layer autolocking is disabled by ArcSDE if any of the following conditions are true:

- The server configuration parameter LAYERAUTOLOCKING is set to FALSE.
- The layer's autolocking property is disabled.
- The layer is in LOAD_ONLY_IO mode.
- The layer's business table is multiversioned.

Displaying the ArcSDE initialization parameters

You can list the current ArcSDE initialization parameters using the sdemon command with the "-o info -I config" options.

```
C:\> sdemon -o info -I config
ArcSDE I/O Manager Configuration Parameters at
Fri Feb 14 15:52:43 2003
_____________________________________-

ArcSDE Version                      9.0
ArcSDE Server Build                 for <your DBMS>
Underlying DBMS                     <your DBMS>
Max. Server Connections             64
Root Path                           D:\arcsde
Temp Path                           C:\Temp
Min. Transmission Buffer Size       16384 Bytes
Max. Transmission Buffer Size       65536 Bytes
Min. Transmission Buffer Count      512 Objects
Max. Initial Features               10000 Objects
Max. stream pool size               3 streams
Max. BLOB Size                      1000000 Bytes
Max. in Memory BLOB Size            1000000 Bytes
Max. Distincts                      512
Autocommit Frequency                1000
Shape Point Buffer Size             400000 Bytes
Attribute Buffer Size               50000 Bytes
Max. Array Size                     100
Max. Array Bytes                    550000 Bytes
Max. Client/Server Time Diff.       Unlimited
State Caching                       On
Stream State Autolocking            Off
Instance name                       arcsde
Port Number                         9000/tcp
TCP/IP Keepalive on Connections     Off
Instance Type                       Read/Write
Max. Raster Buffer Size             102400 Bytes
Default Layer Precision             32-bit
Time last mod (TLM) interval (sec) 1
Num. of logfiles in logfile pool    0
Max. num. of standalone logfiles    0
Allow session logfiles              On
Hold log pool tables                On
Connection counts                   0
Layer Autolocking                   Off
Allow Int64 columns                 Off
```

Managing ArcSDE application servers 4

IN THIS CHAPTER

- **Before starting an ArcSDE application server**
- **Starting a local ArcSDE application server on Windows**
- **Starting a remote ArcSDE application server on Windows**
- **Starting a local ArcSDE application server on UNIX**
- **The sdemon command output**
- **Starting a remote ArcSDE application server on UNIX**
- **Pausing, resuming, and shutting down an ArcSDE application server**
- **Removing ArcSDE sessions (Windows and UNIX)**

The administrator who manages the ArcSDE application servers can place the server in one of three states: running, paused, or shutdown. Although it is desirable to maintain the application server in the most productive state (running), at times it is necessary to shut it down.

The tools you use to manage an application server depend on whether it has been installed on a Windows or a UNIX system. On UNIX systems, the administrator uses the sdemon command. Application servers installed on Windows are started and stopped from the Services menu. In both cases, the sdemon command is used to pause and resume the application server.

Application servers installed on either Windows or UNIX can be managed from a remote computer using the sdemon command. However, only a remote UNIX system can manage application servers installed on a UNIX system.

Before starting an ArcSDE application server

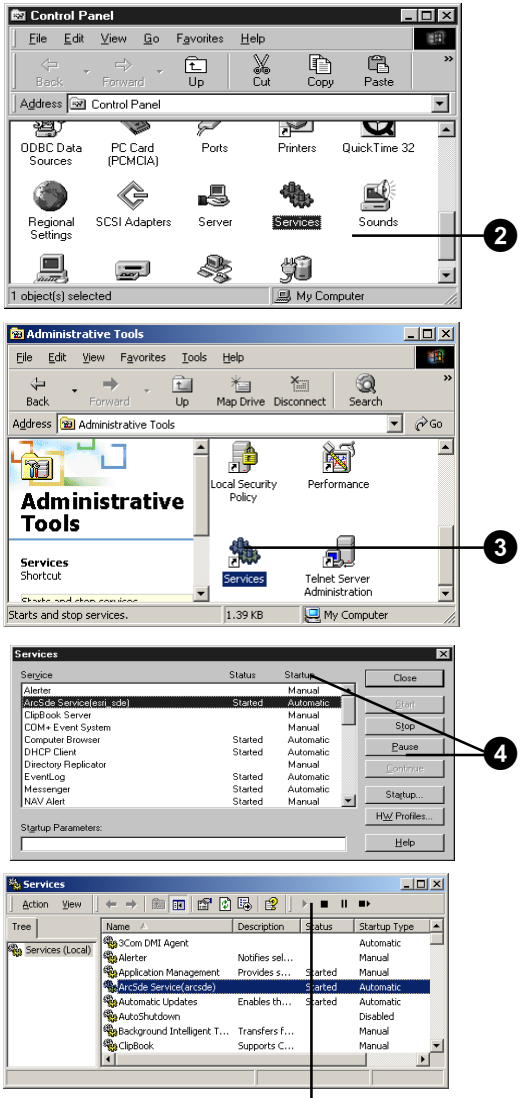
Before you can start the ArcSDE application server, you must satisfy the following conditions:

- The database management system instance must be started.
- The DBMS sde user account must exist.
- The system environment must be set such that the ArcSDE application server can connect to the DBMS as the sde user.
- The ArcSDE home directory must exist.
- An ArcSDE server license must have been installed.

Starting a local ArcSDE application server on Windows

You can start an ArcSDE application server on Windows from the Services menu. The name of the service always begins with “ArcSde service”, and the service name itself is enclosed in parentheses—for example:
ArcSde service(arcsde).
If the service fails to start, make a note of the Windows error number and see the ‘Common ArcSDE startup problems on Windows servers’ in Chapter 6, ‘Troubleshooting the ArcSDE application server’.

1. To open the Control Panel, click Start, click Settings, and click Control Panel.
2. Double-click Administrative Tools to open the Administrative Tools menu.
3. Double-click Services to open the Services menu.
4. Click the ArcSDE application server name and click Start Service arrow. The status changes to Started.



The status changes to Started, and the Start Service arrow is unavailable.

Starting a remote ArcSDE application server on Windows

You can start a remote Windows ArcSDE application server from another Windows machine.

The remote computer must be accessible over the network. The ping command can be used from an MS–DOS prompt to determine whether the remote ArcSDE application server host can be reached.

Remote startup is initiated using the sdemon command from an MS–DOS command prompt. The -s <server> and -i <service> options identify the remote host computer and the remote ArcSDE application server.

Tip

ArcSDE administrator Windows user group

The ArcSDE administrator must belong to the Windows administrator or power user group on the remote machine and have access via the system's environment variables to the sdemon command.

See Also

For more information on the various sdemon command options, see 'Appendix C: ArcSDE application server command references'.

Using the ping command to verify a remote network connection

1. At the MS–DOS command prompt, type the command "ping" followed by the name or TCP/IP address of the remote computer.

```
C:\> ping bruno

Pinging bruno.esri.com [46.1.1.92] with 32
bytes of data:

Reply from 46.1.1.92: bytes=32 time<10ms
TTL=128
Reply from 46.1.1.92: bytes=32 time<10ms
TTL=128
Reply from 46.1.1.92: bytes=32 time<10ms
TTL=128
Reply from 46.1.1.92: bytes=32 time<10ms
TTL=128

C:\>
```

Starting a remote Windows ArcSDE application server

1. To start the remote ArcSDE application server, "arcsde", on host computer "Bruno", type the sdemon command and include the "-s bruno -i arcsde" options.

```
C:\>sdemon -o start -p my_password
    -s bruno -i arcsde
```

Starting a local ArcSDE application server on UNIX

The sdemon command manages ArcSDE application servers configured on UNIX systems.

Tip

Starting the ArcSDE application server on UNIX

You must be logged in as either the owner of the ArcSDE application server's home directory, $SDEHOME, or as the user "root" to start an ArcSDE application server.

Tip

The sdemon command with -p option

You may enter the password as part of the sdemon command—for example, "$ sdemon -o start -p my_password"*—but the password will be displayed on the screen.*

1. Type the command sdemon with the "-o start" option to start the ArcSDE application server.
2. You will be prompted to type in a password. The password will not be displayed on the screen for system security.

❶ `$ sdemon -o start`

❷ `Please enter the ArcSDE DBA password:`

The sdemon command output

The output of the sdemon command during startup looks like this:

```
_____________________________________

ESRI ArcSDE I/O Manager - Release 9.0 - Tue Jun
24  13:59:40 PST 2003

_____________________________________
DBMS Connection established...
RDBMS:                          "Oracle"
Instance Name:                  "topo"
IOMGR Process ID (PID):          21891


ArcSDE Instance topo started Wed Aug 20 08:48:44
2003
```

Starting a remote ArcSDE application server on UNIX

Before an ArcSDE application server on a UNIX system can be started from a remote UNIX or a Windows machine, you must complete four configuration steps.

The dbinit.sde file must contain the database connection and the library path to the ArcSDE and DBMS dynamic libraries.

You must also add additional one-line entries to the /etc/services and the /etc/inetd.conf files, then reinitialize the inetd daemon.

After you have completed the four configuration steps, you can test the remote startup procedure from either a UNIX or a Windows NT® computer using the sdemon command with the "-s" and "-i" options.

Tip

/etc/inetd.conf file entry

This must be a single-line entry with no carriage returns or new lines.

1. Create the $SDEHOME/etc/ dbinit.sde file. This is an example of the variables in the dbinit.sde file.

```
set    ORACLE_HOME=/ultra1/oracle
set    ORACLE_SID=ora
set    LD_LIBRARY_PATH=/usr/lib:/ultra1/
       oracle/lib:/ultra1/oraexe/sdeexe90/lib
unset  TWO_TASK
```

2. As the root user, duplicate the service name in the /etc/ services file as a user datagram protocol (UDP) entry that uses the same port number.

```
# \etc\services

esri_sde      5151/tcp
esri_sde      5151/udp
```

3. Again as the root user, update the /etc/inetd.conf file. Add this line to the bottom of the file.

```
<ArcSDE instance> dgram udp wait <ArcSDE home
owner> <$SDEHOME>/bin/sderemote iomgr_inetd
<$SDEHOME>
```

4. This is an example of a /etc/ inetd.conf file.

```
# SDE remote start-up entries.
esri_sde dgram udp wait sde /ultra1/oraexe/
sdeexe90/bin/sderemote iomgr_inetd /ultra1/
oraexe/sdeexe90
```

5. As the root user, identify the relevant process using the UNIX command "ps -" piped through "grep". Reinitialize the inetd daemon by sending it a signal hang-up or SIGHUP.

   As the ArcSDE administrator, make sure the application server is not started.

```
$ ps -u root | grep inetd
root  112  1 0  Aug 28 ? 0:08 /usr/sbin/inetd -s

$ kill -HUP 112
```

6. From either a UNIX or Windows NT computer, type the sdemon command with the start, server, and service name options to remotely start an ArcSDE application server.

```
$ sdemon -o status
ArcSDE Instance esri_sde Status on ultra at
Thu Aug 28 11:32:56 2003
_____________________________________

ArcSDE instance esri_sde is not available on
ultra.

$ sdemon -o start -p my_password  -s ultra -i
esri_sde
ArcSDE Instance esri_sde started Thu Aug 28
11:35:28 2003
```

Pausing, resuming, and shutting down an ArcSDE application server

The ArcSDE application server has three modes—running, paused, and shutdown. While running mode is the productive state, there will be times when you need to pause the application server or shut it down—for example, to do routine maintenance work.

When the application server is running, client applications can login and access the database.

When the application server is paused, current application connections continue, but additional application requests to connect are denied. This allows current users to complete work before the application server shuts down. You can return a paused application server to running mode by executing the sdemon with the resume option.

When the application server is shut down, the sdemon command notes any ArcSDE processes still running and prompts for confirmation that these tasks should be terminated before continuing with the shutdown. Any user who knows the ArcSDE DBMS ▸

Pausing the ArcSDE application server (Windows and UNIX)

1. To pause an ArcSDE application server, type the sdemon command and specify the pause option.

```
1 $ sdemon -o pause -p my_password
  ArcSDE I/O Manager is paused, no further
  connections will be allowed
```

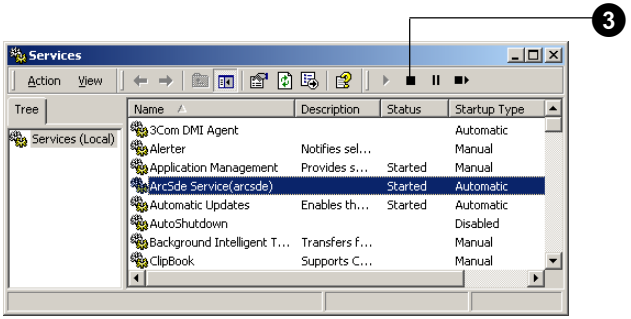
Resuming operation (Windows and UNIX)

1. To resume a paused ArcSDE application server, type the sdemon command and specify the resume option.

```
1 $ sdemon -o resume -p my_password
  ArcSDE I/O Manager is Resuming, new
  connections will now be allowed
```

Shutting down a local Windows ArcSDE application server

1. Click the Start menu, click Settings, click Control Panel, then Administrative Tools.
2. Click Services and scroll through the list of Windows services to find the ArcSDE application server to shut down.
3. Click Stop to shut down the service.



user's password can shut down the application server.

Shutting down the application server relinquishes all of its processes and operating system resources.

If the application server stalls on a Windows machine and it cannot be stopped using the methods discussed, it may be necessary to terminate the giomgr process with the "killp" executable found under %SDEHOME%\tools.

If the ArcSDE application server stalls on a UNIX platform and it cannot be stopped, it may be necessary to terminate the process with the UNIX "kill" command.

Tip

User account permissions

Windows users must have power user or administrator group permissions to pause, resume, or shut down a local or remote ArcSDE application server.

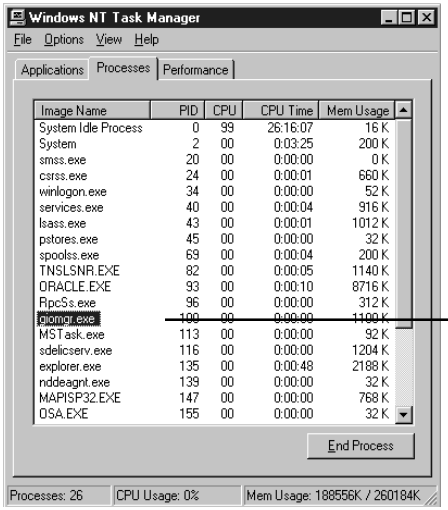
Tip

Remotely pausing and resuming an ArcSDE application server (UNIX and Windows)

You can use the sdemon command to remotely pause, resume, and shut down an ArcSDE application server by including the remote ArcSDE server and service name with the command options—for example, "$ sdemon -o pause -p my_password -s ultra -i esri_sde".

Shutting down a stalled giomgr process on Windows

1. Right-click the Windows taskbar and click Task Manager. Identify the process ID (PID) of the stalled giomgr process.
2. Type "killp" at the MS–DOS command line. Include the PID obtained from the Task Manager window. When prompted, confirm your intent to kill the process by typing "y".



2
```
C:\> cd %SDEHOME%\tools
C:\%SDEHOME%\tools> killp 100
Do you really want to kill process with pid
100(y/n)
y
```

Shutting down a local UNIX ArcSDE application server

1. Type the sdemon command with the shutdown option.
2. Type the sdemon command with the status option to confirm the ArcSDE application server has been shut down.

1
```
$ sdemon -o shutdown -p my_password
ArcSDE I/O Manager is Shutdown
```

2
```
$ sdemon -o status
ArcSDE I/O Manager is not available
```

Shutting down a stalled giomgr process on UNIX

1. Identify the PID of the stalled ArcSDE application server or giomgr process using the UNIX “ps -ef” command that was piped through “grep” to isolate the “giomgr” process.
2. Type the UNIX “kill” command. Include the PID to terminate the giomgr process.

❶
```
$ ps -ef | grep giomgr
arcsde  3403    1  0 06:00:03 ?      0:03
/luke1/sdeexe90/bin/giomgr /luke1/sdeexe90
```

❷
```
$ kill -9 3403
```

Shutting down an ArcSDE application server during the editing of a multiversioned table

Consideration must be given to users before shutting down the ArcSDE application server, especially to those that may be in the midst of an edit session. If you do choose to shut down the application server before users are able to save their changes to a multiversioned table, the following events will occur to ensure that ArcSDE is able to maintain the consistency of the ArcSDE multiversioned database:

1. All unsaved edit sessions are rolled back and lost.
2. Versions being reconciled continue to reference the initial state of the database that existed prior to the start of a reconcile edit session.
3. Posted versions are not affected.
4. Any nonreferenced, orphaned states created as a result of the shutdown are removed the next time the database is compressed.

Removing ArcSDE sessions (Windows and UNIX)

To terminate an ArcSDE user process, list all ArcSDE user processes with sdemon, locate the process identifier, and use the sdemon command with the kill option to remove the process.

Before terminating a user process, be aware that the sdemon command disconnects user processes immediately. If the termination occurs before users issue a commit on a large transaction, the changes are rolled back. This operation should be used as a last resort such as terminating a user process that is no longer responding to the application or when a user abnormally aborts an operation while it was processing a request and the process has hung.

Removing a single ArcSDE user session

1. Locate the PID for the ArcSDE session to be terminated with the sdemon command.
2. Issue the sdemon command with the "-o kill" option and the PID. When prompted, confirm that you wish to kill the process by typing "y".
3. Verify that the process has been terminated.

❶ `$ sdemon -o info -I users` *(list the user processes)*

```
ArcSDE Instance esri_sde Registered Server
Tasks  on luke at Tue Sep 16 11:23:04 2003
_____________________________________________

PID   User  Host:OS         Started
_____________________________________________

90    bob buru:Win32:XDR  Tue Sep 16 09:29:52
10627 bob zanzi:Win32:XDR Tue Sep 16 11:12:31
```

❷ `$ sdemon -o kill -t 10627`

```
Please enter ArcSDE DBA password:

ArcSDE Instance esri_sde Process Management on
luke at Tue Sep 16 11:23:46 2003
_____________________________________________

Kill Server Task 10627?   ARE YOU SURE (Y/N)?:
y
```

❸ `$ sdemon -o info -I users` *(list the user processes)*

```
ArcSDE Instance esri_sde Registered Server
Tasks  on luke at Tue Sep 16 11:25:04 2003
_____________________________________________

PID User  Host:OS        Started
_____________________________________________

90  bob  buru:Win32:XDR Tue Sep 16 09:29:52
$
```

Tip

The sdemon command with the -t option

The -t option specifies the PID of the process to be terminated. It can also be used to terminate all current user processes by typing the "all" keyword in place of a PID.

Tip

Stalled ArcSDE user sessions (UNIX)

To remove stalled ArcSDE user connections or gsrvr processes that cannot be terminated with the "sdemon -o kill" command, you must use the UNIX "kill" command.

Tip

Stalled ArcSDE user sessions (Windows)

The Task Manager cannot be used to terminate gsrvr processes for Windows since the process was started from the admin account. Instead, you must use the killp command under %ARCHOME%\tools.

Removing multiple ArcSDE user sessions

1. Type the sdemon -o command with the kill and -t all options.
2. Verify that all user processes have been terminated.

❶
```
$ sdemon -o kill -t  all
```

❷
```
$ sdemon -o info -I users   (list the user processes)
ArcSDE Instance esri_sde Registered Server
Tasks on luke at Tue Sep 16 11:25:04 2003
_____________________________________________

PID User  Host:OS        Started
_____________________________________________

There are no ArcSDE users logged in.
$
```

Monitoring ArcSDE application servers 5

IN THIS CHAPTER

- **Displaying ArcSDE application server status and lock table information**
- **How ArcGIS uses ArcSDE locking**
- **Displaying ArcSDE application server statistics**
- **Displaying ArcSDE user session information**

To display the status of the ArcSDE application server, use the sdemon command. This command provides a variety of administrative information including:

- The current mode of the ArcSDE service
- The number of clients using the service
- Information about each client/server connection
- Current ArcSDE service configurations

Displaying ArcSDE application server status and lock table information

You can check the status of an ArcSDE application server by using the sdemon command with the status option. You will see the current status of the service—the current connection mode and the number of active server processes—reported onscreen.

```
$ sdemon -o status
ArcSDE Instance esri_sde Status on luke at Tue
Sep 16 10:18:15 2003
   ------------------------------------------------

Server Connection Mode:   Accepting Connections

Active Server Processes:   57

$
```

The ArcSDE locking mechanisms manage concurrent user access and guarantee read consistency when a user queries the database.

Use “sdemon -o info -I locks” to keep track of the number of locks.

```
$ sdemon -o info -I locks
ArcSDE Instance esri_sde Lock Table Information
on luke at Thu Sep 16 10:13:48 2003
------------------------------------------------

1 SDE Id: 1017,Map Layer: 3,Lock Type: Shared
Area

2 SDE Id: 1017,Map Layer: 2,Lock Type: Update
Layer
------------------------------------------------

2 of -1 ArcSDE Layer Lock(s) currently in use.



1  SDE Id: 1017, Object Id:  1, Object Type: 1
Application: [ArcSDE Internal]  Lock Type: Shared
Object                                          2
SDE Id: 1017, Object Id:  3, Object Type: 2
Application: [User]  Lock Type: Exclusive Object

3  SDE Id: 1017, Object Id:  2, Object Type: 1
Application: [User]  Lock Type: Shared Object
------------------------------------------------

3 of -1 ArcSDE Object Lock(s) currently in use.

1   SDE Id: 28129, State: 25685, Lock Type: Auto
Exclusive State
2   SDE Id: 28129, State: 25684, Lock Type: Auto
Exclusive State
3   SDE Id: 28129, State: 25683, Lock Type: Auto
Exclusive State
4   SDE Id: 28129, State: 25685, Lock Type:
Shared State
5   SDE Id: 28129, State: 25684, Lock Type:
Shared   State
6  SDE Id: 28129, State: 25683, Lock Type: Shared
State

------------------------------------------------

6 of  -1 ArcSDE State Lock(s) currently in use.

1  SDE Id: 90, table: 162, Lock Type: Shared
Table

2  SDE Id: 26732, table: 122, Lock Type: Shared
Table

3  SDE Id: 28129, table: 162, Lock Type: Shared
Table

------------------------------------------------

3 of  -1 ArcSDE Table Lock(s) currently in use.

$
```

The following lock table data appears for each lock in use:

SDE Id	SDE Identifier that owns the lock
MAP LAYER	Map layer number to which the lock applies
LOCK TYPE	The type of lock (update/ shared,layer/area, auto)

If the lock type is an area lock or an automatic lock on a feature, the locked area also appears.

ArcSDE provides applications with four kinds of locks:

Object locks—used for versioning and geodatabase activities

Table locks—used to lock tables

Area locks—used to lock a spatial extent of a feature class

State locks—used to lock a versioned state of a feature class or table

How ArcGIS uses ArcSDE locking

Whenever an ArcGIS application connects to an ArcSDE application server, it obtains the default state of the version it is connecting to and requests a shared lock on the state. It acquires the shared lock on the state if no other session currently holds an exclusive lock on all states. Sessions release their shared state locks when they disconnect.

When an edit occurs, the session acquires an exclusive lock for each new state created.

A session that attempts to perform a compress operation first collects a list of candidate states that can be deleted. A state can be deleted if another session does not have it locked and it does not have a version assigned to it. An exclusive lock is acquired on each state before it is deleted.

When an ArcGIS application starts to edit a version, a shared object lock is acquired. The object lock is released when editing is stopped.

When an ArcGIS session references an object class, it acquires a shared table lock on the business table of that object class. For example, if the session references a feature class, a table lock is acquired on the business table.

Before an ArcGIS session can change the properties of the object class or the schema of any of the tables of the object class, an exclusive table lock must be acquired. The exclusive lock cannot be acquired if other sessions have acquired shared locks on the tables.

When an ArcGIS session reconciles a version, the shared object locks are promoted to exclusive locks. The promotion and, therefore, the reconciliation will raise an error if other sessions are editing the object.

ArcGIS does not acquire layer locks, row locks, or area locks while editing.

Displaying ArcSDE application server statistics

You can use the sdemon command with the stats option to display statistical information about each current ArcSDE application server using the "-o info -I stats" option. The stats option displays the information stored in the SDE.PROCESS_INFORMATION table. ArcSDE will only store information in this table when the PROCSTATS server configuration parameter has been set a value greater than 0. For more information on this parameter, consult either Chapter 3, 'Configuring ArcSDE application servers' or Appendix D, 'ArcSDE initialization parameters'.

```
$ sdemon -o info -I stats
ArcSDE Instance gis Server Process Statistics
on luke at Tue Sep 16 10:37:14 2003
-----------------------------------------------

S-ID  OPS   READS   WRITES   BUFFERS   PARTIAL
----- ----- ------- -------- --------- ---------
12566 111   1278    0        12        0

F/BUF   BUF AVG   TOT Kbytes
------- --------- ----------
106          6K       80K
```

The output includes:

S-ID	SDE session identifier
OPS	Number of client/server operations
READS	Number of features/identifiers read from disk
WRITES	Number of features written to disk
BUFFERS	Total number of buffers sent to client task
PARTIAL	Number of features sent to client that were larger than the buffer size
F/BUF	Average number of features/identifiers per buffer
BUF AVG	Average buffer size in bytes
TOT Kbytes	Total kilobytes of data sent to client

If no processes are connected to the ArcSDE application server or if the PROCSTATS parameter is not set to a value greater than 0, a message to that effect will appear.

```
$ sdemon -o info -I stats
ArcSDE Instance gis Server Process Statistics on
luke at Tue Sep 16 10:37:14 2003
_____________________________________________

There are no ArcSDE users logged in.
$
```

Displaying ArcSDE user session information

You can list all the server process information relating to current user connections using sdemon with the "-o info -I users" option.

In this case sdemon reads records from the SDE.PROCESS_INFORMATION table. If the PROCSTATS server configuration parameter is set to a -1 value (the default) session information is not written to this table. Set PROCSTATS to the minimum threshold interval that you want ArcSDE to update the SDE.PROCESS_INFORMATION table with session changes.

For more information on the PROCSTATS parameter, refer to Chapter 3, 'Configuring ArcSDE application servers' or Appendix D, 'ArcSDE initialization parameters'.

```
$ sdemon -o info -I users
ArcSDE Instance esri_sde Registered Server Tasks
on luke at Tue Sep 16 11:10:51 2003
_____________________________________________
S-ID   User  Host:OS         Started
_____________________________________________
90     bob    buru:Win32:XDR  Tue Sep 16 09:29:52
2003
9526   bob    zanzi:Win32:XDR Tue Sep 16 11:02:46
2003
10406vtest    kayak:Win32:XDR Tue Sep 16 11:10:21
2003
$
```

The following process information appears for each registered ArcSDE client:

```
S-ID     Process identifier
USER     User name of client
HOST:OS Name of the host computer and operating
        system
STARTED Name and time the process started
```

When a client terminates its ArcSDE connection, all process statistics are written to the SDEHOME/etc/giomgr_<service>.log file. The output from that file would look similar to the following:

```
Tue Sep 16 10:24:26 2003 - SDE Server Pid 5429
Stopped, User: sdetest.
Tue Sep 16 10:23:30 2003 - SDE Server Pid 5429
Registered, User: sdetest.
Tue Sep 16 10:24:26 2003 - SDE 9526, Process
5429, R/T Calls 22, Features read 0, wrote 3551,
Locks 0, Buffers 5, Partial 0 , Buffered Features
3551, Buffered Data 3320K
Tue Sep 16 10:24:26 2003 - SDE Server 5429 exit'd
with status 0
```

Troubleshooting the ArcSDE application server 6

IN THIS CHAPTER

- What happens when you start an ArcSDE application server
- What happens when an ArcSDE client connects to an application server
- What happens when an ArcSDE client direct connects to the DBMS
- Common ArcSDE startup problems on UNIX servers
- Common ArcSDE startup problems on Windows servers
- Setting the Windows SharedSection
- The Windows Event Viewer
- Examining the ArcSDE error logfiles
- ArcSDE intercept and tracing
- ArcSDE tracing

Most problems associated with starting an ArcSDE application server occur because of a problem with the system environment. Often, a critical step was missed during the installation or configuration of the software.

This chapter provides a walk-through of common troubleshooting tasks as well as solutions to common problems encountered during the startup and connection to an ArcSDE application server.

This section describes the startup of an ArcSDE application, the problems that may occur, and their probable causes.

The ArcSDE application starts the giomgr process

The giomgr executable file must be accessible. On UNIX systems, make sure that $SDEHOME\bin is in the system path and $SDEHOME\lib is in the system library path. On Windows, %SDEHOME%\bin must be in the system path if the sdemon command is used to start the service. Although on Windows the service is normally started from the services menu, sometimes it is appropriate to use the sdemon command to debug a failed startup.

The giomgr reads the system environment variables from the dbinit.sde file

The SDEHOME\dbinit.sde file contains settings for system environment variables that override those set in the system environment for either UNIX or Windows systems. On UNIX systems, if the dbinit.sde file does not exist, the sdemon command displays a warning message during startup. For more information, see 'The dbinit.sde file format' in Chapter 3.

On UNIX systems (and on Windows systems if the application server is started from an MS–DOS command prompt using the sdemon command), the contents of the dbinit.sde file can be displayed during startup by setting the SDEDBECHO system environment variable to TRUE.

Be sure the DBMS connection variables and the license manager variables are set correctly.

You may list the current ArcSDE environment variables by using the sdemon command with "-o info -I vars" options.

```
$ sdemon -o info -I vars
ArcSDE Instance esri_sde's environment variables
on nendrum at Mon Oct 20 10:42:48 2003
-----------------------------------------------
ARCHOME=C:\gis\arcexe90
ARCHOME_USER=C:\Program Files\ESRI\ArcInfo
ARCINFOFONTNAME=Courier New
ARCINFOFONTSIZE=8   ATHOME=C:\gis\arctools
COMPUTERNAME=NENDRUM
ComSpec=C:\WINNT\system32\cmd.exe
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Os2LibPath=C:\WINNT\system32\os2\dll;
Path=d:\oracle\bin;C:\Program Files\Oracle\jre\
1.1.7\bin;C:\WINNT\system32;C:\WINNT;C:\Program
Files\ESRI\ArcInfo\bin;D:\ESRI\ArcInfo\arcsde\
bin;D:\ESRI\ArcInfo\arcsde\lib
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 3
Stepping 4, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=0304
SDEHOME=D:\ESRI\ArcInfo\arcsde
SystemDrive=C:
SystemRoot=C:\WINNT
USERPROFILE=C:\WINNT\Profiles\Default User
windir=C:\WINNT
SDENOEQUIV=true
$
```

The giomgr determines if an ArcSDE server license has been installed

Be sure the ArcSDE license has been installed. For more information, refer to the sdesetup command reference in the ArcSDE online help.

The giomgr reads the Transmission Control Protocol/Internet Protocol service name

On Windows platforms, the service name is read from the registry if the ArcSDE application server started from the Services menu. However, when debugging a startup problem by using the "sdemon -o start" command, the service name is read from the services.sde file.

On UNIX platforms, the service name is always read from the $SDEHOME\etc\services.sde file.

The giomgr attaches to the TCP/IP port assigned to the service name

The service name must be in the system's services file. On Windows, the ArcSDE installation program adds the service name to the system service file automatically. On UNIX platforms, the services file must be updated manually.

On Windows, the system service file is located at <drive>:\winnt\system32\drivers\etc\services, while on UNIX systems the file is located at /etc/services.

The giomgr connects to the DBMS using connection information from dbinit.sde and operating system environment variables

Make sure the DBMS is up and running and that the sde user can connect to the database. The DBMS connection parameters set in the dbinit.sde file have precedence over any parameters that are set in the system environment.

If the dbinit.sde file does not exist, a warning message is sent to standard output; however, this will not prevent the ArcSDE application server from starting since the DBMS connection parameters of the system environment are used.

The giomgr flushes the lock tables

The giomgr process flushes the locks of any process information orphaned by a chaotic shutdown of the application server. The locks of direct connections are not flushed when the application server starts.

The giomgr listens for connections on its TCP/IP port

At this point, the ArcSDE application server has started. Applications can now connect and use the application server.

What happens when an ArcSDE client connects to an application server

This section describes the sequence of events that take place when an ArcSDE client application connects to an ArcSDE application server.

The giomgr process listens for connections on its TCP/IP port

The giomgr must be in a listening state before it can process a connection request. Make sure the ArcSDE application server is started and listening.

On UNIX, use "sdemon -o status" to determine the state of the giomgr process.

On Windows, examine the state of the ArcSDE application server from the Services menu. Click Start, point to Settings, then click Control Panel and Application Tools. Double-click the Services icon to invoke the Services menu. Scroll down until you find the ArcSDE application server. The ArcSDE application server should have a status of 'STARTED' under the status field.

On Windows you can also use the "sdemon -o status" command from an MS–DOS command tool.

Applications submit connection requests to the ArcSDE application server

The giomgr process responds to connection requests serially. Depending on the underlying DBMS, the giomgr process may require anywhere from 1 to 5 seconds to validate a connection request. It's possible that if many applications are trying to obtain an ArcSDE connection at the same time, some may exceed the standard TCP/IP 75-second time out. This may be prevented from happening by setting the SDEATTEMPTS environment. SDEATTEMPTS specifies the number of times an application should retry the connection.

The giomgr compares the application computer's clock time with its host's clock time

If the application computer's clock time is more than MAXTIMEDIFF seconds from the ArcSDE server's clock time, the giomgr process does not allow the application to connect. MAXTIMEDIFF is set in the SDE.SERVER_CONFIG file. For more information see Chapter 3,'Configuring ArcSDE application servers', and Appendix D, 'ArcSDE initialization parameters'.

The giomgr compares the application's client ArcSDE release with the ArcSDE application server's release

If the application's release is older than the application server's release, the connection is refused.

ArcSDE applications are downward compatible. Applications developed with the ArcSDE 9 application programming interface can connect to releases of ArcSDE 9 and lower. Applications built with an earlier version of the ArcSDE API cannot connect to ArcSDE 9 application servers. Check the application's documentation for supported ArcSDE releases.

The giomgr process starts a gsrvr process that will serve the application

The giomgr process must be able to spawn a gsrvr process. If the maximum number of processes determined by current operating system restrictions has been reached, this operation will fail and no gsrvr process will be created.

The gsrvr process attaches to shared memory

Sufficient memory must be available on the ArcSDE application server's host computer. Otherwise, the application connection will fail with a shared memory error. Should this happen, make

more memory available to the gsrvr processes by reconfiguring either the ArcSDE application server or the DBMS server to use less memory. If possible, add more physical memory to the host computer.

The gsrvr process connects to the DBMS

The application must provide a valid username, password, and database name (optional for some DBMSs) when it submits the connection request to the giomgr process. Invalid entries are rejected with a "-9 SE_INVALID_USER" error.

The giomgr process attaches the application to the gsrvr process

Once the giomgr process has attached the application to the gsrvr process, it resumes listening for new connections and performing other ArcSDE application server management tasks. All application communication with the DBMS is conducted through the gsrvr process.

What happens when an ArcSDE client direct connects to the DBMS

This section describes the sequence of events that occur when an ArcSDE client application connects directly to a DBMS server.

The DBMS server listens for local or remote connections

Each of the DBMSs supported by ArcSDE has its own method of accepting the connections of client applications. Make sure you are entering the connection information correctly if you are using an ESRI application. For information on how to perform a direct connection, consult the appendix titled 'Making a Direct Connection' in the *ArcSDE Configuration and Tuning Guide for <DBMS>* PDF file, located in the documentation folder on the ArcSDE CD.

If you still cannot get connected, test the connection using the DBMS' native SQL utility.

Make sure the ArcSDE server license has been installed.

If the client application queries a valid ArcSDE server license (stored in the SDE.SERVER_CONFIG table), it acquires a connection to the database.

Common ArcSDE startup problems on UNIX servers

The following section lists some of the ArcSDE service startup problems that are likely to be encountered in a UNIX environment.

System path variable issues

- If the PATH environment variable does not include the $SDEHOME/bin directory, the following error message is reported:

  ```
  sdemon: Command not found
  ```

- If the library path environment variable does not include the $SDEHOME/lib directory, the following error message is reported:

  ```
  ld.so.1: sdemon: fatal: libsde90.so: open
  failed: No such file or directory
  Killed
  ```

- If the library path environment does not include the necessary DBMS library directory, an error message similar to the following is reported:

  ```
  ld.so.1: /ultra1/ora9iexe/bin/giomgr: fatal:
  libclntsh.so.9.1: open failed: No such file
  or directory
  Killed
  Could not start ArcSDE – Check Network,
  $SDEHOME disk, DBMS settings and dbinit.sde.
  ```

  For more information on how to set the library environment variable for your ArcSDE product, see the *ArcSDE Installation Guide.*

ArcSDE service already started

- If the I/O manager is already running, the following message appears:

  ```
  SDE Already Running
  ```

ArcSDE server license has not been installed

If the ArcSDE service license has not been installed, the application server will not start. You must install the licence using the keymanager administration command. Contact ESRI Customer Support to obtain a valid license.

Temporary file permission problems

- If any ArcSDE temporary files exist and they are not owned by the ArcSDE administrator, the following error message is returned:

  ```
  ERROR: Cannot Initialize Shared Memory (-79)
  Delete /tmp/<service name> and /tmp<service
  name>.lock if present.
  Could not start ArcSDE – Check Network,
  $SDEHOME disk, DBMS settings and dbinit.sde.
  ```

  Delete the temporary files /tmp/<service name> and /tmp/<service name>.lock. For example, if the service name is esri_sde, you would delete the files /tmp/esri_sde and /tmp/esri_sde.lock. You may have to log in as the root user to delete these files.

Files have been deleted from /tmp

If after the ArcSDE application server has been started, the files stored under the /tmp directory are deleted, the ArcSDE application server will fail when a user either connects or disconnects. The application server relies on UNIX socket protocol files created under the /tmp directory. As a rule you should not delete the files under the /tmp directory. However, if you absolutely must, you should shut down the ArcSDE application server before doing so.

Problems relating to the DBMS

- If the DBMS is not started, you will receive an error message similar to the following:

  ```
  init_DB DB_instance_open_as_dba: -51
  DBMS error code: 1034
  ORA-01034: ORACLE not available

  Could not start ArcSDE – Check Network,
  $SDEHOME disk, DBMS settings, and dbinit.sde.
  ```

- If the sde DBMS user password is not correct, you will receive an error message similar to the following:

  ```
  init_DB DB_instance_open_as_dba: -93
  DBMS error code: 1017
  ORA-01017: invalid username/password; login
  denied

  Could not start ArcSDE – Check Network,
  $SDEHOME disk, DBMS settings, and dbinit.sde.
  ```

- If the sde DBMS user does not exist, you will receive an error message similar to the following:

  ```
  init_DB DB_instance_open_as_dba: -93
  DBMS error code: 1017
  ORA-01017: invalid username/password; login
  denied

  Could not start ArcSDE – Check Network,
  $SDEHOME disk, DBMS settings, and dbinit.sde.
  ```

The SE_OUT_OF_MUTEXES (-109) error on a Solaris server

The Solaris™ operating system uses files to implement the POSIX shared semaphores that ArcSDE uses. If these files get left behind after an operating system failure or power outage, they can sometimes cause problems.

The location of these files is controlled by the Solaris operating system. You will find them under either the /tmp or the /var/tmp directories as follows:

```
/tmp/.SEMD/
SDE_9.0_<instance>_iomgr_shared_semaphore
/tmp/.SEML/
SDE_9.0_<instance>_iomgr_shared_semaphore
```

or

```
/var/tmp/.SEMD/
SDE_9.0_<instance>_iomgr_shared_semaphore
/var/tmp/.SEML/
SDE_9.0_<instance>_iomgr_shared_semaphore
```

Following an operating system failure if you fail to start the ArcSDE application server and receive a -109 error, it is probably because the two shared semaphore files exist. If you find either of these files in either the /tmp or /var/tmp locations, delete them and try to start the ArcSDE application server again.

Common ArcSDE startup problems on Windows servers

Normally, the ArcSDE application server is started as a Windows service from the Services control panel. If an error appears after clicking the Start button for the service, try to determine the nature of the problem. The error message contains an error number.

The error number usually relates to a specific type of error. Listed below are errors often encountered when starting an ArcSDE application server. This list includes the error numbers and their likely causes.

1068 Dependency failure

The DBMS to which the ArcSDE application server is trying to connect could not be found. The most likely causes of this problem are:

- The DBMS service is not started.
- The DBMS server has been removed.
- The DBMS connection information, entered when the ArcSDE application server was created, is incorrect.

Make sure the DBMS service is started and independently confirm that this is not the source of the problem. If the error persists, use the sdeservice command to delete the existing ArcSDE application server and re-create it. For more information, see the discussion of the sdeservice command in Appendix C, 'ArcSDE application server command references'.

1069 Login failure

Generally, this error implies that the Windows user who started the ArcSDE application server is neither a Windows administrator nor a Windows power user. An incorrect password is another possibility.

If the system administrator account is not being used to start the service, make sure the user account is a member of the administrator or power user group.

1072 Registry was busy

Something is happening in the registry regarding the ArcSDE application server entry. Perhaps "sdeservice -o delete" was run, or the service has been opened with the registry editor, regedt32. Alternatively, there may have been a problem with the Object Linking and Embedding DataBase (OLE DB) provider. Consult the installation guide for the correct version for the OLE DB provider.

1075 Service dependency deleted

The ArcSDE application server is unable to locate the DBMS service that it will connect to.

Make sure the DBMS service exists and is started. If the problem persists, use the sdeservice command to delete and re-create the ArcSDE application server. For more information, see Appendix C, 'ArcSDE application server command references'.

2140 Internal Windows error

The ArcSDE application server wasn't able to complete the startup process. Examine the sde error logfile %SDEHOME%\etc\sde_<sde_instance>.log for possible clues as to why the ArcSDE service will not start.

Possible causes include:

- Can't connect to the DBMS server
- Can't create the ArcSDE data dictionary
- Can't query a valid ArcSDE server license from the SDE.SERVER_CONFIG table.

Possible solutions include:

- If the ArcSDE user's password was entered incorrectly, use sdeservice -o modify -r SDE_DBA_PASSWORD to correct it.
- If the DBMS connection information is incorrect, edit the %SDEHOME%\etc\dbinit.sde file.
- If an ArcSDE server license cannot be queried, make sure a valid license has been installed.

Setting the Windows SharedSection

If an error message window entitled “gsrvr.exe - DLL Initialization Failed” is displayed on the ArcSDE application server stating that “Initialization of the dynamic link library WINNT\system32\COMCTL32.dll failed. The process is terminating abnormally”, you need to increase the Windows SharedSection for noninteractive desktops.

On Windows, the ArcSDE service is started as a noninteractive desktop. The maximum amount of heap memory allocated to noninteractive desktops is limited by a Windows initialization parameter called SharedSection. This parameter is altered using the Windows registry editor.

To change the SharedSection, click Start, then Run. Enter regedit in the input line, then click OK. Navigate to the following registry path and double-click the Windows registry.

\\HKEY_LOCAL_MACHINE\
SYSTEM\CurrentControlSet\
Control\Session Manager\
SubSystems\Windows

This string contains startup parameters for Windows. Within the string you will find the SharedSection parameter.

The default value is 1024,3072,512. The third argument is the maximum amount of heap memory allocated to noninteractive desktops. By default it is set to 512 kilobytes or one megabyte. At this setting, an ArcSDE application server will accept approximately 56 connections. Increasing the maximum heap size of noninteractive desktops to two megabytes allows the ArcSDE application server to accept up to 270 connections. The ShareSection value would be 1024,3072,2048 when the noninteractive heap size is set to two megabytes.

The maximum amount of heap memory available for all desktops both interactive and noninteractive is 48 megabytes. Since the amount of memory is finite, you should take care in adjusting the SharedSection parameter.

The Windows server must be rebooted to accept a new ShareSection value.

For more information, consult the "Kernel32.dll initialization failure" or "User32.dll initialization failure" in the Microsoft Development Network (MSDN).

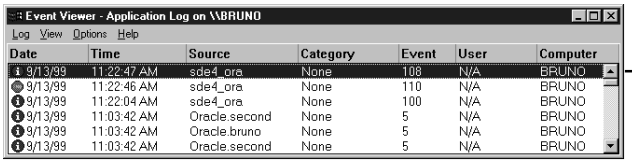
The Windows Event Viewer

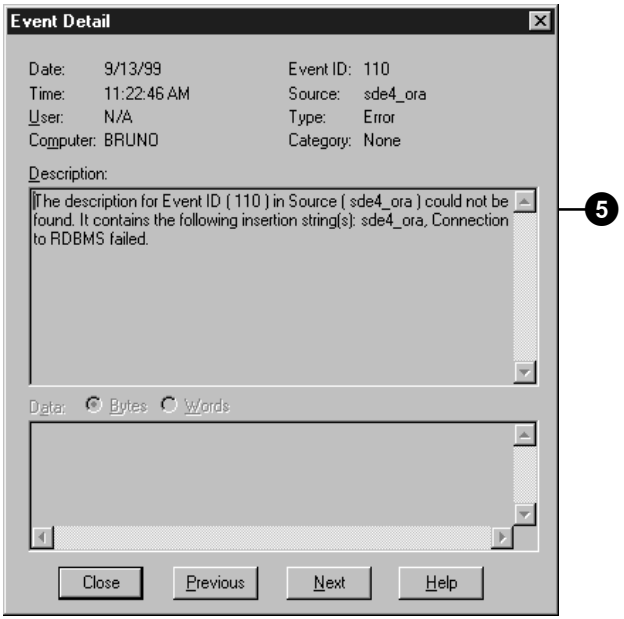
The Windows Event Viewer provides diagnostic information that may also help explain ArcSDE startup problems.

Although the Event Viewer will often include a description of the problem, you can also check the %SDEHOME%\etc\sde_<service>.log file. This file will contain further information relating to the Windows startup problems.

Using the Event Viewer

1. Click the Start menu, click Settings, and click Control Panel.
2. From the Control Panel menu, click Administrative Tools. From the Administrative Tools menu, click Component Services.
3. From the Component Services menu, expand the Event Viewer folder and select the Application option.
4. Look for a red stop sign icon in the Type column and the corresponding name of the ArcSDE service in the Source column. Double-click the ArcSDE service entry to bring up the Event Detail window.
5. The Event Detail window includes a description of the problem. In this example, it is clear that a connection to the DBMS failed.





Examining the ArcSDE error logfiles

ArcSDE and all supported DBMSs track their activities by writing messages of events to ASCII logfiles. The logfiles may be examined to trace errors that have occurred. ArcSDE writes to two log files: giomgr_<service>.log file and sde_<service>.log file. (If you are using a direct connection, the application writes error messages to SDEHOME\etc\sdedc_<dbms>.log instead of the application server's sde_<service>.log file.)

Viewing the giomgr_<service>.log file

The giomgr.log file is an ASCII file that contains an entry for all giomgr process activities. Each time a user connects or attempts to connect to the ArcSDE application server, a message is logged. When the user disconnects, another message is logged. The giomgr.log file also captures the startup and shutdown procedures of the ArcSDE application server.

Viewing the sde_<service>.log file

Whenever a gsrvr process encounters a problem, the ArcSDE service records an entry in the sde_<service>.log. (<service> is the name of the application server's service.) Sometimes the messages are warnings, while other times they point to ArcSDE application server errors that you must address. When examining the sde_<service>.log file, keep in mind that the messages will be written to this file when errors occur in the ArcSDE application server process. Sometimes an ArcSDE application will report an ArcSDE-related problem, but this event will not appear in the sde_<service>.log. That is because the error has occurred on the ArcSDE client-side and not the server-side.

The sde_<service>.log is truncated each time the ArcSDE application service is started.

DBMS error logfiles

Each DBMS has its own way of logging errors. Consult the relevant DBMS administration guide to determine how your DBMS logs errors.

ArcSDE intercept

If you need to contact ESRI technical support, the analyst may ask you to intercept the ArcSDE client or server network broadcasts, depending on the nature of the problem. The ArcSDE intercept facility captures information that the client or server sends across the TCP/IP port to a file for examination.

To intercept ArcSDE server broadcasts, set the relevant variables in the dbinit.sde file (see details below) and restart the ArcSDE application server. The dbinit.sde file is located in the $SDEHOME/etc directory on UNIX systems and in the %SDEHOME%\etc directory on Windows systems.

To stop intercepting ArcSDE server network broadcasts, either comment out the variables by preceding the entry with the pound sign character "#", or delete them from the dbinit.sde file and restart the ArcSDE application server.

To intercept ArcSDE client network broadcasts, set the variables in the client application user's system environment before connecting to the ArcSDE application server. To stop intercepting ArcSDE client network broadcasts, disconnect the application from the ArcSDE application server, unset the variables, then reconnect to the ArcSDE application server.

You may set the SDEINTERCEPT variable with the following flags to intercept network broadcasts:

c—intercept the API command name

r—intercept the Channel broadcasts read-only

w—intercept the Channel broadcasts write-only

t—intercept log time (minute:second)

T—intercept log time (hour:minute:second)

f—intercept flush immediate

For both client and server intercepts, set the SDEINTERCEPTLOC variable to the full pathname of the filename prefix that receives the information. Information is intercepted on a per-session basis. When intercept is enabled, a new file is created and written to each time an application connects to the ArcSDE application server. The file is closed only after the application disconnects. ArcSDE generates a filename from the prefix provided in SDEINTERCEPTLOC by appending a numeric extension that begins at .001 and that increments sequentially for each new file created.

Gathering both the client and server network broadcast often helps the technical support analyst diagnose problems unique to either the client or server, since the broadcasts should be symmetric. Any asymmetic broadcasts would indicate information that is not being received on one end.

If the technical support analyst asks for intercept output from both the client and server, use distinct prefix names to distinguish between the client and server. For example, setting SDEINTERCEPTLOC to d:\tmp\sde_server in the dbinit.sde file captures server network broadcasts. Setting SDEINTERCEPTLOC to d:\tmp\sde_client in the applications environment captures client network broadcasts in the same directory but with a different prefix.

This is an example of the environment variables required to intercept server broadcasts from an ArcSDE application server installed on Windows. These variables would be set in the %SDEHOME%\etc\dbinit.sde file but would not take effect until the ArcSDE application server is restarted.

```
set SDEINTERCEPT=crwtf
set SDEINTERCEPTLOC=D:\tmp\sde_server
```

Following the ArcSDE application server restart, subsequent applications that connect to the ArcSDE application server will each create a file in the D:\tmp directory with the prefix sde_server. These files will contain the server broadcasts generated during each application's ArcSDE session.

ArcSDE tracing

Application programmers building programs with the ArcSDE C API may want to trace the ArcSDE function calls. Programmers can trace the execution of their ArcSDE programs by turning on ArcSDE tracing.

ArcSDE tracing can be turned on by either setting the system environment variable SDETRACELOC, or by calling the SE_trace_on function within the application program.

Calling the SE_trace_on function from within the application program requires the programmer to recompile the program. If it is not desirable to do so, the SDETRACELOC variable can be set instead. However, the SDETRACELOC must be set before the ArcSDE session connects, and the entire session is traced, possibly creating a large trace file. Calling the SE_trace_on and SE_trace_off functions, tracing can be turned on and off at intervals, allowing the programmer to generate trace output of segments of ArcSDE functionality.

Refer to the ArcSDE Online Developer Help for a discussion of the SE_trace_on and SE_trace_off functions.

The SDETRACELOC environment variable must be set prior to connecting to ArcSDE to turn ArcSDE tracing on. Tracing works for either connections to the ArcSDE application server or directly to the DBMS server.

Set the SDETRACELOC to the pathname of the file you want to write the tracing output to. Do not give the file an extension. ArcSDE creates a trace file and appends an extension that is a three-digit number each time an application connects to ArcSDE. For example, for an SDETRACELOC variable set as follows:

```
set SDETRACELOC=D:\temp\sdetrace
```

The following files would be created in the D:\temp folder for three different sessions that have connected to ArcSDE:

```
sdetrace.001 sdetrace.002 sdetrace.003
```

The SDETRACEMODE variable contains a string of flags that indicate what is to be included in the trace. The flags are as follows:

b—brief mode: Prints function names only

v—verbose mode: Prints function names, input, output, and return values

m—minute mode: Prints the time in [minute:second] format

h—hour mode: Prints the time in [hour:minute:second] format

f—force mode: Forces data to be written to the trace file

If the SDETRACEMODE environment variable is not set, the default mode is vhf (verbose, hour, and force). If both verbose mode and brief mode are set, verbose mode is applied. If both minute mode and hour mode are set, hour mode is applied. If SDETRACEMODE contains an invalid parameter, brief mode is applied.

ArcSDE data dictionary

A

IN THIS APPENDIX

- **ArcSDE system tables**
- **Geodatabase system tables**

The ArcSDE data dictionary includes tables that maintain information about the feature classes and feature datasets. The feature class and feature dataset spatial references, states, and versions are also maintained within the ArcSDE data dictionary. The ArcSDE data dictionary combines both the ArcSDE system tables and the geodatabase system tables, created by the sdesetup functionality.

ArcSDE creates and maintains these tables and views. They were not designed to be accessed by external programs. Changing the data within the ArcSDE data dictionary, either by an external program or manually, is not supported.

The intent of describing the structure of the data dictionary is to provide an inventory of database objects that must be backed up for later restoration in the event of system failure.

For each ArcSDE application server, the ArcSDE software creates several system tables within the ArcSDE user's schema to manage spatial data.

The ArcSDE for SQL Server product prefixes each of these tables with SDE_.

VERSION table

The VERSION table maintains information about the ArcSDE version with which the database expects to operate. The table contains the specific release identification for the most recent version of ArcSDE that executed a version update. The ArcSDE giomgr process checks this table to ensure proper version compatibility.

The VERSION table and other ArcSDE system tables are updated by the sdesetup<DBMS> program after a new version of ArcSDE is installed.

VERSION

Name	Data_Type	Null?
major	SE_INTEGER	NOT NULL
minor	SE_INTEGER	NOT NULL
bugfix	SE_INTEGER	NOT NULL
description	SE_STRING_TYPE(96)	NOT NULL
release	SE_INTEGER	NOT NULL

LAYERS table

The LAYERS table maintains data about each feature class in the database. The information helps build and maintain spatial indexes, ensure proper shape types, maintain data integrity, and store the spatial reference for the coordinate data.

It includes the following information:

- Owner, table, and column name for the shape column
- Name of the table containing the actual shape data
- Spatial index grid cell sizes
- Envelope (minx, miny, maxx, maxy)
- ArcSDE software-assigned layer ID for internal use
- Feature class description
- Statistical information on the shapes in the feature class for data transfer buffer configurations

LAYERS

Name	Data_Type	Null?
layer_id	SE_INTEGER_TYPE	NOT NULL
description	SE_STRING_TYPE(65)	NULL
database_name	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
table_name	SE_STRING_TYPE(160)	NOT NULL
spatial_column	SE_STRING_TYPE(32)	NOT NULL
eflags	SE_INTEGER_TYPE	NOT NULL
layer_mask	SE_INTEGER_TYPE	NOT NULL
gsize1	SE_FLOAT64_TYPE	NOT NULL
gsize2	SE_FLOAT64_TYPE	NOT NULL
gsize3	SE_FLOAT64_TYPE	NOT NULL

LAYERS (continued)

Name	Data_Type	Null?
minx	SE_FLOAT64_TYPE	NULL
miny	SE_FLOAT64_TYPE	NULL
maxx	SE_FLOAT64_TYPE	NULL
maxy	SE_FLOAT64_TYPE	NULL
cdate	SE_INTEGER_TYPE	NOT NULL
layer_config	SE_STRING_TYPE(32)	NULL
optimal_array_size	SE_INTEGER_TYPE	NULL
stats_date	SE_INTEGER_TYPE	NULL
minimum_id	SE_INTEGER_TYPE	NULL
srid	SE_INTEGER_TYPE	NOT NULL
base_layer_ID	SE_INTEGER_TYPE	NOT NULL
minz	SE_FLOAT64_TYPE	NULL
minm	SE_FLOAT64_TYPE	NULL
maxz	SE_FLOAT64_TYPE	NULL
maxm	SE_FLOAT64_TYPE	NULL

GEOMETRY_COLUMNS table

The GEOMETRY_COLUMNS table contains the feature class names, their geometry's storage type, and coordinate dimension.

GEOMETRY_COLUMNS

Name	Data_Type	Null?
F_table_catalog	SE_STRING_TYPE(32)	NULL
F_table_schema	SE_STRING_TYPE(32)	NOT NULL
F_table_name	SE_STRING_TYPE(160)	NOT NULL
F_geometry_column	SE_STRING_TYPE(32)	NOT NULL
G_table_catalog	SE_STRING_TYPE(32)	NULL
G_table_schema	SE_STRING_TYPE(32)	NOT NULL
G_table_name	SE_STRING_TYPE(160)	NOT NULL
storage_type	SE_INTEGER_TYPE	NULL
geometry_type	SE_INTEGER_TYPE	NULL
coordinate_dimension	SE_INTEGER_TYPE	NULL
max_ppr	SE_INTEGER_TYPE	NULL
srid	SE_INTEGER_TYPE	NOT NULL

SDE_XML_ COLUMNS table

The XML_COLUMNS table contains a list of XML columns stored in the database. The table registration ID and XML column ID, XML column name, XML index_id, configuration keyword, and minimum ID are included.

SDE_XML_COLUMNS

Name	Data_Type	Null?
column_id	SE_INTEGER_TYPE	NOT NULL
registration_id	SE_INTEGER_TYPE	NOT NULL
column_name	SE_STRING_TYPE(32)	NOT NULL
index_id	SE_INTEGER_TYPE	NULL
minimum_id	SE_INTEGER_TYPE	NULL
config_keyword	SE_STRING_TYPE(32)	NULL

SDE_XML_ INDEXES table

The SDE_XML_INDEXES table contains a list of XML indexes stored in the database.

SDE_XML_INDEXES

Name	Data_Type	Null?
index_id	SE_INTEGER_TYPE	NOT NULL
index_name	SE_STRING_TYPE(32)	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
index_type	SE_INTEGER_TYPE	NOT NULL
description	SE_STRING_TYPE(64)	NULL

SDE_XML_ INDEX_TAGS table

The SDE_XML_INDEX_TAGS table contains a list of XML index tags stored in the database.

SDE_XML_INDEX_TAGS

Name	Data_Type	Null?
index_id	SE_INTEGER_TYPE	NOT NULL
tag_id	SE_INTEGER_TYPE	NOT NULL
is_excluded	SE_INTEGER_TYPE	NOT NULL

SDE_XML_TAGS table

The SDE_XML_TAGS table contains a list of XML tags stored in the database.

SDE_XML_TAGS

Name	Data_Type	Null?
tag_id	SE_INTEGER_TYPE	NOT NULL
tag_nam	SE_INTEGER_TYPE	NOT NULL
data_type	SE_INTEGER_TYPE	NOT NULL
tag_alias	SE_INTEGER_TYPE	NULL
description	SE_STRING_TYPE(64)	NULL

RASTER_COLUMNS table

The RASTER_COLUMNS table contains a list of raster columns stored in the database. The table and raster column names, the owner, creation date, description, database name, configuration keyword, and minimum ID are included. The database name is required for some DBMSs.

RASTER_COLUMNS

Name	Data_Type	Null?
rastercolumn_id	SE_INTEGER_TYPE	NOT NULL
description	SE_STRING_TYPE(65)	NULL
database_name	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
table_name	SE_STRING_TYPE(160)	NOT NULL
raster_column	SE_STRING_TYPE(32)	NOT NULL
cdate	SE_INTEGER_TYPE	NOT NULL
config_keyword	SE_STRING_TYPE(32)	NULL
minimum_id	SE_INTEGER_TYPE	NULL
base_rastercolumn_id	SE_INTEGER_TYPE	NOT NULL
rastercolumn_mask	SE_INTEGER_TYPE	NOT NULL
srid	SE_INTEGER_TYPE	NULL

SPATIAL_REFERENCES table

The SPATIAL_REFERENCES table contains the coordinate system and floating point-to-integer transformation values. Internal functions use the parameters of a spatial reference system to translate and scale each floating point coordinate of the geometry into 64-bit positive integers prior to storage. Upon retrieval, the coordinates are restored to their original external floating point format.

SPATIAL_REFERENCES

Name	Data_Type	Null?
srid	SE_INTEGER_TYPE	NOT NULL
description	SE_STRING_TYPE(64)	NULL
auth_name	SE_STRING_TYPE(256)	NULL
auth_srid	SE_INTEGER_TYPE	NULL
falsex	SE_FLOAT_TYPE	NOT NULL
falsey	SE_FLOAT_TYPE	NOT NULL
xyunits	SE_FLOAT_TYPE	NOT NULL
falsez	SE_FLOAT_TYPE	NOT NULL
zunits	SE_FLOAT_TYPE	NOT NULL
falsem	SE_FLOAT_TYPE	NOT NULL
munits	SE_FLOAT_TYPE	NOT NULL
srtext	SE_STRING_TYPE(1024)	NOT NULL
object_flags	SE_INTEGER_TYPE	NOT NULL

TABLE_REGISTRY table

The TABLE_REGISTRY table manages all registered tables. The values include an ID, table name, owner, and description.

TABLE_REGISTRY

Name	Data_Type	Null?
registration_id	SE_INTEGER_TYPE	NOT NULL
table_name	SE_STRING_TYPE(160)	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
rowid_column	SE_STRING_TYPE(32)	NULL
description	SE_STRING_TYPE(65)	NULL
object_flags	SE_INTEGER_TYPE	NOT NULL
registration_date	SE_INTEGER_TYPE	NOT NULL
config_keyword	SE_STRING_TYPE(32)	NULL
minimum_id	SE_INTEGER_TYPE	NULL
imv_view_name	SE_STRING_TYPE(32)	NULL

COLUMN_REGISTRY table

The COLUMN_REGISTRY table manages all registered columns.

COLUMN_REGISTRY

Name	Data_Type	Null?
table_name	SE_STRING_TYPE(160)	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL

COLUMN_REGISTRY (continued)

Name	Data_Type	Null?
table_name	SE_STRING_TYPE(160)	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
column_name	SE_STRING_TYPE(32)	NOT NULL
sde_type	SE_INTEGER_TYPE	NOT NULL
column_size	SE_INTEGER_TYPE	NULL
decimal_digits	SE_INTEGER_TYPE	NULL
description	SE_STRING_TYPE(65)	NULL
object_flags	SE_INTEGER_TYPE	NOT NULL
object_id	SE_INTEGER_TYPE	NULL

VERSIONS table

The VERSIONS table contains the version metadata. The values include a name, owner, status (public or private), state ID, and description.

VERSIONS

Name	Data_Type	Null?
name	SE_STRING_TYPE(64)	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
version_id	SE_INTEGER_TYPE	NOT NULL
status	SE_INTEGER_TYPE	NOT NULL
state_id	SE_INTEGER_TYPE	NOT NULL
description	SE_STRING_TYPE(65)	NULL

VERSIONS (continued)

Name	Data_Type	Null?
parent_name	SE_STRING_TYPE(64)	NULL
parent_owner	SE_STRING_TYPE(32)	NULL
parent_version_id	SE_INTEGER_TYPE	NULL
creation_time	SE_DATE_TYPE	NOT NULL

STATES table

The STATES table contains the state metadata. The values include a state ID, owner, creation and closing time, state ID of the parent state, and lineage information.

STATES

Name	Data_Type	Null?
state_id	SE_INTEGER_TYPE	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
creation_time	SE_DATE_TYPE	NOT NULL
closing_time	SE_DATE_TYPE	NULL
parent_state_id	SE_INTEGER_TYPE	NOT NULL
lineage_name	SE_INTEGER_TYPE	NOT NULL

STATE_LINEAGES table

The STATE_LINEAGES table contains a state's ID and its ancestry lineage ID.

STATE_LINEAGES

Name	Data_Type	Null?
lineage_name	SE_INTEGER_TYPE	NOT NULL
lineage_id	SE_INTEGER_TYPE	NOT NULL

LINEAGES_MODIFIED table

The LINEAGES_MODIFIED table contains a state lineage ID and its most recent modification time stamp.

LINEAGES_MODIFIED

Name	Data_Type	Null?
lineage_name	SE_INTEGER_TYPE	NOT NULL
time_last_modified	SE_DATE_TYPE	NOT NULL

MVTABLES_MODIFIED table

The MVTABLES_MODIFIED table contains the state and table IDs modified in a given state.

MVTABLES_MODIFIED

Name	Data_Type	Null?
state_id	SE_INTEGER_TYPE	NOT NULL
registration_id	SE_INTEGER_TYPE	NOT NULL

LAYER_LOCKS table

The LAYER_LOCKS table maintains the locks on feature classes.

LAYER_LOCKS

Name	Data_Type	Null?
sde_id	SE_INTEGER_TYPE	NOT NULL
layer_id	SE_INTEGER_TYPE	NOT NULL
autolock	SE_STRING_TYPE(1)	NOT NULL
lock_type	SE_STRING_TYPE(1)	NOT NULL
minx	SE_INTEGER_TYPE	NULL
miny	SE_INTEGER_TYPE	NULL
maxx	SE_INTEGER_TYPE	NULL
maxy	SE_INTEGER_TYPE	NULL

STATE_LOCKS table

The STATE_LOCKS table maintains the version state locks.

STATE_LOCKS

Name	Data_Type	Null?
sde_id	SE_INTEGER_TYPE	NOT NULL
state_id	SE_INTEGER_TYPE	NOT NULL
autolock	SE_STRING_TYPE(1)	NOT NULL
lock_type	SE_STRING_TYPE(1)	NOT NULL

OBJECT_LOCKS table

The OBJECT_LOCKS table maintains locks on geodatabase objects.

OBJECT_LOCKS

Name	Data_Type	Null?
sde_id	SE_INTEGER_TYPE	NOT NULL
object_id	SE_INTEGER_TYPE	NOT NULL
object_type	SE_INTEGER_TYPE	NOT NULL
application_id	SE_INTEGER_TYPE	NOT NULL
autolock	SE_STRING_TYPE(1)	NOT NULL
lock_type	SE_STRING_TYPE(1)	NOT NULL

TABLE_LOCKS table

The TABLE_LOCKS table maintains the locks on ArcSDE registered tables.

TABLE_LOCKS

Name	Data_Type	Null?
sde_id	SE_INTEGER_TYPE	NOT NULL
registration_id	SE_INTEGER_TYPE	NOT NULL
lock_type	SE_STRING_TYPE(1)	NOT NULL

SDE_TABLES_MODIFIED table

The SDE_TABLES_MODIFIED table maintains the list of modified tables.

SDE_TABLES_MODIFIED

Name	Data_Type	Null?
table_name	SE_STRING_TYPE(32)	NOT NULL
time_last_modified	SE_DATE_TYPE	NOT NULL

METADATA table

The METADATA table contains ArcSDE metadata.

METADATA

Name	Data_Type	Null?
record_id	SE_INTEGER_TYPE	NOT NULL
object_name	SE_STRING_TYPE(160)	NOT NULL
object_owner	SE_STRING_TYPE(32)	NOT NULL
object_type	SE_INTEGER_TYPE	NOT NULL
class_name	SE_STRING_TYPE(32)	NULL
property	SE_STRING_TYPE(32)	NULL
prop_value	SE_STRING_TYPE(255)	NULL
description	SE_STRING_TYPE(65)	NULL
creation_date	SE_DATE_TYPE	NOT NULL

PROCESS_INFORMATION table

The PROCESS_INFORMATION table collects ArcSDE session statistics such as the number of records read and the number of records written while the session was active.

PROCESS_INFORMATION

Name	Data_Type	Null?
sde_id	SE_INTEGER_TYPE	NOT NULL
server_id	SE_INTEGER_TYPE	NOT NULL
start_time	SE_DATE_TYPE	NOT NULL
rcount	SE_INTEGER_TYPE	NOT NULL
wcount	SE_INTEGER_TYPE	NOT NULL
opcount	SE_INTEGER_TYPE	NOT NULL
numlocks	SE_INTEGER_TYPE	NOT NULL
fb_partial	SE_INTEGER_TYPE	NOT NULL
fb_count	SE_INTEGER_TYPE	NOT NULL
fb_fcount	SE_INTEGER_TYPE	NOT NULL
fb_kbyte	SE_INTEGER_TYPE	NOT NULL
owner	SE_STRING_TYPE(30)	NOT NULL
direct_connect	SE_STRING_TYPE(1)	NOT NULL
sysname	SE_STRING_TYPE(32)	NOT NULL
nodename	SE_STRING_TYPE(32)	NOT NULL
xdr_needed	SE_STRING_TYPE(1)	NOT NULL

SERVER_CONFIG table

The SERVER_CONFIG table stores ArcSDE server configuration parameters.

SERVER_CONFIG

Name	Data_Type	Null?
prop_name	SE_INTEGER_TYPE	NOT NULL
char_prop_value	SE_STRING_TYPE(32)	NOT NULL
num_prop_value	SE_STRING_TYPE(32)	NOT NULL

DBTUNE table

The DBTUNE table stores the configuration keywords for ArcSDE data objects.

DBTUNE

Name	Data_Type	Null?
keyword	SE_STRING_TYPE(32)	NOT NULL
parameter	SE_STRING_TYPE(32)	NOT NULL
config_string	SE_STRING_TYPE(2048)	NULL

LOCATORS table

The LOCATORS table stores information about locator objects.

LOCATORS

Name	Data_Type	Null?
locator_id	SE_INTEGER_TYPE	NOT NULL
name	SE_STRING_TYPE(32)	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
category	SE_STRING_TYPE(32)	NOT NULL
type	SE_INTEGER_TYPE	NOT NULL
description	SE_STRING_TYPE(64)	NULL

GCDRULES table

The GCDRULES table stores information about geocoding rules.

GCDRULES

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
style	SE_STRING_TYPE(32)	NULL
type	SE_STRING_TYPE(3)	NULL
data	SE_BLOB_TYPE	NULL

SDE_LOGFILE_POOL table

The SDE_LOGFILE_POOL table maintains the list of logfiles currently checked out.

SDE_LOGFILE_POOL

Name	Data_Type	Null?
table_id	SE_INTEGER_TYPE	NOT NULL
sde_id	SE_INTEGER_TYPE	NULL

SDE_LOGPOOL_<N> table

The SDE_LOGPOOL_<N> table can be checked out by users and stores either stand-alone or session-based logfiles. <N> is the sequence number.

SDE_LOGPOOL_<N>

Name	Data_Type	Null?
logfile_data_id	SE_INTEGER_TYPE	NOT NULL
sde_row_id	SE_INTEGER_TYPE	NOT NULL

GDB_ANNOSYMBOLS table

The GDB_ANNOSYMBOLS table contains feature class annotation. The values include annosymbol ID and the annotation string.

GDB_ANNOSYMBOLS

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
symbol	SE_BLOB_TYPE	NULL

GDB_ATTRRULES table

The GDB_ATTRRULES table contains the rules for each attribute domain.

GDB_ATTRRULES

Name	Data_Type	Null?
ruleid	SE_INTEGER_TYPE	NOT NULL
subtype	SE_INTEGER_TYPE	NOT NULL
fieldname	SE_STRING_TYPE(32)	NOT NULL
domainname	SE_STRING_TYPE(160)	NOT NULL

GDB_CODEDDOMAINS table

The GDB_CODEDDOMAINS table contains coded values for each domain.

GDB_CODEDDOMAINS

Name	Data_Type	Null?
domainid	SE_INTEGER_TYPE	NOT NULL
codedvalues	SE_BLOB_TYPE	NOT NULL

GDB_DEFAULTVALUES table

The GDB_DEFAULTVALUES table contains the default values for the subtypes of each object class.

GDB_DEFAULTVALUES

Name	Data_Type	Null?
classid	SE_INTEGER_TYPE	NOT NULL
fieldname	SE_STRING_TYPE(32)	NOT NULL
subtype	SE_INTEGER_TYPE	NOT NULL
defaultstring	SE_STRING_TYPE(160)	NULL
defaultnumber	SE_DOUBLE_TYPE(38,8)	NULL

GDB_DOMAINS table

The GDB_DOMAINS table contains the attribute constraints associated with attribute rules of the GDB_ATTRRULES table.

GDB_DOMAINS

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
domainname	SE_STRING_TYPE(160)	NOT NULL
description	SE_STRING_TYPE(160)	NULL
domaintype	SE_INTEGER_TYPE	NOT NULL
fieldtype	SE_INTEGER_TYPE	NOT NULL
mergepolicy	SE_INTEGER_TYPE	NOT NULL
splitpolicy	SE_INTEGER_TYPE	NOT NULL

GDB_EDGECONNRULES table

The GDB_EDGECONNRULES table contains the edge connectivity rules. Edge connectivity rules, together with junction rules, function to define the geometric networks stored in the gdb_geonetwork table.

GDB_EDGECONNRULES

Name	Data_Type	Null?
Ruleid	SE_INTEGER_TYPE	NOT NULL
Fromclassid	SE_INTEGER_TYPE	NOT NULL
Fromsubtype	SE_INTEGER_TYPE	NOT NULL

GDB_EDGECONNRULES (continued)

Name	Data_Type	Null?
Toclassid	SE_INTEGER_TYPE	NOT NULL
Tosubtype	SE_INTEGER_TYPE	NOT NULL
Junctions	SE_BLOB_TYPE	NOT NULL

GDB_FEATURECLASSES table

The GDB_FEATURECLASSES table contains the feature classes.

GDB_FEATURECLASSES

Name	Data_Type	Null?
objectclassid	SE_INTEGER_TYPE	NOT NULL
featuretype	SE_INTEGER_TYPE	NOT NULL
geometrytype	SE_INTEGER_TYPE	NOT NULL
shapefield	SE_STRING_TYPE(32)	NOT NULL
geomnetworkid	SE_INTEGER_TYPE	NULL
graphid	SE_INTEGER_TYPE	NULL

GDB_FEATUREDATASET table

The GDB_FEATUREDATASET table contains the feature datasets. A *feature dataset* is a grouping of feature classes.

GDB_FEATUREDATASET

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
srid	SE_INTEGER_TYPE	NOT NULL

GDB_FIELDINFO table

The GDB_FIELDINFO table contains the field name, default domain name values, default string, and number values for each attribute field associated with a feature class.

GDB_FIELDINFO

Name	Data_Type	Null?
classid	SE_INTEGER_TYPE	NOT NULL
fieldname	SE_STRING_TYPE(160)	NOT NULL
aliasname	SE_STRING_TYPE(160)	NULL
modelname	SE_STRING_TYPE(160)	NULL
defaultdomainname	SE_STRING_TYPE(160)	NULL
defaultvaluestring	SE_STRING_TYPE(160)	NULL
defaultvaluenumber	SE_DOUBLE_TYPE(38,8)	NULL

GDB_FIELDINFO (continued)

Name	Data_Type	Null?
isrequired	SE_INTEGER_TYPE	NOT NULL
issubtypefixed	SE_INTEGER_TYPE	NOT NULL
iseditable	SE_INTEGER_TYPE	NOT NULL

GDB_GEOMNETWORKS table

The GDB_GEOMNETWORKS table contains the geometric networks of a feature dataset.

GDB_GEOMNETWORKS

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
networktype	SE_INTEGER_TYPE	NOT NULL
datasetid	SE_INTEGER_TYPE	NOT NULL

GDB_EXTENSIONS table

The GDB_EXTENSIONS table contains a list of currently registered workspace extensions in the geodatabase.

GDB_EXTENSIONS

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
clsid	SE_STRING_TYPE(38)	NOT NULL

GDB_JNCONNRULES table

The GDB_JNCONNRULES table contains the junction connectivity rules. Junction connectivity rules, together with edges rules, function to define the geometric networks stored in the GDB_GEOMNETWORKS table.

GDB_JNCONNRULES

Name	Data_Type	Null?
ruleid	SE_INTEGER_TYPE	NOT NULL
edgeclassid	SE_INTEGER_TYPE	NOT NULL
edgesubtype	SE_INTEGER_TYPE	NOT NULL
edgemincard	SE_INTEGER_TYPE	NOT NULL
edgemaxcard	SE_INTEGER_TYPE	NOT NULL
junctionclassid	SE_INTEGER_TYPE	NOT NULL
junctionsubtype	SE_INTEGER_TYPE	NOT NULL
junctionmincard	SE_INTEGER_TYPE	NOT NULL

GDB_JNCONNRULES (continued)

Name	Data_Type	Null?
junctionmaxcard	SE_INTEGER_TYPE	NOT NULL
isdefault	SE_INTEGER_TYPE	NULL

GDB_NETCLASSES table

The GDB_NETCLASSES table contains the network classes of the geometric networks.

GDB_NETCLASSES

Name	Data_Type	Null?
classid	SE_INTEGER_TYPE	NOT NULL
networkid	SE_INTEGER_TYPE	NOT NULL
enabledfield	SE_STRING_TYPE(32)	NULL
ancillaryrole	SE_INTEGER_TYPE	NULL
ancillaryfield	SE_STRING_TYPE(32)	NULL

GDB_RANGEDOMAINS table

The GDB_RANGEDOMAINS table contains the range of possible values allowed in a domain.

GDB_RANGEDOMAINS

Name	Data_Type	Null?
domainid	SE_INTEGER_TYPE	NOT NULL
minvalue	SE_DOUBLE_TYPE(38,8)	NOT NULL
maxvalue	SE_DOUBLE_TYPE(38,8)	NOT NULL

GDB_NETWEIGHTS table

The GDB_NETWEIGHTS table contains the network weights of the geometric networks.

GDB_NETWEIGHTS

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
networkid	SE_INTEGER_TYPE	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
weightid	SE_INTEGER_TYPE	NOT NULL
weighttype	SE_INTEGER_TYPE	NOT NULL
bitgatesize	SE_INTEGER_TYPE	NULL

GDB_NETWEIGHTASOCS table

The GDB_NETWEIGHTASOCS table contains the association between the network classes and the network weights of the geometric networks.

GDB_NETWEIGHTASOCS

Name	Data_Type	Null?
networkid	SE_INTEGER_TYPE	NOT NULL
weightid	SE_INTEGER_TYPE	NOT NULL
tablename	SE_STRING_TYPE(160)	NOT NULL
fieldname	SE_STRING_TYPE(32)	NULL

GDB_NETWORKS table

The GDB_NETWORKS table contains the logical networks.

GDB_NETWORKS

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
networktype	SE_INTEGER_TYPE	NOT NULL
indextype	SE_INTEGER_TYPE	NOT NULL
normalized	SE_INTEGER_TYPE	NOT NULL

GDB_SUBTYPES table

The GDB_SUBTYPES table contains the valid subtypes of the geodatabase object classes.

GDB_SUBTYPES

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
classid	SE_INTEGER_TYPE	NOT NULL
subtypecode	SE_INTEGER_TYPE	NOT NULL
subtypename	SE_STRING_TYPE(160)	NOT NULL

GDB_OBJECTCLASSES table

The GDB_OBJECTCLASSES table contains all of the object classes in the geodatabase, which includes the feature classes, relationship classes, business tables, and columns.

GDB_OBJECTCLASSES

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
aliasname	SE_STRING_TYPE(160)	NULL
modelname	SE_STRING_TYPE(160)	NULL
clsid	SE_STRING_TYPE(38)	NOT NULL
extclsid	SE_STRING_TYPE(38)	NULL
extprops	SE_BLOB_TYPE	NULL
subtypefield	SE_STRING_TYPE(32)	NULL
datasetid	SE_INTEGER_TYPE	NULL

GDB_STRINGDOMAINS table

The GDB_STRINGDOMAINS table stores a domain's format string.

GDB_STRINGDOMAINS

Name	Data_Type	Null?
domainid	SE_INTEGER_TYPE	NOT NULL
format	SE_STRING_TYPE(32)	NOT NULL

GDB_RELCLASSES table

The GDB_RELCLASSES table contains the table relationships required by the geodatabase.

GDB_RELCLASSES

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
originclassid	SE_INTEGER_TYPE	NOT NULL
destclassid	SE_INTEGER_TYPE	NOT NULL
forwardlabel	SE_STRING_TYPE(32)	NULL
backwardlabel	SE_STRING_TYPE(32)	NULL
cardinality	SE_INTEGER_TYPE	NOT NULL
notification	SE_INTEGER_TYPE	NOT NULL
iscomposite	SE_INTEGER_TYPE	NOT NULL
isattributed	SE_INTEGER_TYPE	NOT NULL
originprimarykey	SE_STRING_TYPE(32)	NOT NULL
destprimarykey	SE_STRING_TYPE(32)	NOT NULL
originforeignkey	SE_STRING_TYPE(32)	NOT NULL
destforeignkey	SE_STRING_TYPE(32)	NOT NULL
datasetid	SE_INTEGER_TYPE	NULL

GDB_RELEASE table

The GDB_RELEASE table stores geodatabase version release information as a single record.

GDB_RELEASE

Name	Data_Type	Null?
major	SE_INTEGER_TYPE	NOT NULL
minor	SE_INTEGER_TYPE	NOT NULL
bugfix	SE_INTEGER_TYPE	NOT NULL

GDB_RELRULES table

The GDB_RELRULES table contains the object class relationship rules.

GDB_RELRULES

Name	Data_Type	Null?
ruleid	SE_INTEGER_TYPE	NOT NULL
originsubtype	SE_INTEGER_TYPE	NOT NULL
originmincard	SE_INTEGER_TYPE	NOT NULL
originmaxcard	SE_INTEGER_TYPE	NOT NULL
destsubtype	SE_INTEGER_TYPE	NOT NULL
destmincard	SE_INTEGER_TYPE	NOT NULL
destmaxcard	SE_INTEGER_TYPE	NOT NULL

GDB_SPATIALRULES table

The GDB_SPATIALRULES table contains the spatial rules of the geodatabase. Such rules determine which spatial relationships are permitted.

GDB_SPATIALRULES

Name	Data_Type	Null?
rruleid	SE_INTEGER_TYPE	NOT NULL
subtype	SE_INTEGER_TYPE	NOT NULL
spatialrel	SE_INTEGER_TYPE	NOT NULL
distance	SE_DOUBLE_TYPE(38,8)	NOT NULL
relclassid	SE_INTEGER_TYPE	NOT NULL
relsubtype	SE_INTEGER_TYPE	NOT NULL

GDB_USERMETADATA table

The GDB_USERMETADATA table stores user metadata for all parts of the geodatabase including object classes, feature classes, feature datasets, logical networks, and relationship classes.

GDB_USERMETADATA

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL

GDB_USERMETADATA (continued)

Name	Data_Type	Null?
datasettype	SE_INTEGER_TYPE	NOT NULL
xml	SE_BLOB_TYPE	NOT NULL

GDB_VALIDRULES table

The GDB_VALIDRULES table contains all the valid rules of the geodatabase, which includes the attribute rules, edge connectivity rules, junction connectivity rules, relationship rules, and spatial rules.

GDB_VALIDRULES

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
ruletype	SE_INTEGER_TYPE	NOT NULL
classid	SE_INTEGER_TYPE	NOT NULL
rulecategory	SE_INTEGER_TYPE	NOT NULL
helpstring	SE_STRING_TYPE(160)	NULL

GDB_REPLICADATASETS table

The GDB_REPLICADATASETS table contains one record of metadata for each dataset that is part of a replica (single, check-out/check-in, or multigeneration) in a geodatabase.

GDB_REPLICADATASETS

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
replicaid	SE_INTEGER_TYPE	NOT NULL
datasettype	SE_INTEGER_TYPE	NOT NULL
datasetid	SE_INTEGER_TYPE	NOT NULL
parentowner	SE_STRING_TYPE(32)	NOT NULL
parentdb	SE_STRING_TYPE(32)	NOT NULL

GDB_REPLICAS table

The GDB_REPLICAS table contains one record of metadata for each replica (single, check-out/check-in, or multigeneration) in the geodatabase.

GDB_REPLICAS

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
name	SE_STRING_TYPE(32)	NOT NULL
owner	SE_STRING_TYPE(32)	NOT NULL
version	SE_STRING_TYPE(64)	NOT NULL
parentid	SE_INTEGER_TYPE	NOT NULL

GDB_REPLICAS (continued)

Name	Data_Type	Null?
repdate	SE_DATE_TYPE	NOT NULL
defquery	SE_BLOB_TYPE	NOT NULL
repguid	SE_STRING_TYPE(36)	NOT NULL
repcinfo	SE_STRING_TYPE(1800)	NULL
role	SE_INTEGER_TYPE	NOT NULL

GDB_TOOLBOXES table

The GDB_TOOLBOXES table contains one record of metadata for each toolbox in the geodatabase.

GDB_TOOLBOXES

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
alias	SE_STRING_TYPE(160)	NULL
metadata	SE_BLOB_TYPE	NULL
helpfile	SE_STRING_TYPE(255)	NULL
helpcontent	SE_INTEGER_TYPE	NULL

GDB_TOPOCLASSES table

The GDB_TOPOCLASSES table contains one record of metadata for each feature class in a topology.

GDB_TOPOCLASSES

Name	Data_Type	Null?
classid	SE_INTEGER_TYPE	NOT NULL
topologyid	SE_INTEGER_TYPE	NOT NULL
weight	SE_FLOAT_TYPE(32)	NOT NULL
xyrank	SE_INTEGER_TYPE	NOT NULL
zrank	SE_INTEGER_TYPE	NOT NULL
eventsonanalyze	SE_INTEGER_TYPE	NOT NULL

GDB_TOPOLOGIES table

The GDB_TOPOLOGIES table contains one record of metadata for each topology in the geodatabase.

GDB_TOPOLOGIES

Name	Data_Type	Null?
id	SE_INTEGER_TYPE	NOT NULL
databasename	SE_STRING_TYPE(32)	NULL
owner	SE_STRING_TYPE(32)	NOT NULL
name	SE_STRING_TYPE(160)	NOT NULL
datasetid	SE_INTEGER_TYPE	NOT NULL
properties	SE_BLOB_TYPE	NULL

GDB_TOPORULES table

The GDB_TOPORULES table contains one record of metadata for each rule in each topology.

GDB_TOPORULES

Name	Data_Type	Null?
ruleid	SE_INTEGER_TYPE	NOT NULL
originclassid	SE_STRING_TYPE(32)	NOT NULL
originsubtype	SE_STRING_TYPE(32)	NOT NULL
alloriginsubtypes	SE_STRING_TYPE(64)	NOT NULL
destclassid	SE_INTEGER_TYPE	NOT NULL
destsubtype	SE_DATE_TYPE	NOT NULL
alldestsubtypes	SE_BLOB_TYPE	NOT NULL
topologyruletype	SE_STRING_TYPE(36)	NOT NULL
name	SE_STRING_TYPE(160)	NULL
ruleguid	SE_STRING_TYPE(38)	NOT NULL

GDB_RASTERCATALOGS table

The GDB_RASTERCATALOGS table stores a reference to each raster catalog in the geodatabase.

GDB_RASTERCATALOGS

Name	Data_Type	Null?
objectclassid	SE_INTEGER_TYPE	NOT NULL
rasterfield	SE_STRING_TYPE(32)	NULL
israsterdataset	SE_INTEGER_TYPE	NOT NULL
objectid	SE_INTEGER_TYPE	NOT NULL

GDB_TABLES_LAST_MODIFIED table

The GDB_TABLES_LAST_MODIFIED table is used for the geodatabase system table caching and keeping track of when a table has changed and needs to be updated in the cache.

GDB_TABLES_LAST_MODIFIED

Name	Data_Type	Null?
table_name	SE_STRING_TYPE(160)	NOT NULL
last_modified_count	SE_INTEGER_TYPE	NOT NULL

ArcSDE table definitions

B

IN THIS APPENDIX

- **Business tables**
- **Binary schema implementation**
- **Feature table**
- **Spatial index table**
- **Spatial types and functions schema**
- **Logical network tables**
- **Topology error tables**
- **Logfile tables**
- **Version tables**
- **Raster tables**
- **Geocoding index tables**

This appendix lists the ArcSDE tables that a user can create. These tables should be included within your existing backup arrangements. With the exception of the business table, these tables should only be accessed through the application interface provided either by ArcInfo or an ArcSDE C API program. Direct access to the nonbusiness tables via the Structured Query Language interface is not supported.

Business tables

The business table is an existing DBMS table that ArcSDE spatially enables by adding a spatial column. A business table with a spatial column is a *feature class,* and information about each feature class is maintained in the LAYERS table.

A business table with a raster column is a raster dataset or a raster catalog. A raster dataset can have only one business table row, while a raster catalog can have more than one. Information about the raster dataset or raster catalog is maintained in the RASTER_COLUMNS table.

Information about all business tables, regardless of whether or not they have a spatial column or raster column, is maintained in the TABLE_REGISTRY table.

Binary schema implementation

The data type of the spatial column varies depending on the implementation of ArcSDE.

Some ArcSDE implementations employ the binary schema. Under this implementation, the integer spatial column contains feature identifiers that uniquely reference the spatial data. The feature ID joins the business table with the associated ArcSDE software-managed feature and spatial index tables.

A database trigger is defined on the spatially enabled business table to maintain the relationship between records in the business table and the feature table. The trigger is:

```
TRIGGER  SPCOL_DEL_CASCADE_<layer>
AFTER DELETE OR UPDATE OF <SPATIAL_COL> ON
business_table
    IF  DELETING  THEN
        DELETE  FROM  F<layer>
        WHERE F<layer>.fid = old.<SPATIAL_COL>
        DELETE FROM S<layer>
        WHERE S<layer>.fid = old.<SPATIAL_COL>


   IF UPDATING AND new.<SPATIAL_COL> IS NULL THEN
      DELETE FROM F<layer>
         WHERE F<layer>.fid = old.<SPATIAL_COL>
      DELETE FROM S<layer>
         WHERE S<layer>.fid = old.<SPATIAL_COL>


   IF UPDATING AND new.<SPATIAL_COL> !=
                             old.<SPATIAL_COL> THEN
     raise_application_error
       (-20013,'Invalid  update  of  SDE  Spatial
       Column.')
```

Feature table

Under the binary schema implementation, the feature table stores the geometric shapes for each feature. This table is identified by the spatial column layer number using the name F<layer_id>.

The relationship between the business table and the feature table is managed through the Feature ID, or FID. This key, which is maintained by ArcSDE, is unique for the spatial column.

F<layer_id>

NAME	DATA_TYPE	NULL?
fid	SE_INTEGER_TYPE	NOT NULL
numofpts	SE_INTEGER_TYPE	NOT NULL
entity	SE_SMALLINT_TYPE	NOT NULL
eminx	SE_FLOAT_TYPE(64)	NOT NULL
eminy	SE_FLOAT_TYPE(64)	NOT NULL
emaxx	SE_FLOAT_TYPE(64)	NOT NULL
emaxy	SE_FLOAT_TYPE(64)	NOT NULL
eminz	SE_FLOAT_TYPE(64)	NULL
emaxz	SE_FLOAT_TYPE(64)	NULL
min_measure	SE_FLOAT_TYPE(64)	NULL
max_measure	SE_FLOAT_TYPE(64)	NULL
area	SE_FLOAT_TYPE(64)	NOT NULL
len	SE_FLOAT_TYPE(64)	NOT NULL
points	SE_BLOB_TYPE	NULL

The feature table stores the geometry and has several additional columns to support ArcSDE query processing.

For storing the geometry:

- Points (SE_BLOB_TYPE)—contains the byte stream of point coordinates that define the geometry

For query processing:

- Area (SE_FLOAT_TYPE)—the area of the geometry
- Len (SE_FLOAT_TYPE)—the length or perimeter of the geometry
- Eminx, eminy, emaxx, emaxy (SE_FLOAT_TYPE)—the envelope of the geometry
- Eminz (SE_FLOAT_TYPE)—the minimum z-value in the geometry
- Emaxz (SE_FLOAT_TYPE)—the maximum z-value in the geometry
- Min_measure (SE_FLOAT_TYPE)—the minimum measure value in the geometry
- Max_measure (SE_FLOAT_TYPE)—the maximum measure value in the geometry

For internal ArcSDE use:

- Fid (SE_INTEGER_TYPE)—contains the unique ID that joins the feature table to the business table
- Entity (SE_INTEGER_TYPE)—the type geometry stored in the spatial column (for example, point, line, or string)
- Numofpts (SE_INTEGER_TYPE)—the number of points defining the geometry

As geometries are inserted or updated, the extents, numofpts, and so on, are recalculated automatically. The points column contains the coordinate array for the geometry in a compressed integer format. The binary layout of this data is discussed in the *ArcSDE Developer Help*, located in the SDEHOME\documentation directory.

Spatial index table

The spatial index of the binary implementation is the spatial index table. It stores references to shapes based on a simple, regular grid. This table is identified by the spatial column layer number using the name S<layer_id>.

The spatial index contains an entry for each shape and grid cell combination. A feature that crosses into three grid cells has three entries in the table. When a spatial query is performed, the grid cells within the search area are identified and used to return a list of candidate geometries.

- Sp_fid (SE_INTEGER_TYPE)—contains the ID that joins the spatial index table to the feature table
- Gx, gy (SE_INTEGER_TYPE)—the grid cell coordinate
- Eminx, eminy, emaxx, emaxy (SE_INTEGER_TYPE)—the envelope of the geometry

S<layer_id>

NAME	DATA TYPE	NULL?
sp_fid	SE_INTEGER_TYPE	NOT NULL
gx	SE_INTEGER_TYPE	NOT NULL
gy	SE_INTEGER_TYPE	NOT NULL
eminx	SE_INTEGER_TYPE	NOT NULL
eminy	SE_INTEGER_TYPE	NOT NULL
emaxx	SE_INTEGER_TYPE	NOT NULL
emaxy	SE_INTEGER_TYPE	NOT NULL

Spatial types and functions schema

Some of the implementations of ArcSDE employ the latest object–relation technology. Instead of using a separate table to store the geometries, the spatial types and functions schema stores them directly in the spatial column. Under this implementation, the spatial column is a user-defined type (UDT).

The ArcSDE spatial types and functions implementation includes seven of the UDTs defined by OpenGIS® RFP-1. Those data types are point, line string, polygon, multipoint, multilinestring, multipolygon, and geometry.

The database administrator normally runs a program that adds these UDTs and a number of spatial user-defined functions (UDFs) to a database. After the UDTs and UDFs have been added, the database becomes “spatially enabled”. Users may now create tables that include columns whose types are of the seven spatial UDTs listed above. As a bonus, the spatial types and functions allow users to perform spatial queries using the SQL interface provided by the DBMS vendor.

Logical network tables

The logical network tables provide connectivity and flow direction information between elements, junctions, and edges in a network.

The N_*_DESC table maintains a description of each element in the network.

N_*_DESC

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
userclassid	SE_SMALLINT_TYPE	NOT NULL
userid	SE_INTEGER_TYPE	NOT NULL
usersubid	SE_INTEGER_TYPE	NOT NULL
elementtype	SE_SMALLINT_TYPE	NOT NULL
eid	SE_INTEGER_TYPE	NOT NULL

The N_*_EDESC table contains a description of the edges within a network.

N_*_EDESC

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_ESTATUS table contains the status of each element including its deleted state and disabled state.

N_*_ESTATUS

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_ETOPO table contains the edge topology.

N_*_ETOPO

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_FLODIR table contains the flow direction.

N_*_FLODIR

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_JDESC table contains a description of the junctions of a network.

N_*_JDESC

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_JSTATUS table contains the status of each element including its deleted state and disabled state.

N_*_JSTATUS

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_JTOPO table contains the junction topology.

N_*_JTOPO

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_JTOPO2 table contains the overflow junction topology.

N_*_JTOPO2

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
pagenumber	SE_INTEGER_TYPE	NOT NULL
pageblob	SE_BLOB_TYPE	NULL

The N_*_PROPS table contains the network properties including the number of junctions, edges, and turns.

N_*_PROPS

Name	Data_Type	Null?
propertyid	SE_INTEGER_TYPE	NULL
propertyname	SE_STRING_TYPE(255)	NULL
propertyvalue	SE_INTEGER_TYPE	NULL

The N_*_E* table contains all of the network edge weights.

N_*_E*

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
name	SE_STRING_TYPE	NULL
internalid	SE_INTEGER_TYPE	NULL
weighttype	SE_INTEGER_TYPE	NULL
maxvalue	SE_DOUBLE_TYPE(38,8)	NULL
minvalue	SE_DOUBLE_TYPE(38,8)	NULL

The N_*_J* table contains all of the network junction weights.

N_*_J*

Name	Data_Type	Null?
oid	SE_INTEGER_TYPE	NOT NULL
name	SE_STRING_TYPE	NULL
internalid	SE_INTEGER_TYPE	NULL
weighttype	SE_INTEGER_TYPE	NULL
maxvalue	SE_DOUBLE_TYPE(38,8)	NULL
minvalue	SE_DOUBLE_TYPE(38,8)	NULL

Topology error tables

The topology error tables store the topology errors of the geodatabase.

The T_*_POLYERRORS table contains:

T_*_POLYERRORS

Name	Data_Type	Null?
objectid	SE_INTEGER_TYPE	NOT NULL
originclassid	SE_INTEGER_TYPE	NOT NULL
originid	SE_INTEGER_TYPE	NULL
destclassid	SE_INTEGER_TYPE	NULL
destid	SE_INTEGER_TYPE	NULL
toporuletype	SE_INTEGER_TYPE	NOT NULL
toporuleid	SE_INTEGER_TYPE	NOT NULL
isexception	SE_INTEGER_TYPE	NOT NULL
shape	SE_INTEGER_TYPE	NULL

The T_*_DIRTYAREAS table contains:

T_*_DIRTYAREAS

Name	Data_Type	Null?
objectid	SE_INTEGER_TYPE	NOT NULL
originclassid	SE_INTEGER_TYPE	NOT NULL
originid	SE_INTEGER_TYPE	NULL

The T_*_LINEERRORS table contains:

T_*_LINEERRORS

Name	Data_Type	Null?
objectid	SE_INTEGER_TYPE	NOT NULL
originclassid	SE_INTEGER_TYPE	NOT NULL
originid	SE_INTEGER_TYPE	NULL
destclassid	SE_INTEGER_TYPE	NULL
destid	SE_INTEGER_TYPE	NULL
toporuletype	SE_INTEGER_TYPE	NOT NULL
toporuleid	SE_INTEGER_TYPE	NOT NULL
isexception	SE_INTEGER_TYPE	NOT NULL
shape	SE_INTEGER_TYPE	NULL

The T_*_POINTERRORS table contains:

T_*_POINTERRORS

Name	Data_Type	Null?
objectid	SE_INTEGER_TYPE	NOT NULL
originclassid	SE_INTEGER_TYPE	NOT NULL
originid	SE_INTEGER_TYPE	NULL
destclassid	SE_INTEGER_TYPE	NULL
destid	SE_INTEGER_TYPE	NULL
toporuletype	SE_INTEGER_TYPE	NOT NULL
toporuleid	SE_INTEGER_TYPE	NOT NULL
isexception	SE_INTEGER_TYPE	NOT NULL
shape	SE_INTEGER_TYPE	NULL

Logfile tables

The SDE_LOGFILES table contains the logfile metadata. The values include a logfile name and ID, the registration ID, flags, and session information.

SDE_LOGFILES

Name	Data_Type	Null?
logfile_name	SE_STRING_TYPE(256)	NOT NULL
logfile_id	SE_INTEGER_TYPE	NOT NULL
logfile_data_id	SE_INTEGER_TYPE	NOT NULL
registration_id	SE_INTEGER_TYPE	NOT NULL
flags	SE_INTEGER_TYPE	NOT NULL
session_tag	SE_INTEGER_TYPE	NOT NULL
logfile_data_db	SE_STRING_TYPE(32)	NULL
logfile_data_owner	SE_STRING_TYPE(32)	NULL
logfile_data_table	SE_STRING_TYPE(98)	NULL

The SDE_LOGFILE_DATA table contains the list of business table records that are part of each logfile (8.x logfiles).

SDE_LOGFILE_DATA

Name	Data_Type	Null?
logfile_data_id	SE_INTEGER_TYPE	NOT NULL
sde_row_id	SE_INTEGER_TYPE	NOT NULL

The SDE_LOGDATA<N>_<M> table contains the list of business table records that are part of a standalone logfile. <N> denotes the session ID and <M> denotes the standalone logfile sequence.

SDE_LOGDATA<N>_<M>

Name	Data_Type	Null?
sde_row_id	SE_INTEGER_TYPE	NOT NULL

The SDE_SESSION<N> table stores the list of session-based logfile records. <N> denotes the session ID.

SDE_SESSION<N>

Name	Data_Type	Null?
logfile_data_id	SE_INTEGER_TYPE	NOT NULL
sde_row_id	SE_INTEGER_TYPE	NOT NULL

Version tables

The version tables store information about the changes (additions or deletions) that are made to a versioned business table.

The A<registration_ID> table (or the “adds” table) stores the records added to each state in a versioned business table.

A<registration_ID>

Name	Data_Type	Null?
User-defined column	User-defined data	
names	types	
•	•	
•	•	
•	•	
<row id>	SE_INTEGER_TYPE	NOT NULL
sde_state_id	SE_INTEGER_TYPE	NOT NULL

Conversely, the D<registration_ID> table (or the “deletes” table) stores the records deleted from each state in a versioned business table.

D<registration_ID>

Name	Data_Type	Null?
deleted_at	SE_INTEGER_TYPE	NOT NULL
sde_deletes_row_id	SE_INTEGER_TYPE	NOT NULL
sde_state_id	SE_INTEGER_TYPE	NOT NULL

Raster tables

ArcSDE stores images in three raster tables. The raster blocks table—SDE_BLK_<raster_column_ID>—stores the actual image data for each band of the image. The raster band table—SDE_BND_<raster_column_ID>— stores metadata about the bands of the image. The raster description table—SDE_RAS_<raster_column_ID>—stores the description of the images within a raster column. The raster column that is added to the business table stores the raster ID, which joins the business table to the three raster tables.

The raster blocks table—SDE_BLK_<raster_column_ID>—stores the image tiles.

SDE_BLK_<raster_column_ID>

Name	Data_Type	Null?
rasterband_id	SE_INTEGER_TYPE	NOT NULL
rrd_factor	SE_INTEGER_TYPE	NOT NULL
row_nbr	SE_INTEGER_TYPE	NOT NULL
col_nbr	SE_INTEGER_TYPE	NOT NULL
block_data	SE_BLOB_TYPE	NOT NULL

The raster description table—SDE_RAS_<raster_column_ID>—stores the description of the images stored in a raster column.

SDE_RAS_<raster_column_ID>

Name	Data_Type	Null?
raster_id	SE_INTEGER_TYPE	NOT NULL
description	SE_STRING_TYPE(65)	NULL

The raster band table—SDE_BND_<raster_column_ID>—stores information about the bands of the images. Among other things this table stores are the width and height of the band tiles, the pixel type, and pixel depth.

SDE_BND_<raster_column_ID>

Name	Data_Type	Null?
rasterband_id	SE_INTEGER_TYPE	NOT NULL
sequence_nbr	SE_INTEGER_TYPE	NOT NULL
raster_id	SE_INTEGER_TYPE	NOT NULL
name	SE_STRING_TYPE(65)	NULL
band_flags	SE_INTEGER_TYPE	NOT NULL
band_width	SE_INTEGER_TYPE	NOT NULL
band_height	SE_INTEGER_TYPE	NOT NULL
band_types	SE_INTEGER_TYPE	NOT NULL
block_width	SE_INTEGER_TYPE	NOT NULL
block_height	SE_INTEGER_TYPE	NOT NULL
eminx	SE_FLOAT_TYPE(64)	NOT NULL
eminy	SE_FLOAT_TYPE(64)	NOT NULL
emaxx	SE_FLOAT_TYPE(64)	NOT NULL
emaxy	SE_FLOAT_TYPE(64)	NOT NULL
cdate	SE_FLOAT_TYPE(64)	NOT NULL
mdate	SE_FLOAT_TYPE(64)	NOT NULL
statistics	SE_BLOB_TYPE	NULL

The raster auxiliary table—SDE_AUX_<raster_column_ID>—stores the image color map, image statistics, and the optional bit mask, which is used for image overlays and mosaicking.

SDE_AUX_<raster_column_ID>

Name	Data_Type	Null?
rasterband_id	SE_INTEGER_TYPE	NOT NULL
type	SE_INTEGER_TYPE	NOT NULL
object	SE_BLOB_TYPE	NOT NULL

Geocoding index tables

When the locator framework creates a new locator, it may create one or more geocoding index tables, depending upon the locator style. In general, geocoding index tables are named GC_<$$$$><id#>, where <$$$$> is a string specified by the locator style, and <id#> is the geodatabase object class ID of the reference dataset that is indexed by the geocoding index table.

The content of a geocoding index table depends on the style of the address locator using it. The following table definition describes a geocoding index table schema created for the primary reference data feature class by a US Streets with Zone locator.

GC_SZS<id#>

Name	Data_Type	Null?
sx	SE_STRING_TYPE(4)	NULL
xid	SE_INTEGER_TYPE	NULL
lzone	SE_STRING_TYPE(5)	NULL
rzone	SE_STRING_TYPE(5)	NULL

ArcSDE application server command references

C

IN THIS APPENDIX

- **Command listing**
- **Command syntax**
- **Getting help**
- **ArcSDE administration commands: sdemon**
- **ArcSDE administration commands: sdeservice**
- **ArcSDE administration commands: sdesetup**

Some of the administration commands allow the ArcSDE administrator, usually a database administrator, to manage and monitor the use of an ArcSDE application server. This appendix describes these commands in detail, providing both the syntax and example usage.

Other administration commands allow users to create and modify the schema of an ArcSDE database. Those commands are not documented here but can be found in *ArcSDE Developer Help*.

Command listing

The following commands are used to manage the service. Other commands, such as those used to convert data and create and maintain schema, are documented in *ArcSDE Developer Help*.

ArcSDE command	Description
sdemon	Manages the ArcSDE application server
sdeconfig	Manages the ArcSDE application servers configuration parameters
sdeservice	Manages the ArcSDE service on Windows NT platforms
sdesetupdb2	Initial setup program for ArcSDE for DB2
sdesetupinfx	Initial setup program for ArcSDE for Informix
sdesetupora8i	Initial setup program for ArcSDE for Oracle8*i*™
sdesetupora9i	Initial setup program for ArcSDE for Oracle9*i*™
sdesetupmssql	Initial setup program for ArcSDE for SQL Server

Command syntax

The administration commands use UNIX command syntax and notation according to the following conventions:

-letter	Specifies a command operation or option, for example, "-o" or "-a". Letters are both lowercase and uppercase.
<>	Required argument. Replace appropriately. For example, "-u <DB_User_name>" could become "-u av".
|	Mutually exclusive arguments. Pick one from the list.
{ }	Used with " | " to specify a list of choices for an argument.
[ ]	Optional parameter.

Each command has "operations" and "options". An operation does a specific task related to the command and is specified with "-o <task>". For example, some of the sdetable command operations are:

```
sdemon  -o status
sdemon  -o start
sdemon  -o shutdown
sdemon  -o pause
sdemon  -o resume
sdemon  -o info
sdemon  -o kill
```

Each operation has a set of options. Just like operations, options are specified by "-<letter>". The "-<letter>" used for a particular option is standard across all commands. For instance, the option to specify a service is always "-i". Sometimes a letter is used for two different types of options but never in the same command.

For each command's operation, there are mandatory and nonmandatory options.

```
sdemon  -o status [[-i <service>] [-s
<server_name>] | [-H <sde_directory>]] [-q]
```

The example above has two options. Anything enclosed within "[ ]" isn't required. The status operation is required, while service "[i]" and quiet "[q]" aren't. Sometimes an option marked optional is not truly optional. The most common occurrence is "[-p <DB_User_password]". It is optional on the command line, but ArcSDE will query for the password if it is not given.

```
$ sdesetupora8i  -o install
Password:
```

UNIX users should be careful with special characters such as "?". Depending on the UNIX shell version, you may need to use the appropriate quote character to use a special character. For example, you can use the "-?" operation with any administration command to get help. If you're using a C shell (rather than a Bourne shell), you must use "\", which tells the shell to use the next character directly rather than as a special character.

Therefore, in a C shell, you must use:

```
$ sdemon  -\?
```

In a Bourne shell, you can simply use:

```
$ sdemon  -?
```

Some commands include optional SQL query statements, or "where clauses", to limit the features retrieved from a table or logfile. If this option is included, the query must be quoted (for example, "area < 1000"). If your DBMS encloses character literals with single quotes, enclose the entire expression with double quotes (for example, "state_code = 'CO' "). If your DBMS encloses character literals with double quotes, enclose the entire expression with single quotes, for example, 'state_code = "CO"'.

Getting help

You can list the usage, operations, and options for any ArcSDE administration command with the “-h” or “-?” options.

sdemon -h

```
sdemon -o status {[-i <service>]
       [-s <server_name>] |
       [-H <sde_directory>]} [-q]

sdemon -o start {[-i <service>]
       [-s <server_name>] |
       [-H <sde_directory>]}
       [-p <DB_Admin_password>]

sdemon -o shutdown {[-i <service>]
       [-s <server_name>] | [-H <sde_directory>]}
       [-p <DB_Admin_password>] [-N]

sdemon -o pause {[-i <service>]
       [-s <server_name>] | [-H <sde_directory>]}
       [-p <DB_Admin_password>]

sdemon -o resume {[-i <service>]
       [-s <server_name>] | [-H <sde_directory>]}
       [-p <DB_Admin_password>]

sdemon -o info -I {users | config | stats |
       locks | vars }  [-q]
       {[-i <service>] [-s <server_name>] |
       [-H <sde_directory>]}

sdemon -o kill {[-i <service>] [-s <server_name>]
       | [-H <sde_directory>]}
       [-p <DB_Admin_password>] -t { all | <pid> }
       [-N]

Operations:

       status          Report instance status.
       start           Start the instance.
       shutdown        Shut down the instance.
       pause           Stop the instance from
                        accepting further
                        connections.
       resume           Let the instance accept
                        connections again.
       info             Display requested
                        information about the
                        instance.
       kill             Kill all or a specified
                        connection to the
instance.

Options:

       -o    Operation
       -s    Specify Server other than localhost
       -t    Kill target, either the pid of an
             ArcSDE server, or "all"
       -p    ArcSDE DBA DBMS password
       -H    ArcSDE home directory for instance to
             operate on
       -i    ArcSDE instance name to operate on
             (not allowed for start)
       -I    The requested information type
       -q    Quiet
       -N    No Verification
       -?    Print Options
       -h    Print Options
```

ArcSDE administration commands: sdemon

This command is used to monitor and manage the ArcSDE application server. You can use sdemon to start up, pause, resume, and shut down all connection activity and display current configuration parameters and server task information. Individual server tasks can also be managed.

Security

Current execution privileges are granted to the root or the ArcSDE administrator for the start operation only. All other operations may be executed by any user who knows the password.

Usage syntax

```
sdemon    -o status {[-i <service>]
          [-s <server_name>] |
          [-H <sde_directory>]} [-q]
sdemon    -o start [-p <DB_Admin_password>]
          {[-i <service>] [-s <server_name>] |
          [-H <sde_directory>]}
sdemon    -o shutdown [-p <DB_Admin_password>]
          [-i <service>] [-s <server_name>] | [-H
          <sde_directory>]} [-N]
sdemon    -o pause [-p <DB_Admin_password>]
          {[-i <service>] [-s <server_name>] | [-
          H <sde_directory>]}
sdemon    -o resume [-p <DB_Admin_password>]
          {[-i <service>] [-s <server_name>] | [-
          H <sde_directory>]}
sdemon    -o info -I {users | config | stats |
          locks | vars} {[-i <service>]  [-s
          <server_name>] | [-H <sde_directory>]}
          [-q]
sdemon    -o kill [-p <DB_Admin_password>]
          -t { all | <pid> } {[-i <service>] [-s
          <server_name>] | [-H <sde_directory>]}
          [-N]
sdemon    -h
sdemon    -?
```

Operations

status	Reports the service status.
start	Starts the ArcSDE application server if it's not running. Only the ArcSDE administrator can use this operation.
shutdown	Shuts down the ArcSDE application server immediately if no server sessions are running. If server sessions are running, you're prompted to remove the running sessions before the shutdown takes place. Using the "-N" option shuts down all server sessions and the application server immediately.
pause	Disallows further connection requests to be processed. Currently connected sessions are allowed to continue.
resume	Allows connection requests to be processed again.
info	Displays information about users, configuration, statistics, locks, or environment variables.
kill	Kills all or a specified application server session.

Options

-h	Prints usage and options.
-H	ArcSDE home directory (SDEHOME).
-i	ArcSDE service name (not applicable for "start" option).
-I	Inquire about configuration, locks, statistics, users, or environment variables. config Displays current configuration variables. locks Displays lock information about processes that are holding locks. stats Displays process statistics for each ArcSDE client/server connection. users Lists users' connections to ArcSDE and associated process identifiers. vars Displays ArcSDE application server environment variables.
-N	No verification is performed—the operation begins immediately after being invoked.
-o	Operation.
-p	ArcSDE administrator DBMS password.
-q	Quiet; all titles and warnings are suppressed.
-s	ArcSDE application server hostname (default: localhost).
-t	Kills server tasks: all Forcefully removes all server tasks. pid Removes the task identified by the process identifier.
-?	Prints usage and options (use "-\?" in C shell).

ArcSDE administration commands: sdeconfig

This command manages the ArcSDE application server configuration parameters. Using this command you can edit the parameters stored in the SDE.SERVER_CONFIG table. You can update the entire table by overwriting it with the contents of a configuration file, or edit individual records. The default configuration file is the giomgr.defs file found under the etc folder in the SDEHOME.

Security

All operations may be executed by any user who knows the password.

Usage syntax

```
sdeconfig -o import -f <SERVER_Info file>
          [-i <service>] [-s <server_name>]
          [-D <database>] [-u <DB_User_name>]
          [-p <DB_User_password>] [-N] [-q]
sdeconfig -o export -f <SERVER_Info file>
          [-i <service>] [-s <server_name>]
          [-D <database>] [-u <DB_User_name>]
          [-p <DB_User_password>] [-N] [-q]
sdeconfig -o list [-P <Property_Name>]
          [-i <service>] [-s <server_name>]
          [-D <database>] [-u <DB_User_name>]
          [-p <DB_User_password>] [-N] [-q]
sdeconfig -o alter -v <Property_Name=Value,...>
          [-i <service>] [-s <server_name>]
          [-D <database>] [-u <DB_User_name>]
          [-p <DB_User_password>] [-N] [-q]
sdeconfig -h
sdeconfig -?
```

Operations

import	Imports the contents of a server configuration file into the SDE.SERVER_CONFIG table. The default file that is used during setup is the giomgr.defs file, found under the etc folder in the SDEHOME.
export	Exports the contents of the SDE.SERVER_CONFIG table to a server configuration file for editing and eventual reimportation..
list	List the server configuration parameters stored in the SDE.SERVER_CONFIG table.
alter	Changes the value of a server configuration parmeter in the SDE.SERVER_CONFIG table.

Options

-D	Database or data source name. Not supported by all DBMS
-f	Server coniguration file
-h	Prints usage and options.
-i	ArcSDE service name.
-N	No verification is performed—the operation begins immediately after being invoked.
-p	ArcSDE administrator DBMS password.
-P	The name of the configuration parameter
-q	Quiet; all titles and warnings are suppressed.

-s	ArcSDE application server hostname (default: localhost).
-u	ArcSDE user.
-v	The configuration parameter to be changed and its new value entered as <parameter=value>. For example, to increase the total number of connections that an ArcSDE server will allow from the default 48 to 60 the -v option would be entered as: -v connections=60

ArcSDE administration commands: sdeservice (Windows only)

This command manages the ArcSDE service on Windows.

Usage syntax

```
sdeservice -h

sdeservice -o create -p <DB_Admin_password>
           [-n] [-H <sde_directory>]
           [-d <ORACLE8I,SID |ORACLE9I,SID |
           SQLSERVER,SQLSERVERINSTANCE |
           DB2,DB2INSTANCE | INFORMIX |
           COVERAGES>] [-i <service>]
           [-u <service_user>]
           [-P <service_user_password>]

sdeservice -o delete [-d <ORACLE8I|
          ORACLE9I|SQLSERVER|DB2|INFORMIX|
          COVERAGES>] [-i <service>] [-q] [-N]

sdeservice -o register -r <registry_keyword>
          -v <value>  -d <ORACLE8I|
          ORACLE9I|SQLSERVER|DB2|INFORMIX|
          COVERAGES> -p <SDE_DBA__password>
          [-i <service>]

sdeservice -o unregister -r <registry_keyword>
          -d <ORACLE8I|ORACLE9I|SQLSERVER|
          DB2|INFORMIX|COVERAGES>
          -p <SDE_DB_Admin_password>
          [-i <service>]

sdeservice -o modify -r <registry_keyword>
          -v <new_value> -d <ORACLE8I|
          ORACLE9I|SQLSERVER|DB2|
          INFORMIX|COVERAGES> [-q]
          -p <SDE_DBA_password> [-i <service>]

sdeservice -o list [-i <service>]
```

Operations

create	Creates a service
delete	Deletes a service
register	Registers a service
unregister	Unregisters a service
modify	Modifies a service
list	Displays service information for all or a specified service

Options

-d	A DBMS whose service should start before ArcSDE. Optional if the DBMS is on a remote machine. This must be in uppercase.
-h	Prints usage and options.
-H	ArcSDE home directory (SDEHOME). Only needed if the SDEHOME variable isn't set or multiple services are in use.
-i	ArcSDE service name—only required if the service is not called "esri_sde".
-N	No verification is performed—the operation begins immediately after being invoked.
-o	Operation.
-p	ArcSDE administrator DBMS password.
-P	ArcSDE application server user password (Windows login password).
-q	Quiet; all titles and warnings are suppressed.
-r	Windows registry keyword.

Options	(continued)
-u	ArcSDE service account user—must be a Windows user who has administrator permissions on the server computer. Include the domain name if needed. For example, if you're logged into the "AVWORLD" domain and your username is "joe", enter "AVWORLD\joe". You should be logged in as this user when you create the service.
-v	Registry value.
-?	Prints usage and options (use "-\?" on C shell).

Discussion

The sdeservice administration utility manages the ArcSDE services and registry entries on Windows platforms. Installing the ArcSDE software adds one service and several related registry entries, which include SDE_DBA_PASSWORD, SDEHOME, LICENSE_SERVER, and NLS_LANG.

The create and delete operations will add or delete the ArcSDE service entry, respectively. You can modify the registry values of SDEHOME, SDE_DBA_PASSWORD, LICENSE_SERVER, or NLS_LANG with the modify operation. You can also remove or re-create the SDEHOME or SDE_DBA_PASSWORD entries with the unregister and register operations. Normally, you'll only need to use the register operation after using unregister.

ArcSDE administration commands: sdesetup

This command administers business tables and their data.

Usage syntax

```
sdesetup* -h
sdesetup* -?
sdesetup* -0 upgrade [-H <sde_directory>]
          [-p <DB_Admin_password>] [-N] [-q]

sdesetup* -o list [-H <sde_directory>]
          [-p <DB_Admin_password>] [-q]

sdesetup* -o install [-H <sde_directory>]
          [-p <DB_Admin_password>] [-N] [-q]
```

Operations

install	Creates or updates the ArcSDE and Geodatabase system tables and stored procedures.
list	Lists the installed ArcSDE version.
upgrade	Upgrades the ArcSDE and Geodatabase system tables and stored procedures.

Options

-h	Prints usage and options.
-H	ArcSDE home directory (SDEHOME). Only needed if the SDEHOME variable isn't set or multiple services are in use.
-N	No verification is performed—the operation begins immediately after being invoked.
-o	Operation.
-p	ArcSDE administrator DBMS password.
-q	Quiet—all titles and warnings are suppressed.
-?	Prints usage and options (use "-\?" on C shell).

Discussion

The sdesetup administration utility manages the creation and maintenance of the ArcSDE and geodatabase metadata, which primarily includes system tables, indexes, stored procedures, and some triggers. The actual name of the command varies according to the implementation of ArcSDE as follows:

ArcSDE for Oracle8*i*	sdesetupora8i
ArcSDE for Oracle9*i*	sdesetupora9i
ArcSDE for Informix	sdesetupinfx
ArcSDE for SQL Server	sdesetupmssql
ArcSDE for DB2 UDB	sdesetupdb2

The ArcSDE system tables and stored procedures can be created or updated using the install operation. For new installations of ArcSDE, the install operation will create the tables and stored procedures for the first time. For subsequent executions of the

install operation on an existing database, the metadata is checked to ensure that it is current. If it is not, the install operation will bring it up-to-date.

sdesetupora9i -o install -p bugaboo

Use the update operation to bring an existing database up-to-date with the latest additions or changes to a new ArcSDE installation.

sdesetupmssql -o upgrade -p bugaboo

The list operation returns the current version of an ArcSDE installation.

sdesetupinfx -o list -p bugaboo

ArcSDE initialization parameters

D

IN THIS APPENDIX

- **ArcSDE service initialization parameters**

The ArcSDE application server initialization parameters, stored in the SDE.SERVER_CONFIG table, may be adjusted to control the resources allocated to each ArcSDE session.

The operation of and default settings for each parameter are described in this appendix.

Parameter	Description
ALLOWSESSIONLOGFILE	Set this parameter to TRUE if you want your users to use nonstandard session-based ArcSDE logfiles. Session-based logfiles are optimized for applications that remain connected to ArcSDE over an extended period of time and that select many records.
ATTRBUFSIZE	The size of the attribute array buffer. Defaults to 50,000.
AUTOCOMMIT	The implicit ArcSDE user automatic commit rate within a transaction. If AUTOCOMMIT is set to zero, the transaction will commit only if the application issues an explicit commit. If it is set to a number greater than zero, the operation will commit after the number of updates specified by AUTOCOMMIT have occurred. This feature prevents transactions from becoming too large and exceeding the DBMS logs. Defaults to 1,000.
BLOBBUFSIZE	Obsolete at ArcSDE 8.0.2.
BLOBMEM	When binary large objects are stored, the server must accumulate the BLOB chunks the application sends over the network. If the BLOB size is greater than BLOBMEM, the server writes the BLOB data to a disk file before storing it in the database. If the BLOB size is less than BLOBMEM, the server accumulates the BLOB in memory. If BLOBMEM is a negative number, the server always uses memory, regardless of the BLOB size. Defaults to 1,000,000.
CONNECTIONS	The maximum number of simultaneous connections accepted by the application server. This parameter does not limit the number of direct connections. Defaults to 48.
DEFAULTPRECISION	By default the ArcSDE stores geometry data as 32-bit integers. For most applications this precision is sufficient. However, if your application requires greater precision, you may set this parameter to 64. Spatial columns created while this parameter is set this way will store 64-bit geometry.
DETECT8XNULLSHAPE	Set this parameter to TRUE if you are using ArcSDE for Oracle8*i* and you expect that at least some of your feature classes will contain NULL shapes.

Parameter	Description (continued)
DISABLEAUTOREG	Disables the autoregistration of business tables containing spatial columns created with an object relational datatype. The DB2 Spatial Extender, Informix Spatial DataBlade, and Oracle Spatial all provide object relational spatial types. When the ArcSDE application server is started, it scans the system tables for any spatial columns that may exist in the database not already registered, then registers them. Setting DISABLEAUTOREG to FALSE disables this capability.
DISSMEM	Obsolete at ArcSDE 8.0.2.
ERRLOGMODE	Determines whether the time stamp, session ID, and client computer name are logged with each error when an error is written to the ArcSDE error logfile. By default this parameter is set to TIC. T—time stamp I—Session ID C—Client computer If you wish to limit the items included with the error, set the parameter to the letters of those items. If you wish to include none of these items with the error, set the parameter to NONE.
HOLDLOGPOOLTABLES	By default this parameter is set to TRUE, which directs the ArcSDE application to keep and reuse all session-based logfiles that it has checked out of the session logfile pool until it disconnects. When set to FALSE, ArcSDE applications release logfiles back to the pools whenever they delete the logfile.
INT64TYPES	Set this parameter to FALSE if you want to disable 64-bit integers, as you will need to do if your application does not support them. Set this parameter to TRUE if you want to use 64-bit integers. When set to FALSE, ArcSDE will return 64-bit integer fields as double precision and disallow the creation of 64-bit integer fields.
LAYERAUTOLOCKING	As of ArcSDE 9, layer autolocking has been disabled by default. You can enable it for the ArcSDE server by setting the server configuration parameter LAYERAUTOLOCKING to TRUE, in which case layers that have their autolocking property enabled will autolock shapes when they are edited in NORMAL_IO mode. By default, a layer is created with its autolocking property enabled. To disable autolocking for a particular layer, use the sdelayer administration command's alter operation.

Parameter	Description (continued)
LARGEIDBLOCK	The value of this parameter is used in the calculation of the number of row ID values ArcSDE allocates to a buffered stream. When ArcSDE detects that a buffered stream is attempting to load a large number of records into a table, it calculates the number of row IDs to be allocated as the minimum of this parameter and twice the current row ID allocation. The row ID allocation is initially set to the value of the SMALLIDBLOCK and doubles in size until it reaches LARGEIDBLOCK.
LAYERS	Obsolete at ArcSDE 8.1 for Oracle8*i* and SQL Server. Obsolete at ArcSDE 9 for DB2 and Informix.
LOCKS	Obsolete at ArcSDE 8.1 for Oracle8*i* and SQL Server. Obsolete at ArcSDE 9 for DB2 and Informix.
LOGS	Obsolete at ArcSDE 8.0.2.
MAXARRAYBYTES	The maximum number of array bytes allocated per stream. Defaults to 550,000.
MAXARRAYSIZE	The maximum array fetch size. Defaults to 100.

Parameter	Description (continued)
MAXBLOBSIZE	The maximum size of user-defined BLOBs in bytes. Defaults to 1,000,000.
MAXBUFSIZE	The maximum buffer threshold. The minimum value is 12,288. If the MAXBUFSIZE value is greater than the minimum value but less than the MINBUFSIZE, the two values are switched. Defaults to 65,536.
MAXDISTINCT	This parameter controls the maximum number of distinct values returned by an SE_DISTINCT_STATS statistic in a call to SE_table_calculate_stats or SE_stream_calculate_table_statistics. A value of zero means an unlimited number of distinct values can be returned. Defaults to 512.
MAXINITIALFEATS	The maximum number of features allowed in the initial features argument of the sdelayer administration tool and the SE_layer_create function. This parameter prevents the inadvertent creation of excessively large initial extents for the table of a feature class. This is an ArcSDE for Oracle parameter only. Defaults to 10,000.
MAXSTANDALONELOGS	The maximum number of standalone logfiles a user is allowed to create. Defaults to 0.

Parameter	Description (continued)
MAXSTREAMS	Obsolete at ArcSDE 9.
MAXTABLELOCKS	Obsolete at ArcSDE 8.1 for Oracle8*i* and SQL Server. Obsolete at ArcSDE 9 for DB2 and Informix.
MAXTIMEDIFF	Specified in seconds, the maximum time difference allowed between the application server machine and a client machine. Set this parameter to -1 to disable it. Defaults to 1,800. This parameter does not restrict direct connections.
MINBUFOBJECTS	The minimum number of buffer objects. Defaults to 512.
MINBUFSIZE	The minimum buffer threshold. The minimum value is 4,096. Defaults to 16,384.
OBJECTLOCKS	Obsolete at ArcSDE 8.1 for Oracle8*i* and SQL Server. Obsolete at ArcSDE 9 for DB2 and Informix.
PACKETSIZE	This parameter applies to ArcSDE for SQL Server only. SQL Server allows you to change the network packet size of the packets transferred between the SQL Server instance and the client (in this case ArcSDE). Testing at ESRI has determined that ArcSDE performs best when this packet size is set to 8 KB, the default setting for this parameter.

Parameter	Description (continued)
PRECISION10	By default this parameter is set to FALSE. For ArcSDE for Oracle, setting this parameter to TRUE increases the precision of the interpretation of the LONG integer value from 9 bytes to 10 bytes. At ArcSDE 8.1 the LONG integer was shortened to 9 bytes as a bug fix. However, this affected the storage of legacy data, which requires the LONG to be interpreted as 10 bytes. Unless you are having a problem with legacy data created prior to ArcSDE 8.1, do not change the setting of this parameter.
PROCSTATS	The process statistics parameter controls the interval by which sessions update the SDE.PROCESS_INFORMATION table. By default, this parameter is set to -1, which disables entries to the PROCESS_INFORMATION table. To enable entries, set the parameter to a positive integer that reflects the interval in seconds in which the sessions will write their statistics to the table. The interval represents the time that must pass before the session last wrote statistics. The session will only write statistics if a change has occurred. Writing statistics to the SDE.PROCESS_INFORMATION can be expensive and should be avoided.

Parameter	Description (continued)
RASTERBUFSIZE	The size of the raster buffer specified in bytes. This value must be large enough to store the largest raster tile accessed. Defaults to 102,400.
RASTERCOLUMNS	Obsolete at ArcSDE 8.1 for Oracle8*i* and SQL Server. Obsolete at ArcSDE 9 for DB2 and Informix.
READONLY	When set to TRUE, the ArcSDE service will not allow edits to be performed by ArcSDE clients. The default setting of FALSE allows editing.
REGISTRATIONS	Obsolete at ArcSDE 8.1 for Oracle8*i* and SQL Server. Obsolete at ArcSDE 9 for DB2 and Informix.
SERVER_CON_INFO	The optional path to the keycode file. Direct connections use this parameter to locate the file containing keycodes.
SESSIONLOGPOOLSIZE	The size of the session-based logfile pool. By default this parameter is set to 0, which represents the number of logfiles that will be created in the sde user's schema and shared among users. A user may check out a logfile from the pool as opposed to creating their own.

Parameter	Description (continued)
SHAPEPTSBUFSIZE	The size of the shape POINTS array buffer. The default value is 400,000, which is calculated for a two-dimensional area feature with 500 points.
SHAPEBUFSIZE	Obsolete at ArcSDE 8.0.2.
SMALLIDBLOCK	The value of the parameter is used to calculate the minimum number of row IDs that ArcSDE allocates to the stream. Defaults to 16.
SPINDEXBUFSIZE	Obsolete at ArcSDE 8.0.2
STATEAUTOLOCKING	By default this parameter is set to FALSE to disable the autolocking of the states. Unless you have non-ESRI applications that are editing states, you should not set this parameter to TRUE, as it will degrade the overall performance of your system.
STATELOCKS	Obsolete at ArcSDE 8.1 for Oracle8*i* and SQL Server. Obsolete at ArcSDE 9 for DB2 and Informix.

Parameter	Description (continued)
STREAMPOOLSIZE	The maximum number of allocated stream resources added to the stream pool. Until this value is exceeded, the resources of released streams are not deallocated but are added to the stream pool. The resources of the stream pool are reused whenever new streams are created. If the stream pool is full when a stream is released, its resources are deallocated. Defaults to 6.
TCPKEEPALIVE	Setting TCPKEEPALIVE to TRUE will allow the ArcSDE application server to use the current system's TCP/IP KEEPALIVE settings. ArcSDE application servers will then be able to detect clients whose machines have crashed or have been deliberately terminated by the Windows Task Manager or the UNIX kill command. TCPKEEPALIVE set to TRUE turns on a form of probing where, after two hours of idle time (usually), a packet is sent on the idle connection to see if there is anything on the other end. Be aware that if TCPKEEPALIVE is set to TRUE, a disconnection can be triggered by short-term network outages (~10 minutes). By default, TCPKEEPALIVE is set to FALSE.

Parameter	Description (continued)
	TCPKEEPALIVE will not disconnect a direct connection.
TEMP	The temporary file directory. Defaults to /tmp.
TIMEOUT	Obsolete at ArcSDE 8.0.2.
TLMINTERVAL	The ArcSDE application server caches information about the spatial columns, raster columns, registered tables, and XML columns. ArcSDE queries the SDE.SDE_TABLES_MODIFIED to determine if these tables have been modified. The time last modified interval is the number of seconds that must pass before the SDE.SDE_TABLES_MODIFIED is queried again to determine if a change has been made to an ArcSDE object. By default this value is set to 1 second. If the ArcSDE metadata is not changing (i.e., you are not adding, removing, or altering layers, raster columns, or XML columns or changing the registration of business tables), you can set this parameter higher to avoid the overhead of queries to the SDE.SDE_TABLES_MODIFIED.

Glossary

Abstract Data Types (ADT)

Spatial data types defined by the Open GIS Consortium (OGC) and documented in the OpenGIS RFP-1 (SimpleFeatures) document. They include point, line string, polygon, multipoint, multilinestring, multipolygon, and geometry. This provides a standard feature-based abstraction of real-world phenomenon. This vector data consists of geometric and topological primitives used, separately or in combination, to construct objects that express the spatial characteristics of geographic features.

ANSI SQL 89

Industry-standard language used in querying, updating, and managing relational databases. SQL can be used to retrieve, sort, and filter specific data to be extracted from the database. American National Standards Institute (ANSI) adopted SQL as the standard language for database management systems. See also DBMS.

API

Application programming interface. Refers to a defined and documented set of tools or functions that application developers use to build or customize a program or set of programs. APIs can be built for programming languages such as C, COM, Java™, and so on.

ArcCatalog

Spatial data browser application that facilitates data management and data access.

ArcIMS

Application for creating, designing, and managing Web sites with mapping and geographic information system capabilities.

ArcMap

Spatial data display, editing, query, and analysis application.

ArcSDE

ArcSDE is a gateway to a multiuser commercial DBMS—for example, Oracle, Microsoft SQL Server, Informix, and DB2. ArcSDE is an open, high-performance spatial data server that employs client/server architecture to perform efficient spatial operations and manage large, shared geographic data. See also client/server.

ArcSDE data types

ArcSDE implementation of native DBMS data types: smallint (small integer), integer, float, double, string, BLOB, date.

ArcSDE for Coverages

An ArcSDE application server that provides read-only access to ArcInfo coverages, shapefiles, ArcStorm™ library layers, and ArcInfo Librarian™ layers. Uses the same data transfer technology as ArcSDE for DBMS products.

ArcStorm

ArcStorm (Arc Storage Manager) is a data storage facility and transaction manager for ArcInfo file-based coverage data. ArcStorm manages a feature-oriented database that can be closely integrated with a DBMS. An ArcStorm database is a collection of libraries, layers, INFO tables, and external DBMS tables. Data stored in an ArcStorm database benefits from the transaction management and data archiving capabilities of ArcStorm.

ArcView

Easy-to-use desktop GIS for exploring, analyzing, displaying, and querying geographic data.

attribute

1. A characteristic of a geographic feature described by numbers, characters, images, and computer-aided design (CAD) drawings, typically stored in tabular format and linked to the feature. For example, the attributes of a well might include depth and gallons per minute.
2. A column in a database table. See also column.

BLOB

Binary large object. The binary data type of a column in a DBMS table that stores large image, text, or geometry data as attributes.

CAD

Computer-aided design. An automated system for the design, drafting, and display of graphically oriented information.

clean

An ArcInfo command that builds coverage polygon and arc–node topology by performing a geometric analysis on arcs and label points to identify coverage nodes and polygons.

client/server

A software system is said to have a client/server architecture when there is a central process (server) that accepts requests from multiple user processes (clients). The architecture enables the separation of local client processing from the server that manages the databases, access, and data integrity. ArcSDE is one example of a client/server architecture.

column

The vertical dimension of a table that holds attribute values. A column has a name and a data type applied to all values in the column. A table has rows and columns. See also row and table.

coordinate

A set of numbers that designate location in a given reference system such as x,y in a planar coordinate system or x,y,z in a three-dimensional coordinate system. Coordinates represent locations on the earth's surface relative to other locations.

coordinate system

1. A reference system used to measure horizontal and vertical distances on a planimetric map. A coordinate system is usually defined by a map projection, a spheroid of reference, a datum, one or more standard parallels, a central meridian, and possible shifts in the x- and y-directions to locate x,y positions of point, line, and area features.

2. In ArcInfo, a system with units and characteristics defined by a map projection. A common coordinate system is used to spatially register geographic data for the same area. See also spatial reference.

coverage

A file-based vector data storage format for storing the location, shape, and attributes of geographic features. A coverage stores geography as primary features (such as arcs, nodes, polygons, and label points) and secondary features (such as tics, map extent, links, and annotation). Associated feature attribute tables describe and store attributes of the geographic features.

data dictionary

A catalog of all data held in a database or a list of items giving data names and structures. Also referred to as DD/D for data dictionary/directory. Commercial DBMSs have online data dictionaries stored in special tables called system tables. ArcSDE and the geodatabase have data dictionary tables containing information about the spatial data in the database.

data integrity

Maintenance of data values according to data model and data type. For example, to maintain integrity, numeric columns will not accept character data.

data type

The characteristic of columns that defines what types of data values they can store. Examples include character, floating point, and integer.

database

1. A collection of related data organized for efficient retrieval of information.
2. A logical collection of interrelated information managed and stored as a unit, usually on some form of mass storage system such as magnetic tape or disk. A GIS database includes data about the spatial location and shape of geographic features recorded as points, lines, areas, pixels, grid cells, or TINs, as well as their attributes.

database administrator (DBA)

One who manages a database—sets up users, security, backup, and recovery procedures for all data and optimizes physical data storage for best performance.

database connection

A connection to a DBMS server, an ArcSDE application server, or an Object Linking and Embedding Database.

database locking

A database management process for maintaining the consistency of the data while supporting simultaneous access by more than one user. A typical technique is to use a system of locking data to prevent data corruption caused by multiple users editing and reading it.

database trigger

Triggers are stored procedures associated with a specific operation on a specific database table. A trigger is automatically fired when the operation with which it is associated is performed on the table. For example, a trigger may be used in conjunction with a database sequence to enforce a primary key constraint. For every insert operation, the trigger ensures that a new sequence number is assigned to the table as a primary key.

dataset

A named collection of logically related data items arranged in a prescribed manner.

DB2 Spatial Extender

One of IBM's DB2 Universal Database Extenders that defines new data types and functions to support the storage and manipulation of spatial data.

DBMS

Database management system. A set of computer programs for organizing the information in a database. A DBMS supports the structuring of the database in a standard format and provides tools for data input, verification, storage, retrieval, query, and manipulation. See also RDBMS.

domain

A named constraint in the database. This named constraint can be associated with a field for the subtype of a feature class or table to make an attribute validation rule. Types of attribute domains include range and coded value domains.

dynamic link library (DLL)

A precompiled library file that contains functions and data. A DLL is loaded at run time by its calling modules (EXE or DLL). Also referred to as an in-process server.

entity

A collection of objects (persons, places, things) described by the same attributes. Entities are identified during the conceptual design phase of database and application design.

executable file (.exe)

A program file created from one or more source code files translated into machine code and linked together. Also referred to as an out-of-process server.

export

A platform-independent data transfer utility—generates a binary interchange file.

feature

1. A vector object in a geodatabase that has a geometry type of point, line, polygon, or annotation. Features are stored in feature classes.
2. A representation of a real-world object in a layer on a map.
3. A point, line, or polygon in a coverage or shapefile.

feature class

1. The conceptual representation of a geographic feature. When referring to geographic features, feature classes include point, line, area, and annotation. In a geodatabase, an object class that stores features and has a field of type geometry. See also layer.
2. A classification describing the format of geographic features and supporting data in a coverage. Coverage feature classes for representing geographic features include point, arc, node, route–system, route, section, polygon, and region. One or more coverage features are used to model geographic features; for example, arcs and nodes can be used to model linear features such as street centerlines. The tic, annotation, link, and boundary feature classes provide supporting data for coverage data management and viewing.
3. The collection of all the point, line, or polygon features or annotation in a CAD dataset.

feature dataset

In geodatabases, a collection of feature classes that share the same spatial reference. Because the feature classes share the same spatial reference, they can participate in topological relationships with each other such as in a geometric network. Several feature classes with the same geometry may be stored in the same feature dataset. Object classes and relationship classes can also be stored in a feature dataset.

field

The intersection of a table row and a column—each field contains the values for a single attribute. See also attribute.

firewall

A combination of filters and gateways that protect a site's computers from external unauthorized access or outright attack.

geodatabase

An object-oriented geographic database, or collection of spatial data and related descriptive data, organized for efficient storage and retrieval by many users and hosted inside a DBMS.

Object behavior is implemented using validation rules, relationships, and topological associations.

geographic data

The locations and descriptions of geographic features—the composite of spatial data and descriptive data.

geographic database

A collection of spatial data and related descriptive data organized for efficient storage and retrieval by many users.

geographic information system

An organized collection of computer hardware, software, geographic data, and personnel designed to efficiently capture, store, update, manipulate, analyze, and display all forms of geographically referenced information.

geolocation

The process of creating features from tabular data by matching the tabular data to a spatial location. One example of geolocation is creating point features from a table of x,y coordinates. Points can also be created by matching addresses to streets.

geometric network

1. A one-dimensional nonplanar graph, or logical network, that is composed of features. These features are constrained to exist within the network and can, therefore, be considered network features. ArcInfo 8 will automatically maintain the explicit topological relationships between network features in a geometric network.
2. Represents a one-dimensional linear network such as a road system, a utility network, or a hydrologic network. Geometric networks contain feature classes that play a topological role in the network. These feature classes are homogeneous collections of one of these four network feature types: simple junction feature, complex junction feature, simple edge feature, and complex edge feature. More than one feature class can have the same type of network feature.

geometry

The properties, measurements, and relationships of points, lines, angles, surfaces, and solids. In ArcInfo, geometry is used to represent the spatial component of geographic features.

grid

A geographic data model representing information as an array of equally sized square cells arranged in rows and columns. Each grid cell is referenced by its geographic x,y location.

image

Represents geographic features by dividing the world into discrete squares called cells. Examples include satellite and aerial photographs, scanned documents, and building photographs.

import

A platform-independent data transfer utility—reads data into an ArcSDE database from a binary interchange file.

index

Special data structure used in a database to facilitate searching for records in tables or spatial features in geographic datasets. Provides faster access to data than doing a full table scan. ArcInfo supports both spatial and attribute indexes.

Informix

A commercial DBMS supported by ArcSDE.

intercept

An ArcSDE facility to capture the TCP/IP-based communication that is passed to and from ArcSDE clients and ArcSDE servers.

layer

1. A collection of similar geographic features—such as rivers, lakes, counties, or cities—of a particular area or place for display on a map. A layer references geographic data stored in a data source, such as a coverage, and defines how to display it. You can create and manage layers as you would any other type of data in your database.
2. A feature class in a shared geodatabase managed with ArcSDE. See also feature class.

License Manager

Software that allocates and monitors ArcInfo licenses and their usage.

line

A line connects two or more x,y coordinates. Rivers, roads, and electric and telecommunication networks are all linear features.

local area network

A computer data communications technology that connects computers at the same site—for example, all computers in the same building. Computers and terminals on an LAN can freely share data and peripheral devices such as printers and plotters. LANs are composed of cabling hardware and software.

Map LIBRARIAN

A set of software tools to manage and access large geographic datasets in a map library. LIBRARIAN commands create and define a map library, move data in and out of a library, query the data in a map library, and display the results of a query.

MapObjects

A collection of software "building blocks" that can be used by developers to create applications that include GIS and mapping capabilities.

Microsoft Access

A commercial DBMS supported by ArcSDE.

Microsoft SQL Server

A commercial DBMS that is supported with ArcSDE.

network packet

Requests and results transferred between clients and servers are sent in fixed-size chunks of data known as "packets". If an application does bulk-copy operations or sends or receives large amounts of text or image data, a packet size larger than the default may improve efficiency because it results in fewer network reads and writes. If an application sends and receives small amounts of information, setting the packet size to 512 bytes would be sufficient for most data transfers.

OLE DB provider

Object linking and embedding database provider. OLE DB is an open specification designed to build on the success of ODBC by providing an open standard for accessing all kinds of data. Each provider communicates with and retrieves data from a different database, but you can work with data retrieved by any provider the same way. Typically, they can only retrieve nonspatial data. However, if an OLE DB provider can retrieve geographic data in OpenGIS format, you can work with that data in ArcInfo.

OpenGIS

Open Geodata Interoperability Specification—a series of standards being developed by the Open GIS Consortium to support interoperability of GIS systems in a heterogeneous computing environment.

Oracle

A commercial DBMS supported by ArcSDE.

Oracle Spatial Geometry Type

Geometries are stored using a table with a single column of type MDSYS.SDO_GEOMETRY and a single row per geometry instance.

paging

At the operating system level, this process transfers information from volatile system memory (RAM) to disk and back again. This enables the system to handle more information than it normally could handle in real memory.

point

A single x,y coordinate that represents a single geographic feature such as a telephone pole.

polygon

A two-dimensional feature representing an area such as a state or county.

port number

An entry/exit mechanism that controls and synchronizes the flow of data into and out of the central processing unit (CPU) to external devices. Ports, and their associated numbers, are used in TCP/IP to distinguish between different types of communication between TCP/IP addresses. Communication can be established at different ports to keep conversations separate. Some ports are assigned to particular types of communication. For example, port 80 is for HTTP, and port 443 is for Secure HTTP. The default port number for ArcSDE client/server communication is 5151.

RDBMS

Relational database management system. A database management system with the ability to access data organized in tabular files that can be related to each other by a common field (item). An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage. ArcSDE supports several commercial RDBMSs. See also DBMS.

regions

A coverage feature class used to represent a spatial feature as one or more polygons. Many regions can be defined in a single coverage. Regions have attributes that describe the geographic feature they represent. See also coverage.

relationship

An association or link between two objects. See also relationship class.

relationship class

Objects in a real-world system often have particular associations with other objects in the database. These kinds of associations between objects in the geodatabase are called relationships. Relationships can exist between spatial objects (features in feature classes), nonspatial objects (rows in a table), or spatial and nonspatial objects. While spatial objects are stored in the geodatabase in feature classes, and nonspatial objects are stored in object classes, relationships are stored in relationship classes.

rollback

Changes (updates, inserts, and deletes) made to the information stored in a database form part of a transaction. Until the user, or application, explicitly acts to make these changes permanent, the changes are pending. Issuing a ROLLBACK command will reverse all pending modifications to restore the database to the state it was prior to the beginning of the transaction. See also transaction.

row

A record in an attribute table—the horizontal dimension of a table composed of a set of columns containing one data item each. Also referred to as a tuple.

A horizontal group of cells in a grid or pixels in an image.

schema

1. The structure or design of a database or database object such as a table.
2. The definition of the database. The schema can either be modeled in UML using a CASE tool or defined directly within ArcCatalog using wizard dialog boxes.

SDE®

See ArcSDE.

server

A computer on which a service process runs. See also service.

service

A computer program that receives a request from a client, processes it to generate results, and returns those results to the client.

shape

The characteristic appearance or visible form of a geographic object. Geographic objects can be represented on a map using one of three basic shapes: points, lines, or polygons.

shapefile

A vector data storage format for storing the location, shape, and attributes of geographic features.

Spatial DataBlade

A database enhancement module introduced by Informix to extend the functionality of the INFORMIX-Universal Server to provide support for spatial data management.

spatial reference

Describes both the projection and spatial domain extent for a feature dataset or feature class in a geodatabase. See also coordinate system.

SQL

Structured Query Language. A syntax for defining and manipulating data from a relational database. Developed by IBM in the 1970s, it has become an industry standard for query languages in most DBMSs. See also ANSI SQL 89.

table

Information formatted in rows and columns. A set of data elements that has a horizontal dimension (rows) and a vertical dimension (columns) in a DBMS. A table has a specified number of columns but can have any number of rows. See also attribute.

TCP/IP

The Transmission Control Protocol (TCP) is a communication protocol layered above the Internet Protocol (IP). These are low-level communication protocols that allow computers to send and receive data.

topology

1. In geodatabases, relationships between connected features in a geometric network or shared borders between features in a planar topology.
2. In coverages, the spatial relationships between connecting or adjacent features (for example, arcs, nodes, polygons, and points). The topology of an arc includes its from- and to-nodes and its left and right polygons. Topological relationships are built from simple elements into complex elements: points (connected arcs), routes (sets of sections, which are arcs' simplest elements), arcs (sets of connected points), and areas (sets of or portions of arcs). Redundant data (coordinates) is eliminated because an arc array represents a linear feature, part of the boundary of an area feature, or both.

transaction

A logical unit of work conducted in a database environment, comprising one or more SQL statements. Transactions can involve data definition (create an object), data manipulation (update an object), or reading data (select from an object).

UDP (User Datagram Protocol)

UDP is a connectionless datagram transport protocol. Connectionless protocols allow data to be exchanged without setting up a link between processes. Each unit of data, with all the necessary information to route it to the intended destination, is transferred independently of other data packets and can travel over different paths to reach the final destination. Some data packets might be lost in transmission or might arrive out of sequence to other data packets. It is known as a datagram protocol because it is analogous to sending a letter where you don't acknowledge receipt.

URL

Universal Resource Locator—convenient and succinct way to direct people and applications to a file or other electronic resource via a number of different Internet protocols.

version

A version is an alternative representation of the geodatabase that has an owner, a description, and a level of access (private, protected, and public). See also version merging and version reconciliation.

version merging

The process of reconciling two versions of a feature dataset into a common version. If conflicting edits have been made in either of the merged versions, these conflicts are resolved, either automatically or by an interactive process.

version reconciliation

The process of updating a version of a dataset with changes made in another version. Using this technique, a version can remain up-to-date with changes even if it is within a long transaction lasting many months.

wizard

A tool that leads a user step-by-step through an unusually long, difficult, or complex task such as software installation.

Index

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W