ArcGIS® 9

ArcSDE® Administration Command Reference

Table Of Contents

Welcome to the ArcSDE Administration Commands Reference

The administration commands allow the ArcSDE administrator to manage and monitor the use of an ArcSDE service and geodatabase. These commands can only be used with ArcSDE geodatabases licensed under ArcGIS Server Enterprise edition. This reference describes the administration commands in detail, providing both the syntax and example usage. For an explanation of the syntax format, see Command syntax.

To use the commands, you must be able to connect to the geodatabase. See Connections to the geodatabase for information on making a connection using an ArcSDE service or a direct connection.

The available ArcSDE administration commands can be categorized into two functional groups—commands for managing the ArcSDE server or system tables and commands for managing data.

ArcSDE server administration commands

The ArcSDE server administration commands are used to monitor and control certain aspects of the geodatabase configuration, as well as the ArcSDE service.

Most of the server/system table administration commands can only be run by the ArcSDE administrator. Some, such as sdesetup, require database administrator privileges to execute.

The following is a list and brief description of each server administration command. See the individual command topics for more complete descriptions, syntax, and examples.

sdeconfig	Manages your ArcSDE server configuration table (SERVER_CONFIG), which stores parameters and values that define how the ArcSDE server uses memory.
sdedbtune	Manages parameters of the DBTUNE table, which contains parameters and values, grouped by configuration keywords, that specify how data is stored in the database.
sdegcdrules	Manages geocoding rules

sdegdbrepair	Identifies and repairs any inconsistencies between the adds (A) and the deletes (D) tables of a versioned geodatabase
sdelocator	Manages locators
sdelog	Administers log files (used primarily for shared log files)
sdemon	Monitors and manages the ArcSDE service
sdeservice	Manages the ArcSDE service on Windows
sdesetup	Does the initial geodatabase creation within the DBMS, upgrades the geodatabase, and updates your license file

Data management commands

It is recommended that you manage your data using the ArcGIS Desktop software whenever possible. Note that under certain circumstances, using the data management commands to alter or delete your geodatabase datasets can lead to their corruption.

At present, some of the operations performed by the data management commands cannot be performed using the Desktop software. (For example, you must use the sdetable command to create multiversioned views.) For this reason, and for the continued support of those sites that use the commands in existing scripts, the data management commands are provided below. The first set of commands perform at least some operations that currently cannot be done using the Desktop software. The second set of commands are used to accomplish tasks that can be entirely performed using the Desktop software. See the individual command topics for more complete descriptions, syntax, and examples.

sdeexport	Creates an ArcSDE export file

sdeimport	Imports data from an ArcSDE export file
sdegroup	Merges features by combining their geometries into multipart shapes Features are grouped by tiles or by a business table attribute.
sdelayer	Administers feature classes
sderaster	Manages raster layers
sdetable	Administers business tables and their data
sdexinfo	Describes an ArcSDE export file
sdexml	Administers XML columns

cov2sde	Converts ArcInfo coverages to geodatabase feature classes
sde2cov	Converts geodatabase feature classes to ArcInfo coverages
sde2shp	Extracts features from a geodatabase feature class or log file and writes them to a shapefile
sde2tbl	Converts geodatabase tables to INFO or dBASE tables
sdeversion	Manages geodatabase versions
shp2sde	Converts shapefiles to geodatabase feature classes

tbl2sde	Creates a table in the geodatabase, appends data to an existing table in the geodatabase, or replaces records in an existing table in the geodatabase

Command syntax

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

-<letter or word>	Specifies a command option, for example, -o, -a, or interleave Letters are both lowercase and uppercase, so it is important that you input the correct case.
< >	Required argument—replace with appropriate value. 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.

Operations

An operation performs a specific task related to the command and is specified by -o <task>.
For example, some of the sdetable command operations are:

```
   sdetable -o create
   sdetable -o delete
   sdetable -o truncate
   sdetable -o list
   sdetable -o describe
   sdetable -o create_index
   sdetable -o delete_index
```

Each operation has a set of options.

Options

Like operations, options are specified by -<letter>. The -<letter> used for a particular option is standard across all commands, with a few exceptions. For instance, the option to specify a service is always -i, whereas the option -a indicates attribute mode on most commands, but indicates column names on sdetable, sets NoData pixels on sderaster, and lists the attribute data values on sdexinfo. Although a letter is sometimes used for two different types of options, it is never used for two different options in the same command.

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

```
sdelog -o list -u <username> [-i <service>] [-q]
```

The example above has three options. Anything enclosed within brackets [ ] isn't required. The user name option is required, but service [-i] and quiet [-q] are not. 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. For example:

```
$ sdetable -o delete -t test30 -u sde
Enter Database User password:
```

The -p <DB_user_password> is not encrypted. Therefore, leaving this option out of the command and typing the password when prompted allows you to input the password without it appearing on the screen.

Some options have a list of choices such as:

```
shp2sde -o append [-a {none | all | file=<file_name>}]
```

The -a option in this example allows you to choose how to execute the option. These elements are enclosed in curly brackets {}.

A few options have two or more parameters separated by commas. For example, the following option:

```
-l <table,column>
```

specifies a business table and column. Do not add spaces between the parameters.

Special characters

In identifiers

Identifiers containing special characters, such as a user name based on a Windows login in the form DOMAIN\login, need to be delimited with double-quotes. In the Windows login identifier example, the back slash (\) is a special character.

For Windows commands, the backslash (\) is an escape character. Escape characters indicate to the administration command that the character that follows should be passed to the database as a simple character.

The administration commands do not recognize double-quotes; therefore, a command for which you would specify "DOMAIN\login", such as sdelayer or sdetable, will not function correctly unless you place an escape character before each double-quote. In this example, you would place a back slash (\) before each quote mark. This tells the administration command to pass the subsequent character as part of the character string. For example:

```
sdelayer -o describe -O \"WORLD\shannon\" ...
```

In the example above, sdelayer will pass \"WORLD\shannon\" as "WORLD\shannon".

Delimited identifiers can be used with any of the supported DBMSs, but will mostly only be used with ArcSDE for SQL Server.

On UNIX

UNIX users should be careful with special characters such as ?. Depending on the UNIX shell version, you may need to include 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 include \, which tells the shell to use the next character directly rather than as a special character.

Therefore, in a C shell, you would use

```
$ sdetable -\?
```

In a Bourne shell, you can simply use

```
$ sdetable -?
```

Some commands include optional SQL query statements, or WHERE clauses to limit the features retrieved from a table or log file. 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:

```
" state_code = 'CO' "
```

If your DBMS encloses character literals with double quotes, enclose the entire expression with single quotes:

```
' state_code = "CO" '
```

Check your DBMS documentation to determine which is correct for your DBMS.

Geodatabase connections

You can make connections to an ArcSDE geodatabase using either an ArcSDE service or a direct connection. You provide different information with the -i command depending on the type of connection you use.

ArcSDE services

An ArcSDE service can be specified in several ways. By default, the service is named in the SDEHOME/etc/services.sde file. However, you can override the service in the services.sde file in three ways:

- The SDEINSTANCE environment variable can be set to the name of a service.

  ```
  setenv SDEINSTANCE esri_sde3
  ```

- The sdemon command -i <service> option accepts the service name as its argument.
- The sdemon command -H <sde_directory> option specifies the home directory (normally one not specified by SDEHOME) within which the etc/services.sde file contains the service name.

The -i <service> and -H <sde_directory> options are mutually exclusive and cannot be used together. Either of the options overrides the service specified in both the SDEINSTANCE variable and the services.sde file.

Either <service> is specified in the -i option or <service> is listed in the services.sde file located in the home directory specified by the -H <sde_directory> option.

If neither of the above have been specified, the service is specified by the SDEINSTANCE system variable.

If no SDEINSTANCE variable exists, the service is specified in the services.sde file.

The port:<port number> can be substituted for the service name if you want to enter the port number of the service name directly.

```
$ sdelayer -o list -u av -p mo -i port:5151
```

The above set of rules does not apply to the sdemon -o start operation. For this operation, specifying the service on the command line indicates that you want to perform a remote startup. The start operation doesn't use the service specified in the SDEINSTANCE variable.

Administration commands and services

All administration commands, except sdesetup, connect to the service through the giomgr process and are clients of the ArcSDE service.

For these commands, the default service is esri_sde. To operate on a service other than the default, you must set the SDEINSTANCE variable:

```
$ setenv SDEINSTANCE production

$ sdelayer -o list -u arcview -p mapobjects
```

or include the -i <service> option on the administration command line:

```
$ sdelayer -o list -u arcview -p mapobjects -i production
```

The sdesetup command behaves differently because it must be able to directly connect to the DBMS instance.

The sdesetup command also requires you to enter either the database administrator's password or the ArcSDE administrative user password, depending on the DBMS used. Please refer to your ArcSDE Installation Guide for details.

Direct connections to the geodatabase

To specify a direct connection to an ArcSDE geodatabase, you need to provide direct connect information with the -i option. For example, if you are connecting to an Oracle9*i* geodatabase, you would specify -i sde:oracle9i. Other DBMS direct connections are specified as follows:

Oracle 10*g*	-i sde:oracle10g
SQL Server	For a default instance: -i sde:sqlserver:<name or IP Address of the server> For a named instance: -i sde:sqlserver:<machine_name>\<instance_name>
DB2	-i sde:db2
Informix	-i sde:informix:<odbc_dsn> <odbc_dsn> = ODBC data source name

System administration commands

ArcSDE Administration Commands

sdeconfig

The **sdeconfig** command manages the server configuration file and the contents of the SERVER_CONFIG table.

Usage syntax

```
sdeconfig -o alter -v <Property_Name=Value,...> [-i <service>]
[-s <server_name>] [-D <database>]
-u <ArcSDE_admin_user> [-p <ArcSDE_admin_password>] [-N] [-q]


sdeconfig -o import -f <SERVER_Info file> [-i <service>]
[-s <server_name>] [-D <database>]
-u <ArcSDE_admin_user> [-p <ArcSDE_admin_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 -h


sdeconfig -?
```

Operations

Operation	**Description**
import	Imports the entire configuraton file into the SERVER_CONFIG table (SDE_server_config in SQL Server databases) for the specified database, replacing the

	existing contents
export	Exports the entire contents of the SERVER_CONFIG table (SDE_server_config in SQL Server databases) to the specified configuration file
list	Lists the value of a specified server configuration parameter
alter	Alters the value of the specified configuration parameter

Options

Options	**Description**
-D	Database name (not supported on Oracle)
-f	Server configuration file
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-s	ArcSDE server host name (default: localhost)

-N	No verification is performed; the operation begins immediately after being invoked.
-o	Operation
-p	DBMS user password for ArcSDE administrator (for list and export, can be password of a nonadministrative DBMS user.)
-P	The name of a server configuration parameter to be listed
-q	Quiet—all titles and warnings are suppressed.
-u	DBMS user name for ArcSDE administrator (for list and export, can be a nonadministrative DBMS user)
-v	The value to assign to a server configuration parameter formatted as <property name>=<value>

Discussion

The sdeconfig command allows you to edit and view the values of the server configuration parameters that are stored in the SERVER_CONFIG table. The server configuration parameters are read by client applications upon connection to an ArcSDE server. They are not reread until the application connects again.

Use the export operation to write the parameters to a file. The values of the parameters can be edited using an ASCII text editor. Use the import operation to write the parameters back to the SERVER_CONFIG table. The alter operation allows you to change a single server configuration parameter value. The list operation displays the value of a specified parameter.

The following table lists the default values that will be written to the file if you have not altered any of the parameter values.

Parameter name	**Default value**
ALLOWSESSIONLOGFILE	Oracle, DB2, and Informix = FALSE SQL Server = TRUE
ATTRBUFSIZE	50000
AUTOCOMMIT	1000
BLOBMEM	1000000
CONNECTIONS	48
DEFAULTPRECISION	64
DETECT8XNULLSHAPE (Oracle only)	FALSE
DISABLEAUTOREG (not used with SQL Server)	TRUE
DISABLEDC	FALSE
ERRLOGMODE	TIC
HOLDLOGPOOLTABLES	TRUE
INT64TYPES	FALSE
LARGEIDBLOCK	0
LAYERAUTOLOCKING	FALSE
LOGFILEPOOLSIZE	0
MAXARRAYBYTES	550000

MAXARRAYSIZE	100
MAXBLOBSIZE	1000000
MAXBUFSIZE	65536
MAXDISTINCT	512
MAXGRIDSPERFEAT	8000
MAXINITIALFEATS	10000
MAXSTANDALONELOGS	0
MAXSTREAMS	25
MAXTIMEDIFF	-1
MINBUFOBJECTS	512
MINBUFSIZE	16384
PACKETSIZE (SQL Server only)	0
PRECISION10 (Oracle only)	FALSE
PROCSTATS	-1
RASTERBUFSIZE	10485760
READONLY	FALSE
SHAPEPTSBUFSIZE	400000
SMALLIDBLOCK	16
STATEAUTOLOCKING	FALSE
STATECACHING	TRUE
STATUS	1

STREAMPOOLSIZE	6
TCPKEEPALIVE	FALSE
TEMP	\tmp
THRESHOLD (not used with SQL Server)	0
TLMINTERVAL	1
TRIMLOCKINGTYPE (Oracle only)	ONLY DURING COMPRESS

For an explanation of possible settings for configuration parameters, see the ArcGIS Desktop or ArcGIS Server help. Search for the topic "ArcSDE initialization parameters".

Examples

Alter the parameters in the SERVER_CONFIG table

Rather than export and import all the configuration parameters, the alter operation allows you to edit the values of a list of one or more parameters. The -v option requires that the parameter name be followed directly by the equal sign and the value. There must not be any white space surrounding the equal sign; if there is, a syntax error will result.

```
$ sdeconfig -o alter -u bob -p cat -N -q -v procstats=5

Successfully altered SERVER_CONFIG Table.
```

Export contents of SERVER_CONFIG table

The export operation exports the entire contents of the SERVER_CONFIG table to the specified configuration file.

```
$ sdeconfig -o export -u bob -p cat -f config.txt -N -q

Successfully exported to file config.txt on server jolex
```

Note: You don't need administrative privileges to export the SERVER_CONFIG table.

Import the configuration file

The import operation allows you to import the parameters listed in a configuration file to the SERVER_CONFIG table. The contents of the table are completely replaced by the contents of the file. If the file does not contain a parameter, a default value is inserted for the missing parameter. While reading the file, if a parameter is encountered that is not recognized, the parameter is ignored. Therefore, misspelled parameter names will be ignored.

```
$ sdeconfig -o import -u sde -p sde -f config.txt -N -q

Successfully imported SERVER_CONFIG Table.
```

List parameter values

Using the list operation, you either list all the parameter values of the SERVER_CONFIG table or just one. If the parameter cannot be found, you will receive a warning that the parameter does not exist and a list of all the parameters contained within the table. The example below lists the value for parameter PROCSTATS.

```
$ sdeconfig -o list -u bob -p cat -N -q -P procstats

PROCSTATS -1
```

Note: Any DBMS user can run the list operation.

ArcSDE Administration Commands

sdedbtune

The **sdedbtune** command manages parameters of the DBTUNE table (SDE_dbtune in SQL Server databases).

Usage syntax

```
sdedbtune -o alter [-i <service>]-k <keyword_name>
-P <parameter_name> -v <parameter_value> [-s <server_name>]
[-D <database>] -u <ArcSDE_admin_user_name> [-p
<ArcSDE_admin_password] [-N] [-q]


sdedbtune -o delete_data [-i <service>] -k <keyword_name>
-P <parameter_name> [-s <server_name>][-D <database>]
-u <ArcSDE_admin_user_name> [-p <ArcSDE_admin_password>][-N] [-q]


sdedbtune -o export -f <DBTune file>
[-i <service>][-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]


sdedbtune -o import -f <DBTune file>
[-i <service>] [-s <server_name>] [-D <database>]
-u <ArcSDE_admin_user_name> [-p <ArcSDE_admin_password>] [-N] [-q]


sdedbtune -o list [-i <service>] {[-l <keywords|parameters>]|
[-k <keyword_name>]} [-P <parameter_name>] [-s <server_name>]
[-D <database>][-u <DB_user_name>] [-p <DB_user_password>][-q]


sdedbtune -h


sdedbtune -?
```

Operations

Operation	**Description**
alter	Allows you to alter the value of a configuration parameter in the

	DBTUNE table
delete_data	Deletes a configuration keyword or keyword/parameter combination from the DBTUNE table
export	Exports entire DBTUNE table to the specified dbtune file
import	Imports dbtune file into the DBTUNE table for the specified database.
list	Returns all parameters for a specified keyword, a list of all keywords, or a list of allowed parameters

Options

Options	**Description**
-D	Database name (not supported on Oracle)
-f	DBTUNE file name and path to file If no directory is specified with the path name, the file is exported to and imported from whichever directory the command is issued.
-h or -?	Use either of these options to see the usage and options for the command. **Note:** If using a C shell, use -h or "-\?".

-i	ArcSDE service name or direct connect information
-k	Configuration keyword present in DBTUNE table
-l	List options
-N	No verification is performed; the operation begins immediately after being invoked.
-o	Operation
-p	DBMS user password for ArcSDE administrator (for list and export, can be password of a nonadminstrative DBMS user)
-P	DBTUNE configuration parameter name
-q	Quiet—all titles and warnings are suppressed.
-s	ArcSDE server host name (default: localhost)
-u	DBMS user name for ArcSDE administrator (for list and export, can be a nonadministrative DBMS user)
-v	The value to assign to a configuration parameter, formatted as <property name>=<value>

Discussion

The DBTUNE table (SDE_dbtune in SQL Server databases) contains configuration keyword and parameter name/configuration string pairs that control how data is stored in your database. You should edit values in the DBTUNE table either by using the alter or delete_data operations. To add new keywords, you will need to use the export and import operations.

In ArcSDE 9.1 or lower, the only way to edit the contents of the DBTUNE table was to export its contents to a dbtune file in the SDEHOME/etc directory, alter the file with a text editor, and import it back to the table. At release 9.2, the dbtune file doesn't have to be in the SDEHOME/etc directory.

If you want to delete a keyword and its parameters or just specific parameters for a particular keyword, you must use the delete_data operation, not export/import.

```
     sdedbtune –o delete_data -k <keyword_name>  [-P <parameter_name]
```

For more information regarding the DBTUNE configuration keywords and parameters, refer to the topic "The dbtune file and the DBTUNE table" in the ArcGIS Desktop on-line help, which can be accessed from the Knowledge Base tab of the ESRI support site (http://support.esri.com).

Examples

Change a configuration string value

The alter operation can be used by the ArcSDE administrator to change the configuration string value of a given configuration parameter for a specific configuration keyword. Since configuration parameters can be used with multiple configuration keywords, you must specify the keyword and parameter name in the command.

The following example alters the A_STORAGE parameter under the DEFAULTS keyword in the DBTUNE table of an Oracle database.

```
$ sdedbtune -o alter -k DEFAULTS -P A_STORAGE -v "TABLESPACE
GIS_DATA PCTFREE 0 INITRANS 4" -u sde -N

Enter Database User password:

ArcSDE 9.2 for Oracle 10g        Wed Aug 3 16:00:54 2005
Attribute            Administration Utility
----------------------------------------------------

            Successfully altered dbtune data.
```

Delete a configuration keyword and its parameters

To delete an entire configuration keyword and its parameters or a specific parameter, the ArcSDE administrative user would use the delete_data operation. Be sure the keyword is truly no longer needed by other users or applications, such as an ArcIMS Map Service, before you delete it.

In the first example below, the entire keyword is being deleted. In the next example, only the UI_TEXT parameter of the WKB_GEOMETRY keyword is being deleted.

```
$ sdedbtune -o delete_data -k DEFAULTS_ARCHIVE -u sde -p sdepswd

ArcSDE 9.2 for Oracle 10g       Wed Aug 3 16:00:54 2005
Attribute            Administration Utility
----------------------------------------------------
Delete DBTUNE data. Are you sure? (Y/N): y

             Successfully deleted data from dbtune table.
```

```
c:\> sdedbtune -o delete_data -k WKB_GEOMETRY -P UI_TEXT -u sde -p
sdepswd

ArcSDE for SQL Server           Mon Aug  1 16:00:45  2005
Attribute           Administration Utility
----------------------------------------------------
Delete DBTUNE data. Are you sure? (Y/N): y

              Successfully deleted data from dbtune table.
```

Export the contents of the DBTUNE table

The export operation exports the entire DBTUNE table to a file for editing. The file can then be edited with a file-based editor, such as vi or Microsoft's Notepad, and imported back into the DBTUNE table using the import operation. Any database user can export the contents of the DBTUNE table; however, only the ArcSDE administrative user can import the dbtune file.

In this example, the dbtune file dbtune3.sde is exported to the ArcSDE directory located in c:\GISfiles. If no directory had been included with the -f option, the file would have been saved to c:\ because that is the directory from which the command was executed.

```
c:\> sdedbtune -o export -f c:\GISfiles\ArcSDE\dbtune3.sde -N -u sde

Enter Database User password:

ArcSDE 9.2 for SQL Server        Mon Aug  1 16:00:45  2005
Attribute               Administration Utility
---------------------------------------------------------------------
-----------------------

            Successfully exported to file
"c:\GISfiles\ArcSDE\dbtune3.sde"
            on server rocky
```

Note that since the -p <password> option is not specified in the command, the password prompt appears. Also note that in ArcSDE 9.1 and prior releases, the dbtune file had to be exported to SDEHOME\etc.

Import the dbtune file

The import operation imports the contents of the dbtune file into the DBTUNE table for the specified database. Only the values present in the dbtune file will be imported back into the DBTUNE table after the table is initially created. Only the ArcSDE administrator can import the dbtune file.

Note: Do not attempt to import the file while tables or layers are being created or altered.

```
c:\> sdedbtune -o import -f c:\GISfiles\ArcSDE\dbtune3.sde -u sde

Enter Database User password:

ArcSDE 9.2 for SQL Server        Mon Aug  1 16:00:45  2005
```

```
Attribute           Administration Utility
----------------------------------------------------
Import DBTUNE Table. Are you sure? (Y/N): y

            Successfully imported from file
"c:\GISfiles\ArcSDE\dbtune3.sde"
```

Since -N was not specified in the example above, the verification prompt appears and "y" must be typed before the import completes.

List keywords and parameters

The list operation returns either all parameters for a specified keyword, all of the allowed parameters, all of the keywords in the DBTUNE table, or, if no keywords are specified, list returns the entire contents of the DBTUNE table. Any database user can use the list operation. The example below lists the parameters for the WKB_GEOMETRY keyword to ensure that the UI_TEXT parameter was actually deleted when the previous example was executed.

```
c:\> sdedbtune -o list -k WKB_GEOMETRY

Enter Database User password:

ArcSDE for SQL Server Build 721 Mon Aug  1 16:00:45  2005
Attribute           Administration Utility
----------------------------------------------------
 ##WKB_GEOMETRY

     GEOMETRY_STORAGE          "OGCWKB"

END
```

Note that the -u option didn't need to be specified in the command, but a database user's password was still required. For the list operation, the password provided can be that of any valid database user.

When you specify the -l parameters option, a list of the parameters allowed for your DBTUNE table is returned.

```
c:\> sdedbtune -o list -l parameters

Enter Database User password:

ArcSDE for SQL Server         Mon Aug  1 16:00:45  2005
Attribute           Administration Utility
----------------------------------------------------
Allowed parameters for DBTUNE table are:

     F_STORAGE
     F_INDEX_AREA
     F_INDEX_LEN
     F_INDEX_FID
     S_STORAGE
     S_INDEX_SP_FID
     S_INDEX_ALL
     B_STORAGE
```

```
        ...the rest of the parameters...
```

When you specify the -l keywords option, a list of the keywords currently found in the DBTUNE table is returned.

```
c:\> sdedbtune -o list -l keywords

Enter Database User password:

ArcSDE for SQL Server       Mon Aug  1 16:00:45  2005
Attribute           Administration Utility
----------------------------------------------------
Keywords found in DBTUNE table are:

     DATA_DICTIONARY
     DEFAULTS
     LOGFILE_DEFAULTS
     NETWORK_DEFAULTS
     NETWORK_DEFAULTS::DESC
     NETWORK_DEFAULTS::NETWORK
     SURVEY_MULTI_BINARY
     TOPOLOGY_DEFAULTS
     TOPOLOGY_DEFAULTS::DIRTYAREAS
     WKB_GEOMETRY
```

If you specify only the list operation, all keywords and parameters will be returned. This list is usually long, however, and in most cases when using the MS-DOS cmd window, you will not be able to scroll back through the entire list.

```
c:\> sdedbtune -o list
Enter Database User password:
```

sdegcdrules

The **sdegcdrules** command manages ArcSDE geocoding rules. The geocoding rule base is a collection of files that directs the geocoding engine how to standardize address data and match it to the related location in the reference data. Each address style uses a specific set of rule base files. The functions of the rule base are twofold: the first is the address data standardization process; the second is the matching of the address to the reference data.

Usage syntax

```
sdegcdrules -o delete -f <rule_file> [-N] [-i <service>] [-s
<server_name>] [-D <database>]
[-p <ArcSDE_admin_password>] [-q]


sdegcdrules -o export -f <rule_file> [-N] [-d <directory_name>] [-i
<service>]
[-s <server_name>] [-D <database>] [-u <DB_user_name>] [-p
<DB_user_password>] [-q]


sdegcdrules -o import -f <rule_file> [-N] [-i <service>] [-s
<server_name>]
[-D <database>] [-p <ArcSDE_admin_password>] [-q]


sdegcdrules -o list [{-S <style> | {-T <type>}] [-i <service>] [-s
<server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdegcdrules –h


sdegcdrules -?
```

Operations

Operation	Description

delete	Deletes the specified geocoding rule record
export	Exports a geocoding file from the geodatabase
import	Imports a geocoding rule file to the geodatabase
list	Lists the geocoding rules stored in the geodatabase

Top

Options

Options	**Description**
-d	Directory to which the geocoding rule file is exported
-D	Database name (not supported on Oracle)
-f	Name of and path to the geocoding rule file
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information

-N	No verification is performed; the operation begins immediately after being invoked.
-o	Operation
-p	DBMS user password (for import and delete operations, must be the ArcSDE administrative user's DBMS password)
-q	Quiet—all titles and warnings are suppressed.
-s	ArcSDE server host name (default: localhost)
-S	Geocoding rule style name
-T	Geocoding rule type
-u	DBMS user name (for import and delete operations, must be the ArcSDE administrative DBMS user)

Top

Description

The sdegcdrules command manages the set of geocoding rules that are stored in an ArcSDE geodatabase. Locators use a set of geocoding rules that specify to the geocoding engine how to parse, standardize, and match addresses. In a multiuser geodatabase, each geocoding rule file is stored as a record in the GCDRULES table. For more information on geocoding rule file types, refer to the *Geocoding Rule Base Developer Guide*, which you can download from http://support.esri.com.

Examples

Delete a geocoding file

The delete option deletes a geocoding rule record from the GCDRULES table in the geodatabase.

```
$ sdegcdrules –o delete –f us_addr.cls –i sde90_geocoding –D
geocoding
```

Export a geocoding file

The export operation exports a specified geocoding rule file from the geodatabase to a file on disk. The name of the created file is the name of the geocoding rule record stored in the GCDRULES table.

```
$ sdecgdrules –o export –d d:\workspace –f us_addr.cls –i
sde90_geocoding –D geocoding
```

Import a geocoding file

The import operation imports a geocoding rule file stored on disk into the GCDRULES table in the geodatabase.

```
$ sdegcdrules –o import –f d:\us_addr.cls –i sde90_geocoding –D
geocoding
```

List the contents of the GCDRULES table

The list option lists the contents of the GCDRULES table in the multiuser geodatabase. The listing includes the ID, the geocoding rule style, and the geocoding rule type. If you specify a type with the -T option, the list only contains those rules of that type. The –S option can be used to restrict the list to a particular geocoding rule style name. In the first example below, the list is restricted to rules of the type tbl. In the second example, the list is restricted to rule style us_addr.

```
c:\> sdegcdrules –o list -T tbl

ArcSDE 9.2 for SQL Server Build 721 Mon Aug  1 16:00:45  2005
Geocoding Rule Table                 Administration Utility
-----------------------------------------------------
ID           STYLE                        TYPE
-----------------------------------------------------
11           direct                        tbl
12           expand                        tbl
24           prefix                        tbl
25           spedit                        tbl
36           suffix                        tbl



$ sdegcdrules –o list –S us_addr –i sde90_geocoding –D geocoding

ArcSDE 9.2 for Oracle10g Server Build 723 Tues Aug  2 11:00:35  2005
Geocoding Rule Table                 Administration Utility
-----------------------------------------------------
```

```
ID           STYLE                        TYPE
-------------------------------------------------------
41           us_addr                       cls
42           us_addr                       dct
44           us_addr                       pat
45           us_addr                       stn
```

ArcSDE Administration Commands

sdegdbrepair

The **sdegdbrepair** command is used to identify and repair any inconsistencies between the adds (A) and the deletes (D) tables and versioning system tables of a versioned geodatabase.

Usage syntax

```
sdegdbrepair -o diagnose_metadata -d
<{ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX}>
[-H <home_directory>] [-u <ArcSDE_admin_user>] [-p
<ArcSDE_admin_password>]
[-D <database>] [-s <server_name>] [-q]


sdegdbrepair -o diagnose_tables -d
<{ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX}>
[-r versioned table name | file=<table_list>] [-V {ALL | NONE |
<version_name>}]
[-H <home_directory>] [-u <ArcSDE_admin_user>] [-p
<ArcSDE_admin_password>]
[-D <database>] [-s <server_name>] [-q]


sdegdbrepair -o repair_metadata -d
<{ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX}>
[-H <home_directory>] [-u <ArcSDE_admin_user>] [-p
<ArcSDE_admin_password>]
[-D <database>] [-s <server_name>] [-N] [-q]


sdegdbrepair -o repair_tables -d
<{ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX}>
[-r versioned table name | file=<table_list>] [-V {ALL | NONE |
<version_name>}]
[-H <home_directory>] [-u <ArcSDE_admin_user>] [-p
<ArcSDE_admin_password>]
[-D <database>] [-s <server_name>] [-N] [-q]


sdegdbrepair -h


sdegdbrepair -?
```

Operations

Operation	Description
diagnose_metadata	Shows inconsistent rows for the versioning tables
diagnose_tables	Shows the number of unreferenced (orphaned) rows in the delta (A and D) tables
repair_metadata	Repairs inconsistent rows (as possible) for the versioning tables
repair_tables	Removes and repairs orphaned rows in delta (A and D) tables

Options

Options	Description
-d	Type of DBMS; Oracle 9i, Oracle 10g, SQL Server, DB2, or Informix
-D	Database name (not supported on Oracle)
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".

-H	ArcSDE home directory (SDEHOME) This is 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	DBMS user password for ArcSDE administrator
-q	Quiet—All titles and warnings are suppressed.
-r	The name of the versioned table or list of tables to be repaired If providing a list of tables, you must specify a file that contains the list. Each table must be on a separate line in the file.
-s	ArcSDE server host name (default: localhost)
-u	DBMS user name for ArcSDE administrator; must be provided in the command
-V	Option for checking and repairing duplicate rows ALL—All registered versioned tables will be checked for duplicates for all versions. NONE—No check for duplicates Version_name—Specify a version to check and repair duplicates.

Discussion

What the repair_table and repair_metadata operations for sdegdbrepair do

When you use the repair_tables operation, the sdegdbrepair utility scans through all the versioned tables and attempts to repair any inconsistencies discovered between the adds and the deletes tables. This is shown in the Examples section under "Running sdegdbrepair with the repair_tables operation".

If you know a certain versioned table or a specific version is affected, you can run the sdegdbrepair command with the –r option to repair the specific versioned table or use the –V option to repair the specific version. The –V option is to control the level of checking duplicate entries in the versioned tables. The default is to check duplicates for all versions.

Repairing just one table or version will take less time than scanning all the versioned tables for inconsistencies, so if you know which table or version is affected, you can expedite the repair process. For instance, if you compress the geodatabase and it fails with a unique constraint violation, you can check the compress output to determine which version failed. Then you can run the sdegdbrepair command with the –V option to only repair that version.

Examples using the –V and –r options are shown in the sections "Running sdegdbrepair with the repair_tables operation and repairing a selected versioned table" below.

The repair rules for metadata are:

- Versions pointing at a nonexistent state will be made to point to the state of its parent version if possible. If the DEFAULT version is pointing at a nonexistent state, it will be updated to point at the base state (0).

- States that have an invalid parent state are not repaired but a warning is issued.

- Rows containing an extra lineage_id in the STATE_LINEAGES table will be removed.

- Rows containing state ids in the MVTABLES_MODIFIED table that are pointing to nonexistent states will be removed.

Before running sdegdbrepair with the repair operations

Prior to running the repair operations:

1. Make a backup copy of your database.
2. **ORACLE ONLY:** If you are going to run the repair operations against your Oracle instance, you must grant the following additional privileges to the ArcSDE administrator's DBMS user in Oracle:

ALTER ANY INDEX
ALTER ANY TABLE
ANALYZE ANY
CREATE ANY INDEX
CREATE ANY PROCEDURE
CREATE ANY SEQUENCE
CREATE ANY TRIGGER
CREATE ANY VIEW
CREATE SESSION
DROP ANY INDEX
DROP ANY TABLE
DROP ANY VIEW
DROP ANY PROCEDURE
DROP ANY SEQUENCE
EXECUTE ANY PROCEDURE
SELECT ANY SEQUENCE
SELECT ANY TABLE
UNLIMITED TABLESPACE

3. Connect as the ArcSDE administrator. Be sure the privileges listed above have been granted to the ArcSDE administrator DBMS account if you are using ArcSDE for Oracle.
4. If you don't use the -r option to specify tables that you are sure no other user is editing, then be sure there are no users connected to the database when you run the utility.
5. Be sure you are using the correct version of the utility; for example, the 9.1 version of the utility will not run on an ArcSDE 9.0 geodatabase. Similarly, the DBMS specified with the -d option must match the type of DBMS you are using; for example, you wouldn't specify -d ORACLE10G if you were running the command on a SQL Server database.

Note: ORACLE ONLY: After you finish running the sdegdbrepair utility with the repair operations, you may revoke the following privileges from the ArcSDE administrator DBMS user account:

ALTER ANY INDEX
ALTER ANY TABLE
ANALYZE ANY
CREATE ANY INDEX
CREATE ANY TRIGGER
CREATE ANY VIEW
DROP ANY INDEX
DROP ANY TABLE
DROP ANY VIEW
DROP ANY PROCEDURE
DROP ANY SEQUENCE
EXECUTE ANY PROCEDURE
SELECT ANY SEQUENCE

Top

Examples

Determining if there are potential errors in your database

Run the sdegdbrepair command with the diagnose_tables and diagnose_metadata operations to determine if your database contains orphaned or duplicate rows in the delta tables or versioned business table. If the list operation returns results similar to the examples below, you need to run the repair option to repair the inconsistencies. Results are returned on screen but are also written to the sde_repair.log located in the SDEHOME\etc directory. To discover if your database contains inconsistencies, first, run –o diagnose_tables.

Orphaned rows

In this example, the diagnose_tables operation is being run against a local Oracle database. It discovers orphaned records in the Adds table for the feature class TOPO_AREA.

```
sdegdbrepair -o diagnose_tables -d ORACLE10G
Enter DBA Password:

ESRI ArcSDE Server Repair Utility Wed Sep 06 08:51:17 2006
----------------------------------------------------------
1 unreferenced states found in GDB.A42 (GDB.TOPO_AREA)
Diagnose Tables: 220 multiversioned tables examined,
                 1 multiversioned tables had orphaned or missing
rows.
```

The output will report the cumulative total number of versioned tables affected. This indicates that you have inconsistencies and should run the repair utility.

Next, execute sdegdbrepair with the diagnose_metadata operation.

```
sdegdbrepair -o diagnose_metadata -d ORACLE10G
Enter DBA Password:

ESRI ArcSDE Server Repair Utility Wed Sep 06 08:56:36 2006
----------------------------------------------------------
All versions reference valid states.

All states have valid parent ids.

Lineage entry for lineage 6162 has a state id of 6292, which is
invalid.

Table GDB.TOPO_AREA was reported modified in non-existing state 6292
```

Duplicate entries in deletes tables

The diagnose_tables operation can also detect if there are duplicate entries in the deletes tables. In the following example, duplicate entries are found in the deletes table in a local Oracle database.

```
sdegdbrepair -o diagnose_tables -d ORACLE9I
Enter DBA password:

ESRI ArcSDE Server Repair Utility Thu Feb 15 10:49:55 2007
----------------------------------------------------------
1 redundant deletes entries found in MAP.D40 (MAP.CITIES)
```

```
Diagnose Tables: 110 multiversioned tables examined,
         1 multiversioned tables had orphaned, duplicate, missing,
or redundant rows.
```

Entries missing from the deletes tables

Another inconsistency the diagnose_tables operation will detect involves duplicates in the adds table that don't have corresponding rows in the deletes table. In the example that follows, duplicate rows are found in the adds table that are missing rows in the deletes tables for two feature classes in a remote SQL Server database.

```
sdegdbrepair -o diagnose_tables -d SQLSERVER -D porcupine -s monstro
Enter DBA password:

ESRI ArcSDE Server Repair Utility Tue Jan 23 17:27:44 2007
----------------------------------------------------------
1 duplicate rows found in PORCUPINE.POTENTIAL_SITES
1 duplicate rows found in PORCUPINE.STREETS
Diagnose Tables: 66 multiversioned tables examined
         2 multiversioned tables had orphaned, duplicate, or missing
rows
```

No inconsistencies

If no inconsistencies are detected when you run the diagnose_tables operation, as shown below, no further action is required. In the example below, the diagnose_tables operation is being run against a DB2 database.

```
sdegdbrepair -o diagnose_tables -d DB2 -s rocco
Enter DBA Password:

ESRI ArcSDE Server Repair Utility Wed Sep 06 16:28:48 2006
----------------------------------------------------------
Diagnose Tables: 180 multiversioned tables examined,
     0 multiversioned tables had orphaned, duplicate, missing, or
redundant rows.
```

Running sdegdbrepair with the repair_tables operation

To repair the inconsistencies reported by the diagnose operations, run the sdegdbrepair command with the repair_tables/metadata operations. The output for the repair is put in the sde_repair.log located in the etc directory of SDEHOME.

The following is an example of running the sdegdbrepair –o repair_tables command:

```
sdegdbrepair -o repair_tables -d ORACLE10G -p sde42

ESRI ArcSDE Server Repair Utility Wed Sep 06 11:14:35 2006
----------------------------------------------------------
Repair Instance Delta Tables, Are you sure? (Y/N): y
Repair operation completed without error.

The output in the SDEHOME/etc/sde_repair.log
```

```
[Wed Sep 06 11:14:38 2006] Mvdata Clean: Successfully cleaned
GDB.TOPO_AREA
[Wed Sep 06 11:14:38 2006] Clean: 1 adds rows removed.
[Wed Sep 06 11:14:38 2006] Clean: 1 delete rows removed, 0 delete
rows recreated.
[Wed Sep 06 11:14:38 2006] Mvdata Clean: Nothing to fix for
GDG.CARTO_TEXT
[Wed Sep 06 11:14:38 2006] Mvdata Clean: Nothing to fix for
GDB.TOPO_LINE
```

You can run the command with the diagnose_table operation again to make sure all inconsistencies have been repaired, as shown in the following example:

```
sdegdbrepair -o diagnose_tables –d ORACLE10G -p sde42

ESRI ArcSDE Server Repair Utility Wed Sep 06 11:30:45 2006
----------------------------------------------------------
Diagnose Tables: 220 multiversioned tables examined,
                 0 multiversioned tables had orphaned or missing
rows.
```

Note: If you run the repair_tables operation without specifying the -r option and any of the tables are locked, the repair will fail with a message similar to the one shown below:

```
sdegdbrepair -o repair_tables -d ORACLE10G -H
C:\ArcSDE\ArcSDE\ora10gexe
Enter DBA password:

ESRI ArcSDE Server Repair Utility Mon Feb 05 18:56:03 2007
----------------------------------------------------------
Repair Instance Delta Tables, Are you sure? (Y/N): y
Error: Lock request conflicts with an established lock (-49).
Error: Unable to complete repairing orphaned rows.
```

Similarly, if you run the repair_tables operation and do specify the -r option and the specified table (or tables) are locked, the repair will also fail. If, however, you run the repair_tables operation and specify the -r option and none of the tables specified are locked, the repair will complete successfully. See the section "Running sdegdbrepair with the repair_table operation and repairing a selected versioned table" below.

Running sdegdbrepair with the repair_metadata operation

The following is an example of running the sdegdbrepair –o repair_metadata command against a database named develop on a named SQL Server instance:

```
sdegdbrepair -o repair_metadata -d SQLSERVER -D develop -s
santo\fortuna
Enter DBA password:
```

```
ESRI ArcSDE Server Repair Utility Wed Sep 06 11:22:57 2006
----------------------------------------------------------
Repair Instance Versioning Metadata, Are you sure? (Y/N): y
Repair operation completed without error.
```

The output in the SDEHOME/etc/sde_repair.log would be as follows:

```
[Wed Sep 06 11:22:59 2006] All versions already reference valid
states.
[Wed Sep 06 11:22:59 2006] All states have valid parent ids.
[Wed Sep 06 11:22:59 2006] 1 lineage entries with invalid state ids
removed.
[Wed Sep 06 11:22:59 2006] 1 mvtables modified rows with invalid
state ids removed.
```

Running sdegdbrepair with the repair_table operation and repairing a selected versioned table

The following is an example of running the sdegdbrepair –o repair_table command to repair a single table.

First, execute the sdegdbrepair command with the diagnose_table operation to determine if you need to repair your database. In this example, the versioned table name is tom.fittings.

```
sdegdbrepair -o diagnose_tables -d INFORMIX -r tom.fittings -p sde

ESRI ArcSDE Server Repair Utility Wed Sep 06 11:52:14 2006
----------------------------------------------------------
1 unreferenced states found in tom.fittings
Diagnose Tables: 28 multiversioned tables examined,
                 1 multiversioned tables had orphaned or missing
rows.
```

Next, execute the command using the repair_tables operation and –r option, specifying the name of a versioned table you want to repair.

```
sdegdbrepair -o repair_tables -d INFORMIX -r tom.fittings -p sde

ESRI ArcSDE Server Repair Utility Wed Sep 06 15:42:07 2006
----------------------------------------------------------
Repair Instance Delta Tables, Are you sure? (Y/N): y
Repair operation completed without error.

[Wed Sep 06 15:42:09 2006] Mvdata Clean: 1 duplicates found in
tom.fittings at state 6318, lineage name 6162.
[Wed Sep 06 15:42:09 2006] Mvdata Clean: Successfully cleaned
tom.fittings
[Wed Sep 06 15:42:09 2006] Clean: 1 adds rows removed.
[Wed Sep 06 15:42:09 2006] Clean: 2 delete rows removed, 1 delete
rows recreated.
```

To make sure the repair worked and there are no other inconsistencies in the database, execute the command with the diagnose_tables operation again:

```
sdegdbrepair -o diagnose_tables -d INFORMIX -p sde

ESRI ArcSDE Server Repair Utility Wed Sep 06 15:17:12 2006
----------------------------------------------------------
Diagnose Tables: 28 multiversioned tables examined,
                 0 multiversioned tables had orphaned or missing
rows.
```

You can also specify a list of tables to repair using the -r option by indicating a file that contains a list of the tables to repair. In the following example, the file tbls_list.txt contains a list of tables to be repaired. Each table in the file is on a separate line, as shown below.

```
map.cities
map.parcels
map.hydrants
map.LAKES
map.powerlines
map.DISPATCH
```

To repair these tables, specify the file with the -r option.

```
sdegdbrepair -o repair_tables -d INFORMIX -r file=tbls_list.txt -p
sde
```

Running sdegdbrepair with the repair_tables operation and repairing duplicates for a selected version

The following is an example of detecting and repairing duplicates for a specific version. The duplicates would typically be detected with the trim operation and compress or save edits.

```
sdeversion -o compress -u sde -p sde

ArcSDE 9.2 for Oracle10g Build 1069 Tue Sep 5 16:01:14 2006
Version Administration Utility
------------------------------------------------------------
Compress state tree: Are you sure? (Y/N): y
Error: Underlying DBMS error (-51).
Error: Unable to compress state tree.
ORA-00001: unique constraint (GDB.A40_PK) violated
```

Use the –V option to specify a particular version. In the example below, the name of the selected version is GDB.V1.

```
sdegdbrepair -o repair_tables -d ORACLE10G -r GDB.TOPO_AREA -V
GDB.V1 -p sde

ESRI ArcSDE Server Repair Utility Wed Sep 06 15:16:27 2006
----------------------------------------------------------
Repair Instance Delta Tables, Are you sure? (Y/N): y
Repair operation completed without error.
```

The output in the SDEHOME/etc/sde_repair.log is shown below.

```
[Wed Sep 06 15:16:29 2006] Mvdata Clean: 1 duplicates found in
GDB.TOPO_AREA at state 6315, lineage name 6162.
[Wed Sep 06 15:16:29 2006] Mvdata Clean: Successfully cleaned
GDB.TOPO_AREA
[Wed Sep 06 15:16:29 2006] Clean: 0 delete rows removed, 1 delete
rows recreated.
```

ArcSDE Administration Commands

sdelocator

The **sdelocator** command manages ArcSDE locators.

Usage syntax

```
sdelocator -o create -n <locator_name> -f <locator_properties_file>
-T <{t|v}> [-S <locator_description>] [-i <service>] [-s
<server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>] [-q]


sdelocator -o delete {-n <locator_name> | -I <locator_id>} [-N] [-i
<service>]
[-s <server_name>] [-D <database>] -u <DB_user_name> [-p
<DB_user_password>] [-q]


sdelocator -o describe [-n <locator_name> | <-I locator_id>] [-i
<service>]
[-s <server_name>] [-D <database>] -u <DB_user_name> [-p
<DB_user_password>] [-q]


sdelocator -o list [-T <{t|v|d}>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>] [-q]


sdelocator -o refresh {-n <locator_name> | -I <locator_id>} [-N] [-i
<service>]
[ -s <server_name>] [-D <database>]-u <DB_user_name> [-p
<DB_user_password>] [-q]


sdelocator -h


sdelocator -?
```

Operations

Operation	**Description**
create	Creates a locator from the

	property file
delete	Removes a locator's entries from the LOCATORS (SDE_locators in SQL Server) and METADATA (SDE_metadata in SQL Server) tables
describe	Describes the locator properties
list	Provides a list of the locators in the database
refresh	Rebuilds the geocoding indexes for a locator

Options

Options	**Description**
-D	Database name (not supported on Oracle)
-f	Name of and path to the locator properties file
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information

-l	Locator ID
-n	Locator name
-N	No verification is performed; the operation begins immediately after being invoked.
-o	Operation
-p	DBMS user password
-q	Quiet—all titles and warnings are suppressed.
-s	ArcSDE server host name (default: localhost)
-S	Locator description string
-T	Locator type (t: template, d: system defined, v: validated)
-u	DBMS user name

Discussion

The sdelocator command manages the set of locators and locator styles stored in a multiuser geodatabase. For information on the format and contents of locator and locator style files, refer to *Geocoding in ArcGIS* in the ArcGIS user documentation set.

Note: Locators can be created, modified, and rebuilt using the ArcGIS Desktop software, which is the recommended method for performing these tasks.

Examples

Create a locator

The create operation reads the locator's properties from a property file and, if the locator does not already exist in the LOCATORS table (SDE_locators in SQL Server databases), adds an entry. The locator's properties are stored in the METADATA table (SDE_metadata in SQL Server). The -T option specifies which type of locator is being created. Typically, you will only use the create operation to load locator styles into the ArcSDE database (using the –T t option) and create new locators based on this style using ArcCatalog.

```
$ sdelocator -o create -n "Custom Locator Style" -f d:\custom.lot -T
t -S “Custom locator style” -i sde90_geocoding –D geocoding -u sde -
p sde
```

Delete a locator

The delete operation removes a locator's entries from the LOCATORS and METADATA tables.

```
$ sdelocator -o delete -n "Custom Locator Style" -i sde90_geocoding
–D geocoding -u sde -p sde
```

Describe a locator's properties

The describe operation describes the locator properties. If no locator is specified, all locators are described.

```
$ sdelocator -o describe -n "US Streets" -i sde90_geocoding –D
geocoding -u sde -p sde
```

List the locators stored in the database

The list operation provides a list of the locators in the database. The listing includes the locator ID, type, name, owner, category, and description. The example below shows a partial list.

```
c:\> sdelocator -o list -i sde90_geocoding –D geocoding -u sde -p
sde

ArcSDE 9.2 for SQL Server Build 721 Mon Aug 1 16:00:45 2005
------------------------------------------------------------------
-----
ID T  Name                     Owner    Category    Description
------------------------------------------------------------------
-----
1  t  Single Field             SDE      Address     Single Field
2  t  StreetMap USA            SDE      Address     StreetMap USA
(EDGE)
3  t  US Alphanumeric Ranges   SDE      Address     US Alphanumeric
Ranges
```

```
...
24 t  World Cities with Country SDE     Address     World Cities
with Country
25 t  ZIP+4                    SDE      Address     Zip+4
```

The list operation can be qualified with the locator type option -T to include only those locators of a certain type. There are three main types of locators stored in a multiuser geodatabase:

- Locator styles (specified using the -T t option) are used as templates for creating new locators.
- Locators that are created based on a locator style are designated by the -T v option.
- Other locators, called attached locators (specified using the -T d option), are copies of a locator, including runtime options, used to geocode a feature class, and are used when the feature class is rematched.

```
c:\> sdelocator -o list -T v -i sde90_geocoding –D geocoding -u sde
-p sde

ArcSDE 9.0 for SQL Server Build 1798 Fri Nov 21 11:31:31 PST 2003

Locator Administration Utility
-------------------------------------------------------
ID T Name Owner Category Description
--------------------------------------------------------------------
----------------
36 v Redlands_Streets SDE Address US Streets
```

Rebuild a locator's geocoding indexes

The refresh operation rebuilds the geocoding indexes for a locator. The geocoding indexes for a locator should be rebuilt after the reference data for the locator has been edited.

```
$ sdelocator –o refresh –I 29 –i sde90_geocoding –D geocoding –u sde
-p sde
```

sdelog

The **sdelog** command administers log files. This command is primarily useful if you are utilizing shared log files.

Usage syntax

```
sdelog -o alter -L <logfile> -k <PERSISTENT,TEMPORARY> [-O <Owner>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]


sdelog -o clean [-O <Owner>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
[-N] [-A] [-q]


sdelog -o delete -L <logfile> [-O <Owner>] [-i <service>]
[-s <server_name>] [-D <database>] -u <DB_user_name>
[-p <DB_user_password>] [-N] [-q]


sdelog -o display -L <logfile> [-O <Owner>] [-i <service>]
[-s <server_name>] [-D <database>] -u <DB_user_name>
[-p <DB_user_password>] [-q]


sdelog -o grant -U <user> [-O <Owner>] [-i <service>] [-s
<server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
[-q]


sdelog -o list [{-O <Owner> | -l <table,column> | -T <table>}]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdelog -o revoke -U <user> [-O <Owner>] [-i <service>] [-s
<server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
[-q]


sdelog -o truncate -L <logfile> [-O <Owner>] [-i <service>]
[-s <server_name>] [-D <database>] -u <DB_user_name>
[-p <DB_user_password>] [-N] [-q]
```

```
sdelog -h

sdelog -?
```

Operations

Operation	Description
alter	Modifies log file's persistence
clean	Deletes all temporary log files
delete	Deletes a single log file
display	Prints the contents of a log file
grant	Grants access to the owner's log file for a user
list	Lists a user's log files
revoke	Revokes access from the owner's log files for a user
truncate	Deletes the contents of a log file

Options

Options	Description

-A	Deletes all log files: temporary, persistent, and pre-8.0 format
-D	Database name (not supported on Oracle)
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-k	Log file persistence keyword, either TEMPORARY or PERSISTENT (Doesn't affect session or stand-alone log files.)
-l	Layer table and spatial column name
-L	Log file name (not containing user)
-N	No verification is performed; operation begins immediately after being invoked.
-o	Operation
-O	Owner of the log file
-p	DBMS user password

-q	Quiet—all titles and warnings are suppressed.
-s	ArcSDE server host name (default: localhost)
-T	Table name
-u	DBMS user name
-U	The user to whom log file access will be granted or revoked or whose log files will be cleaned

Discussion

ArcSDE shared log files are stored in the DBMS tables SDE_LOGFILES and SDE_LOGFILE_DATA. Prior to ArcSDE 8, they were stored on disk as files. Some of the usage syntax still reflects that old style of use. During the migration to ArcSDE 8, the files on disk became DBMS table records. A log file is represented as a single row in the SDE_LOGFILES table. The records of the log file are stored in the SDE_LOGFILE_DATA table and are referenced by the foreign key LOGFILE_ID, which is also the primary key of the SDE_LOGFILES table.

Beginning with ArcSDE 9.0, two additional types of log files were introduced—session and stand-alone log files. This command (sdelog) is primarily used to administer shared log files.

Examples

Alter how long a log file persists

The alter operation modifies a log file's persistence from either temporary to persistent or vice versa. This example shows log file log0630, which is owned by user bernard, being changed from temporary to persistent.

```
c:\> sdelog -o alter -L log0630 -k PERSISTENT -O bernard -u av
```

Clean up temporary log files

Both the delete and clean operations of sdelog will remove log files. The clean operation, by default, removes a user's temporary log files. Essentially, all rows are truncated from the SDE_LOGFILES and SDE_LOGFILE_DATA tables for the user specified by the -U flag.

```
$ sdelog -o clean -u slojoe
```

Specify the -N option if you don't want to be prompted for verification.

```
$ sdelog -o clean -u slojoe -N
```

Specify the -A option to delete all of a user's logs: temporary, persistent, and the style preceding SDE 3.x.

```
$ sdelog -o clean -u slojoe -N -A
```

Delete a log file

The delete operation of the sdelog administration command removes a single log file. You can enter the -N option to suppress the verification prompt. In this case, the single record in the SDE_LOGFILES table containing the LOGFILE_NAME column value of logBCAa06599 is deleted. From the SDE_LOGFILE_DATA table, all records with a LOGFILE_ID foreign key that matches the SDE_LOGFILES table's primary key LOGFILE_ID value are deleted as well.

```
$ sdelog -o delete -L logBCAa06599 -u av -N
```

Print the contents of a log file

The log file utility display operation provides a list of shape IDs contained in the log file. When inputting the log file name, do not include the user name at the beginning. In this case, the log file name including the user name would have been RJP.XX00000131_0.

```
c:\> sdelog -o display -L XX00000131_0 -u RJP

Logfile               Administration Utility
-------------------------------------------------------------
2
9
12
19
95
97
182
514
620
961
1026
1111
2002
```

Grant access to another user

The grant operation allows another user to have access to a specific user's log file. The log file's owner must grant this access. Here, rocketjay is being given access to cc's log files.

```
c:\> sdelog -o grant -U rocketjay -O cc -u cc
```

List log files

You can list the log tables with the log file utility list operation.

```
$ sdelog -o list -u RJP
Enter Database User password:

Logfile          Administration Utility
-----------------------------------------------

-------------------------------------------------------------
Logfile Name  :  RJP.XX00000131_0
Logfile Id    :  1
Logfile Type  :  TABLE
Table Id      :  41
Persistence   :  TEMPORARY
-------------------------------------------------------------
```

Revoke access to a log file

The revoke operation removes access from the owner's log files for a specified user.

```
c:\> sdelog -o revoke -U doc -O cc -u av
```

Truncate

The truncate operation deletes the contents of a log file but does not delete the log file itself. Here, log file log_perm4's contents are being deleted.

```
c:\> sdelog -o truncate -L log_perm4 -u sde
```

ArcSDE Administration Commands

sdemon

The **sdemon** command is used to monitor and manage the ArcSDE service.

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: Execution privileges are granted exclusively to the ArcSDE administrator for the start operation. 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 {[-i <service>] [-s <server_name>] | [-H
<sde_directory>]}
[-p <ArcSDE_admin_password>]


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


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


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


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


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

```
sdemon -h
```

```
sdemon -?
```

Operations

Operation	Description
info	Displays information about users, configuration, statistics, locks, or environment variables Does not require the ArcSDE administrator password to execute
kill	Kills all connections or a specified connection to the service
pause	Disallows further client connection requests to be processed (does not disallow direct connect clients)
resume	Allows client connection requests to be processed again; client tasks are allowed to connect to ArcSDE servers through this ArcSDE service.
shutdown	Shuts down the ArcSDE service immediately if no server tasks are running
start	Starts the ArcSDE service if it's not running **Note:** Only the ArcSDE administrator can use this operation.
status	Reports the service status

Options

Options	**Description**
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-H	ArcSDE home directory (SDEHOME)
-i	ArcSDE service name
-l	Inquires about configuration, locks, statistics, users, environment variables, or user geodatabases stored in Oracle 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 user's connections to ArcSDE and associated process identifiers vars: Displays ArcSDE service environment variables instance: Name of the user-owned geodatabase to be queried (Oracle only; used only with the info operation)
-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 server host name (default: localhost)
-t	Kills server tasks: *all*: Forcefully removes all server tasks *pid*: Removes the task identified by the process identifier

Discussion

The sdemon command allows you to manage and monitor ArcSDE services. Use it to start, stop, and view the current properties of an ArcSDE service.

The start, stop, pause, and resume operations normally manage ArcSDE services installed on UNIX systems. ArcSDE services installed on Microsoft Windows systems are usually managed via the Services window. However, the sdemon command may be used from a Microsoft command prompt but only if the ArcSDE service has been created with the sdeservice command.

The ArcSDE service may be managed remotely by including the server -s and service -i options on the command line. UNIX operating system ArcSDE services can be remotely managed by an sdemon operation executed on a Windows operating system. However, Windows operating system ArcSDE services can only be remotely managed by an sdemon operation executed from another Windows operating system. To remotely manage an ArcSDE service installed on a UNIX system, additional steps are required. Refer to the topics Starting an ArcSDE service and Stopping an ArcSDE service in the ArcGIS Desktop or ArcGIS Server Help for a detailed discussion.

Note that you only need to specify the -H option (SDEHOME) if the variable is not set or you have multiple services and, therefore, multiple SDEHOME directories.

For a detailed discussion of the management of an ArcSDE service, refer to the Connecting to an ArcSDE geodatabase book in the ArcGIS Desktop or ArcGIS Server Help.

Examples

Obtain information about an ArcSDE service

To display information about users, configuration, statistics, locks, or environment variables for a running ArcSDE service, use the info operation. This example is requesting information about the ArcSDE service's configuration (only partial results are shown):

```
$ sdemon -o info -I config

ArcSDE I/O Manager Configuration Parameters at
Fri May 13 11:52:43 2005
————————————————————————————————————-
ArcSDE Version          9.1
ArcSDE Server Build     for DB2
Underlying DBMS         DB2
Max. Server Connections 50
Root Path               D:\arcsde
Temp Path               C:\Temp
...
```

In this example, information is requested for the SDE and user-schema geodatabases on an Oracle database:

```
$ sdemon -o info -I instances

ArcSDE Instance sdeora10g's instances on verde at Thu Feb 24
08:23:14 2005
------------------------------------------------------------
Instance  Type          Created                        Id
-------- ----- -------------------------------- ------------
SDE      MASTER   Thu Oct 28 15:30:45 2004        0
FIELD    PROJECT  Thu Oct 28 16:38:46 2004        1
```

The following example shows the users who are connected to the ArcSDE geodatabase.

```
c:\> sdemon -o info -I users

ArcSDE Instance esri_sde Registered Server Tasks on datenbank at Wed
Aug 30 10:29:00 2006
--------------------------------------------------------------------
----
S-ID   User       Host:OS                    Started
----- ---------- ------------------------- ------------------------
----
546   RJP        hundehutte:Win32           Wed Aug 30 09:32:56
2006
547   DBO        halter:Win32               Wed Aug 30 10:11:05
2006
```

Note: The users parameter for the -I option will return connections made through both the service and a direct connection as long as the ArcSDE service is running. If the service is stopped, no information on any connections will be returned.

Remove connections to a service

The kill operation disconnects all connections or a specified connection to the ArcSDE service. Kill should generally only be used if a user process has shut down incorrectly or hung. Here, the connection for process ID 102 is being killed.

```
$ sdemon -o kill -t 102 -p pswd
```

Pause an ArcSDE service

Using the pause operation prevents any additional client connection requests to be processed. No client tasks are allowed to connect to ArcSDE servers through this ArcSDE service until the service is resumed. (Direct connections can still be made.)

```
c:\> sdemon –o pause

Please enter ArcSDE DBA password:

ArcSDE instance esri_sde on Rabbit is Paused, no further connections
will be allowed.
```

Resume a paused service

To allow client connection requests to be processed again, use the resume operation. Client tasks are allowed to connect to ArcSDE servers through this ArcSDE service once this command is executed.

```
c:\> sdemon –o resume

Please enter ArcSDE DBA password:

ArcSDE instance esri_sde on Rabbit is Resuming, new connections will
now be allowed.
```

Shut down an ArcSDE service

Use the shutdown operation to immediately stop an ArcSDE service if no server tasks are running. If server tasks are running, you will be prompted to remove the running tasks before the shutdown takes place. If you use the -N option when shutting down, all server tasks stop and the system shuts down immediately.

```
c:\> sdemon -o shutdown -N

Please enter ArcSDE dba password:

ArcSDE instance esri_sde on Rabbit is shutdown!
```

Start an ArcSDE service

The ArcSDE administrator can use the start operation to start an ArcSDE service that isn't running. The administrator can start the service locally (execute the command on the machine on which the service will be running) or remotely. If starting a service locally on a Windows machine, you will usually do this through the Services menu. If starting a Windows service remotely, it must be performed from another Windows machine that can reach the host computer through the network, and the command must include the host computer and service name. For example:

```
c:\> sdemon -o start -s chile -i dev -p secret
```

Note: The service name (in this case, dev)—not the port number—must be used. Using the port number will result in an error.

You can start a service from the command line on a local Windows server. If you only have one ArcSDE service, you don't need to specify the service name. If you don't specify the administrator password, you will be prompted for it.

```
c:\> sdemon –o start

Please enter ArcSDE DBA password:

ArcSDE instance esri_sde started Thu Aug 04 09:27:30 2005
```

To remotely start a service on a UNIX server, you can work from either a Windows or another UNIX machine. Before you can do this, however, there are some configurations you must perform.

1. Create the $SDEHOME/etc/dbtune.sde file with the database connection and the library path to the ArcSDE and DBMS dynamic libraries.

   For example

   ```
   set ORACLE_HOME=/bluebeard/oracle
   set ORACLE_SID=ora
   set
   LD_LIBRARY_PATH=/usr/lib:/bluebeard/oracle/lib:/bluebeard/oraexe/
   sdeexe91/lib
   ```

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.

   ```
   esri_sde  5151/tcp
   esri_sde  5151/udp
   ```

3. 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>
   # SDE remote start-up entries.
   ```

4. 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.

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

   $ kill -HUP 112
   ```

Once these configuration steps have been completed, you can remotely start the service using the start operation.

```
$ sdemon -o start -s bluebeard -i esri_sde -p my_password
```

This example shows starting a local service on a UNIX server:

```
$ sdemon -o start -s orr -i sde -p crab_apples
```

Find out the status of an ArcSDE service

The status operation reports the service status. The example below is reporting on the status of the esri_sde instance. It shows the service is running (not stopped or paused) and currently has two connections.

```
$ sdemon –o status

ArcSDE Instance esri_sde Status on Rabbit at Thu Aug 04 09:23:35
2005
--------------------------------------------------------------------
-------------------
Server Connection Mode:                  Accepting Connections
Active Server Processes:                 2
```

sdeservice (Windows only)

The **sdeservice** command manages the ArcSDE service on Windows operating systems.

Usage syntax

```
sdeservice -o create -p <ArcSDE_admin_password> [-n]
[-H <sde_directory>] [-d <ORACLE,SID | SQLSERVER,SQLSERVERINSTANCE |
DB2,DB2INSTANCE | INFORMIX>] [-i <service>]
[-u <service_user>] [-P <service_user_password>]


sdeservice -o delete
[-d <ORACLE,SID | SQLSERVER,SQLSERVERINSTANCE |
DB2,DB2INSTANCE | INFORMIX>]
[-i <service>] [-N]


sdeservice -o register -r <registry_keyword> -v <value>
-p <ArcSDE_admin_password> [-i <service>]
[-d <ORACLE,SID | SQLSERVER,SQLSERVERINSTANCE |
DB2,DB2INSTANCE | INFORMIX>]


sdeservice -o unregister -r <registry_keyword>
-p <ArcSDE_admin_password> [-i <service>]
[-d <ORACLE10G,SID | ORACLE9I,SID | ORACLE8I,SID |
SQLSERVER,SQLSERVERINSTANCE | DB2,DB2INSTANCE | INFORMIX>]


sdeservice -o modify -r <registry_keyword>
-p <ArcSDE_admin_password> -v <new_value> [-i <service>]
[-d <ORACLE,SID | SQLSERVER,SQLSERVERINSTANCE |
DB2,DB2INSTANCE | INFORMIX>]


sdeservice -o list [-i <service>]


sdeservice -h
```

Operations

Operation	Description
create	Creates a service
delete	Deletes a service
register	Adds a registry keyword to the Windows registry
unregister	Removes a registry keyword from the Windows registry
modify	Modifies an existing registry keyword
list	Displays service information for all or a specified service

Options

Options	Description
-d	Used to identify to which RDBMS the service connects and service dependency A service dependency identifies a service that should start before ArcSDE. DB2INSTANCE, SQLSERVERINSTANCE, or SID are optional and are used to identify an instance of a database, either remote or local. SID (default: ORCL). DB2INSTANCE (default: DB2-0). SQLSERVERINSTANCE SQL Server named instance or data

	source (default: MSSQLServer).
-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; required if the service is not called esri_sde
-n	Exclude database service from list of services that are ArcSDE dependent.
-N	No verification is performed; the operation begins immediately after being invoked
-o	Operation
-p	DBMS password for ArcSDE administrator ArcSDE for Coverages password syntax: password; layer_admin_password.
-P	ArcSDE service user password (Windows service log on account password)
-r	Registers/Unregisters/Modifies the following Windows registry keywords: • SDEHOME—ArcSDE software location • SDE_DBA_PASSWORD—

	ArcSDE administrative user's password • SDE_DBA_USER—ArcSDE administrative user name • ADMIN_DATABASE (DB2/INFORMIX/SQLSERVER only)—Database with ArcSDE system tables • NLS_LANG (ORACLE only)—ArcSDE server code page • NET_SERVICE_NAME (ORACLE only)—Oracle TWO_TASK or LOCAL environment variable
-u	ArcSDE Windows service account user* 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 in to the LAMBERT domain and your user name is joe, enter LAMBERT\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. Services are created with either this tool or with the Post-Installer. Creating a service will add several registry entries, which can include SDE_DBA_PASSWORD, SDEHOME, and ADMIN_DATABASE. The delete operation will delete the ArcSDE service and all related registry entries.

Use the modify option to modify existing registry entries such as SDEHOME or SDE_DBA_PASSWORD. Use the register option to register new entries not provided by default, such as ADMIN_DATABASE. The unregister option removes the entry.

Examples

Create a new ArcSDE service

Use the create operation to make a new ArcSDE service on a machine with a Windows operating system. In all create examples, you must manually edit %windir%\system32\drivers\etc\services and %SDEHOME%\etc\services.sde to add the service name and port number.

The following example will create a new ArcSDE service for Microsoft SQL Server:

```
c:\> sdeservice -o create -d SQLSERVER -p spatial.data -i esri_sde -
H z:\arcgis\arcsde\sqlexe -s
```

This example creates a new ArcSDE service for Informix:

```
c:\> sdeservice -o create -d INFORMIX -p sde.space -i esri_inf -u
informix -P sde.inf
```

This example creates a new SQL Server ArcSDE service pointing to a named instance called Oceans\GIS1:

```
c:\> sdeservice -o create -d SQLSERVER,Oceans\GIS1 -p spatial.data -
i esri_sde01 -H z:\arcgis\arcsde\sqlexe
```

Delete an ArcSDE service

The delete operation deletes an existing ArcSDE service on a machine with a Windows operating system.

This example deletes an ArcSDE service on Informix:

```
c:\> sdeservice -o delete -i esri_inf -d INFORMIX
```

Add a registry keyword to the Windows registry

This example registers an ADMIN_DATABASE (database with system tables) named Seamounts.

```
c:\> sdeservice -o register -d SQLSERVER -r ADMIN_DATABASE -v
Seamounts -i esri_sde -p spatial.data
```

Remove a registry keyword from the Windows registry

The unregister operation removes an existing keyword from the Windows registry. Here, the registry keyword added in the last example is unregistered.

```
c:\> sdeservice -o unregister -r ADMIN_DATABASE -p spatial.data
```

Modify a registry keyword

Use the modify operation to modify an existing Windows registry keyword.

```
c:\> sdeservice -o modify -r SDEHOME -p spatial.data -v
c:\ArcSDE2\ora10gexe -d ORACLE10G,ORCL
```

Display service information

The list operation displays service information for all ArcSDE services or an ArcSDE service specified in the command. Below, information will be listed for the service miss.

Note: If the service name is esri_sde, you do not need to specify the -i option; you could just type sdeservice -o list.

```
c:\> sdeservice –o list –i miss

SDE service Information
--------------------------------------------------------------------
---------------------
RDBMS: SQLServer
Name: ArcSde Services(miss)
SDEHOME: C:\ArcGIS\ArcSDE\sqlexe
Datasource: MOOSE
Admin_database: sde
Version: 9.2.0
Status: SERVICE_RUNNING
```

ArcSDE Administration Commands

sdesetup

The **sdesetup** command creates or upgrades the geodatabase repository in the DBMS, and authorizes the software so the service can start and/or direct connections will work.

For ArcGIS Server 9.2 and onward, use the -d option to specify the database in which the geodatabase will be created (or already resides, in the case of an upgrade): either DB2, INFORMIX, ORACLE9I, ORACLE10G, or SQLSERVER.

For ArcSDE releases prior to 9.2, see the discussion section.

Usage syntax

```
sdesetup -o install -d <ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX>
[-H <sde_directory>]
[-u <ArcSDE_admin_user>] [-p <ArcSDE_admin_password>] [-D
<database>] [-s <datasource>]
[-i <service>] [-l <key>] [-N] [-q]
```

```
sdesetup -o list -d <ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX> [-H
<sde_directory>]
[-u <ArcSDE_admin_user>] [-p <ArcSDE_admin_password>] [-D
<database>] [-s <datasource>]
[-i <service>] [-q]
```

```
sdesetup -o update_key -d
<ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX> -l <key>
[-u <ArcSDE_admin_user>] [-p <ArcSDE_admin_password>] [-H
<sde_directory>]
[-D <database>] [-s datasource] [-i <service>] [-N] [-q]
```

```
sdesetup -o upgrade -d <ORACLE9I|ORACLE10G|SQLSERVER|DB2|INFORMIX>
[-H <sde_directory>]
[-u <ArcSDE_admin_user>][-p <ArcSDE_admin_password>][-D <database>]
[-s <datasource>]
[-i <service>] [-l <key>] [-N] [-q]
```

```
sdesetup -h
```

```
sdesetup -?
```

Operations

Operation	Description
install	Creates or updates the geodatabase system tables and stored procedures
list	Lists the installed ArcSDE version
update_key	Updates the license for ArcGIS Server

upgrade	Updates the geodatabase system tables and stored procedures
	
	You must stop the giomgr process before running an upgrade.
	
	To upgrade, the ArcSDE administrator must have write permission to either SDEHOME/geocode or to a directory to which the TEMP environment variable is set, because backups of the locator files are written to the geocode directory or, failing that, to the TEMP directory during an upgrade. If the ArcSDE administrator does not have write permission to either of these directories, a warning message will be issued similar to the following:
	
	`Upgrading geocoding rules.....`
	`[Mon Mar 06 18:14:09 2006] Cannot create backup directory e\geocode\backup.1141697649 for geocoding rules.`
	`[Mon Mar 06 18:14:09 2006] Installing`

	`geocoding rules..` The upgrade will load new locator files, but no backup files will be created of existing locators.

Options

Options	**Description**
-d	The type of DBMS you are using to store the geodatabase (ORACLE9I, ORACLE10G, SQLSERVER, DB2, or INFORMIX) **Note:** There are differences between the setup for Oracle9*i* and Oracle10*g*; be sure to specify the correct one for your site.
-D	Database name (not supported on Oracle)
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-H	ArcSDE home directory (SDEHOME) This is only needed if the SDEHOME variable isn't set or multiple services are in use.
-i	**Only used for user-schema geodatabases in Oracle** Specify the ArcSDE service name or port number information for the master SDE geodatabase followed by the name of the owner of the user-schema geodatabase. For example:

	`-i 5151:bpanju` **Note:** When creating or upgrading a user-schema geodatabase using a direct connection, use the same syntax as shown in the preceding example. Even though there is no service, use 5151. You will also need to add a 5151 service in the Windows services file of the machine from which you are issuing the sdesetup command.
-l	ArcSDE authorization key or location and name of the authorization file
-N	No verification is performed; the operation begins immediately after being invoked.
-o	Operation
-p	ArcSDE administrator's DBMS password The list operation can be run by any DBMS user, therefore, for the list operation, this could be the password of any DBMS user.
-q	Quiet—all titles and warnings are suppressed.
-s	Data source name
-u	ArcSDE administrator DBMS user name **For install and upgrade, the ArcSDE administrator must have database administrator privileges.** After installation or upgrade, the ArcSDE administrator user's privileges can be returned to their run-time state. This is not true, however, for installations on SQL Server. If you run sdesetup -o install as a user with database administrator

	privileges, you will automatically create a DBO-schema geodatabase. If you want to create an SDE-schema geodatabase, do not grant sysadmin privileges to the SDE user. Any DBMS user can run the list operation.

Discussion

The sdesetup administration utility manages the creation and maintenance of the geodatabase system tables, indexes, stored procedures, and triggers. These database objects are owned by the ArcSDE administrator. For Oracle, DB2, and Informix databases, if you do not specify the user name in the command, it is assumed that the ArcSDE administrator (usually sde) is the user. You must supply the ArcSDE administrator password to create these tables. If you do not supply it in the command with the -p option, you will be prompted for it.

For example, you could issue commands with both the administrator user name and password:

```
sdesetup -o install -d INFORMIX -u sde -p password
```

OR with neither the administrator user name nor password, in which case you will be prompted to supply the ArcSDE administrator password

```
sdesetup -o upgrade -d ORACLE10G

Please enter ArcSDE DBA password:
```

OR specify only the administrator password and—for DB2, Oracle, and Informix databases—the ArcSDE administrator user name is assumed.

```
sdesetup -o install -d DB2 -p password
```

For SQL Server databases, if you do not specify the -u option, the user is assumed to be the Windows login of the user currently logged into the computer. If that user does not have administrator privileges, sdesetup will fail.

```
sdesetup -o upgrade -d SQLSERVER -p mine

SDE release upgrade not complete (-51)
```

The geodatabase system tables and stored procedures can be created or updated using the install operation. For the creation of new geodatabases, the install operation will create the tables and stored procedures for the first time. For subsequent executions of the install operation on an existing geodatabase, the system tables are checked to ensure they are current. If not, the install operation will bring them up to date.

Upgrading

Before you upgrade by issuing the upgrade operation (or the install operation), stop the giomgr process. Note that you can run an upgrade even if you have existing versioned data in the database.

To upgrade the system tables on a remote server, you can include the service name with the password in the sdesetup -o upgrade command (-p <password>@<service_name>).

You need to upgrade user-schema geodatabases in Oracle separately. In this case, the owner of the geodatabase (the schema owner) must be given the permissions necessary to run an upgrade and should execute the sdesetup -o upgrade operation on that geodatabase. Consult the install guide for ArcSDE for Oracle to see what permissions are needed to run an upgrade. Be sure to specify the schema name in which the geodatabase resides with the instance (-i) option, as shown below.

```
c:/> sdesetup -o upgrade -d Oracle10G -i 5151:immanuel -u immanuel -
p reason
```

For more information on using geodatabases created in user schemas with your ArcSDE geodatabase in Oracle, consult the topic 'Using multiple geodatabases within a DBMS' in the ArcGIS Desktop or ArcGIS Server Help.

Client applications are not upward compatible with new versions of a multiuser geodatabase. Therefore, before you upgrade your geodatabase, be sure all of your client programs have been updated to a compatible release. Check the install guide of your client application or the ESRI support site for compatibility information. If you are using a custom built application that was created with the ArcSDE C-API or the ArcSDE JAVA API, you must rebuild the application using the new ArcSDE client API libraries before it can connect to an upgraded geodatabase.

Pre-9.2

Prior to ArcGIS Server 9.2, the sdesetup command was different for each type of DBMS. The different commands are listed below:

sdesetupinfx—Used for an Informix geodatabase
sdesetupdb2—Used for a DB2 geodatabase
sdesetupmssql—Used for a SQL Server geodatabase
sdesetupora8i—Used for an Oracle8*i* geodatabase
sdesetupora9i—Used for an Oracle9*i* geodatabase
sdesetupora10g—Used for an Oracle 10*g* geodatabase

Oracle8*i*, Oracle9*i*, and Oracle10*g* setup programs have significant differences. It is important to run the sdesetup command built for the version of Oracle that you are using if you are running a version of ArcSDE 9.1 or older.

Examples

Creating the system tables and stored procedures

The install operation is used to create the geodatabase system tables and stored procedures. As mentioned in the discussion, if the system tables already exist, the install operation will upgrade them.

```
c:\> sdesetup -o install -d SQLSERVER -p fredericton
```

To create a geodatabase in a user's schema in Oracle, run sdesetup -o install and specify the user's name (which is also the name of the schema where the geodatabase will be stored) for the user and use the -i option to specify the connection information to the master SDE geodatabase followed by the schema name. In the following example, a geodatabase is being created in the gwillagers schema and the ArcSDE service port number for the master geodatabase is 6200. There is no need to specify the -l option because the user-schema geodatabase utilizes the license of the master geodatabase.

```
sdesetup -o install -d ORACLE10G -i 6200:gwillagers -u gwillagers -p
not4u
```

Note: To run this command, gwillagers needs to be granted the same privileges the SDE user requires to create the master geodatabase. Consult the ArcSDE for Oracle installation guide for a list of these privileges.

Finding out what version of the ArcSDE component is installed

The list operation will return the version number of the ArcSDE component that is installed on your machine.

```
$ sdesetup -o list -d INFORMIX -p charlottetown
```

Updating your ArcGIS Server license file

The update_key operation copies the supplied license key into the server configuration table, SERVER_CONFIG (SDE_server_config in SQL Server databases). The license key allows you to use the ArcGIS Server software. The license string can also be included with both the install and upgrade operations. The -l argument will accept either a file containing the license key or the license key itself. In the first example, the license key is entered directly after the -l option. (**Note:** You will need to enclose it in quotes.) In the second example, the license key is contained within a text file, and the file is supplied as an argument to the -l option.

```
$ sdesetup -o update_key -d ORACLE10G -p quebec -l
'arcsdeserver,91,0002a56451f8,31-mar-2005,TRA4CAZYXMM00TEJH192'

$ sdesetup -o update_key -d ORACLE9I -p quebec -l
/crow/sde/license.dat
```

Upgrading your installation

Use the upgrade operation to bring an existing geodatabase up to date with the latest additions or changes to a new ArcSDE component installation.

```
c:\> sdesetup -o upgrade -d DB2

Please enter ArcSDE DBA password:
```

Geodatabases in user schemas in Oracle must be upgraded separately from the master SDE geodatabase. In the following example, the geodatabase in the neri schema is being upgraded. Note that the user name specified is also neri. That is because for user-schema geodatabases, the schema owner is the ArcSDE administrative user, not the sde user.

```
sdesetup -o upgrade -d ORACLE10G -u neri -p sword.fish -i 5151:neri
```

Note: To run this command, neri needs to be granted the same privileges the sde user requires to upgrade the master geodatabase. Consult the ArcSDE for Oracle installation guide or the online help topic User permissions for a list of these privileges.

Data management commands

cov2sde

The **cov2sde** converts ArcInfo coverages to ArcSDE feature classes.

Usage syntax

```
cov2sde -o append -l <table,column> [-V <version_name>]
-f <cover,feature_cls> [-r <anno_relate>] [-d <dissolve_item>]
[-a {none | all | file=<file_name>}] [-c <commit_interval>]
[-i <service>] [-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>] [-v]


cov2sde -o create -l <table,column> -f <cover,feature_cls>
[Spatial_Index] [{-R <SRID> | [Spatial_Ref_Opts]}]
[-r <anno_relate>] [-d <dissolve_item>] [-M <minimum_ID>]
[-S <layer_description_str>] [-e <entity_mask>] [-v]
[-a {none | all | file=<file_name>}] [-k <config_keyword>]
[-V <version_name>] [-C <row_id_column>[,{SDE|USER},<min_ID>]]
[-c <commit_interval>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]


cov2sde -o init -l <table,column> -f <cover,feature_cls>
[-r <anno_relate>] [-d <dissolve_item>]
[-a {none | all | file=<file_name>}]
[-c <commit_interval>]
[-i <service>] [-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>] [-v]
```

Where

```
[Spatial_Ref_Opts] := [-x <xoffset,yoffset,xyscale>] [-z
<zoffset,zscale>] [-m <moffset,mscale>] [-G {<projection_ID> |
file=<proj_file_name>}]

[Spatial_Index] := [-g [Grid_Options] | GRID,[Grid_Options] |
AUTOMATIC | NONE | RTREE }]

[Grid_Options] : = [<grid_sz0>[,<grid_sz1>[,<grid_sz2>]] [,][{FULL |
SPARSE}]
```

```
cov2sde -h

cov2sde -?
```

Operations

Operation	Description
append	Adds features from the coverage to an existing feature class in the geodatabase (default)
create	Creates a new feature class in the geodatabase and imports features into the new feature class from the coverage feature class An error is returned if the feature class already exists in the geodatabase.
init	Deletes all the features of an existing feature class in the geodatabase before importing new features from the coverage feature class

Top

Options

Options	Description
-a	**Attribute mode** *none*: Do not load any attributes (default). *all*: Load all attribute columns. If no attribute table exists for the feature class, one will be created. If one exists, incoming schema must be union compatible with the existing table if the APPEND option is used. *file=<file_name>*: File containing lines in the form <INFO_item>

	[SDE_column] The INFO_item selects the column to be output; the SDE_column (optional) is the new column name to be specified. Use this option to select only those INFO columns that you want to create in the feature class in the geodatabase. The columns will be ordered in the feature class as they appear in the list.
-c	Commit rate (default: The value of the AUTOCOMMIT variable in the SERVER_CONFIG table)
-C	RowID's column_name, column_type, and minimum_ID
-d	Dissolve item for Map Librarian layer only
-D	Database name (not supported on Oracle)
-e	Entity types allowed (npsla3+MA) n Nil p Point features s Line (spaghetti) features l Simple line (line string) features a Area features A Annotation c CAD data 3 Three-dimensional features, which can be added to the entity type mask only with the -o add operation + Multipart features M Measures on coordinates The -m option is required to store measure values on each

	coordinate. Measures can be added to the entity type mask only with the -o add operation.
-f	Input feature class name from: Coverage: -f <cover,feature_cls> Librarian: -f <lib_name.layer_name,feature_cls> ArcStorm: -f <db_name.lib_name.layer_name,feature_cls> For annotation, regions, and routes, the feature class subclass is specified as: <feature_cls.subclass_name>.
-g	Spatial index type and parameters GRID,<grid_sz1>[,<grid_sz2>[,<grid_sz3>]],[FULL|SPARSE] Creates multilevel grid index (Grid2 and grid3 are optional.) If you don't enter grid values, it is the same as specifying an AUTOMATIC spatial index type. FULL grids create a spatial index grid on the entire feature envelope. This is the default option if GRID is specified and is the type of grid always created in ArcSDE 9.1 and lower. SPARSE grids create spatial index grids

	only where grids actually include parts of the feature. See the Discussion section for more details.
AUTOMATIC	For Oracle Spatial and Informix, this is RTREE. For all other storage types, if the layer is in normal IO mode and there are 0 features, the grid type is NONE. If one or more features is present and the layer is in normal IO mode, grid sizes will be calculated and set based on the calculations below. If the layer is in load_only IO mode, grid sizes will be calculated and set based on the calculations below when the layer is returned to normal IO mode. If there are no

	valid envelopes detected, grid sizes are calculated as:
	
	grid_size1 = 1000000.0 / layerGrid->xyUnits
	grid_size2 = 1000000000.0 / layerGrid->xyUnits
	
	If it is a point layer, grid sizes are calculated as:
	
	grid_size1 = 1000.0 / layerGrid->xyUnits

NONE	No spatial index is created.
RTREE	Creates Rtree index

-G	Projection specifier (default: coverage's projection, if exists) <projection_ID> Projection ID file=<proj_file_name> File containing projection description string
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-k	Configuration keyword present in DBTUNE table (default: DEFAULTS)
-l	Feature class table and spatial column name
-m	Measure offset, measure scale
-M	Layer's minimum ID (default: 1)

-o	Operation
-p	DBMS user password
-r	Annotation relate name to load anno and feature at the same time; relate will be loaded from the INFO table <cover.REL>.
-R	Spatial reference ID (SRID)
-s	ArcSDE server host name (default: localhost)
-S	Map layer description string
-u	DBMS user name
-v	Verbose option; reports records committed at the commit interval
-V	Version name; if specified, uses only the data that belongs to the version, if not specified, uses the default version (default: sde.DEFAULT) Version names are case sensitive; for example, SDE.DEFAULT is a different version from SDE.default.
-z	z-offset, z-scale (default: 0.0, 1.0)

Discussion

Cov2sde won't load feature-associated annotation from Map Librarian layers. To load feature-associated annotation from Map Librarian, extract the data to a coverage then import it from the coverage.

If loading annotations with no subclass using the -o create operation, you must use the -a none option. You can also load annotations with no subclass into an existing feature class with the -o append or -o init operations.

Column names that contain a pound sign (#) or a dash (-) will have different names in the new ArcSDE feature class, because these characters are converted to underscores. Although <cover#> and <cover-id> will be different, cov2sde will automatically match these columns. If multiple coverages with the same items are imported, create only one feature class, then load all subsequent coverages with the -o append operation; the <cover#> and <cover-id> items will be matched automatically.

Feature class to ArcSDE entity (feature) type mapping

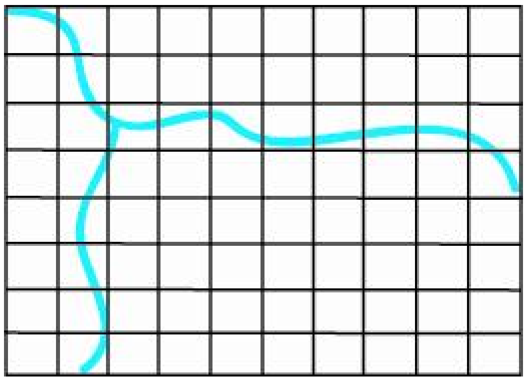
Coverage feature class	**ArcSDE entity (feature) type**
point, node	point, node
line	simple line, line (sl)
polygon	area (a)
region.<subclass>	multipart area (a+)
route.<subclass>	multipart lines with measures (slM+)
anno.<subclass>	point, line, simple line, anno (pslA)

It may be necessary to widen the output width of any item using the INFO ALTER command. If the INFO LIST command is displaying asterisks (******) for any of the items of the coverage feature attribute table, increase the items output width. The cov2sde command uses the INFO item definition to derive the ArcSDE column definition. If an item value doesn't display, it will not load into the DBMS table. This problem will manifest itself as a DBMS error stating that the value is larger than the specified column precision allows.

The create operation requires you to provide a spatial reference. You can use an existing spatial reference system by entering the SRID number. Valid SRID numbers may be obtained by querying the sde.spatial_ref_sys table.

You can also enter the spatial reference by supplying the information directly. Spatial reference information includes the x- and y-coordinate offsets, x- and y-coordinate system units, z-coordinate offset, z-coordinate system units, measures offset, measure system units, and the coordinate reference system. If no false x,y and scale values are provided by the -x option, the extent of the coverage, ArcStorm library, or map library is used to calculate these values automatically. Note that the resulting feature class will have the highest attainable scale factor for that extent. If features that fall outside the original coverage's extent are appended to the feature class with subsequent loading operations, they will be rejected.

Beginning with ArcGIS Server 9.2, you have the option to create sparse spatial index grids. Sparse grids populate the spatial index with information for only those grids that actually contain a portion of a feature. For example, if you created a spatial index on a river feature class, only those grids that the river crossed would be indexed for the river.



This could be more efficient for spatial queries, because grids that do not qualify as crossing the feature (in this case, rivers) will be eliminated from the query results during the primary filter process ArcSDE performs when executing spatial queries. Be aware that if you use third-party applications to perform envelope-on-envelope spatial queries, using a sparse index grid will affect query results.

If instead you used a full grid for the rivers, the nonqualifying grids would still be eliminated, but they wouldn't be eliminated until the secondary filter process.

Also beginning with 9.2, data can be stored in the database in UNICODE format. There is a parameter (UNICODE_STRING) under the DEFAULTS keyword in the DBTUNE table that controls whether or not string data is stored in UNICODE format. By default, this parameter is set to TRUE, meaning character strings will be stored in UNICODE (SE_NSTRING) format.

When you append or use the init operation to import coverage data into an existing feature class, the data is imported using the same configuration keyword with which the feature class was originally made. You need to determine what that configuration keyword is and then add or alter the UNICODE_STRING parameter so it is set to FALSE under that configuration keyword. Since a feature class created prior to 9.2 would have been created without UNICODE storage, adding data to the feature class and trying to store string data **with** UNICODE storage will fail.

To find out with which configuration keyword the feature class was created, use the sdetable command with the describe_reg operation. In the following example, the properties of the feature class wstreets are returned:

```
sdetable -o describe_reg -t big.wstreets -i sde2

ArcSDE 9.2 for Oracle Mon Feb 26 13:52:09 2007
Attribute                   Administration Utility
---------------------------------------------------

---------------------------------------------------
Table Database              : GCDS
Table Owner                 : BIG
Table Name                  : WSTREETS
Registration Id             : 39
Row ID Column               : OBJECTID
Row Id Column Type          : SDE Maintained
Row ID Allocation           : Many
Row Lock                    : Not Enable
Minimum Row ID              : 1
```

```
Dependent Objects          : Layer
Dependent Object Names     : F1, S1, I39
Registration Date          : 02/20/06 11:48:51
Config. Keyword            : HIGH_EDITS
User Privileges            : SELECT, UPDATE, INSERT, DELETE
Visibility                 : Visible
```

The feature class was created using a configuration keyword named HIGH_EDITS. Next, check to see what parameters HIGH_EDITS contains by executing sdedbtune -o list.

```
sdedbtune -o list -k HIGH_EDITS

Enter Database User password:

ArcSDE 9.2 for Oracle Mon Feb 26 13:52:09 2007
Attribute                  Administration Utility
---------------------------------------------------
##HIGH_EDITS

GEOMETRY_STORAGE "SDEBINARY"

ATTRIBUTE_BINARY "BLOB"

A_INDEX_RASTER "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS NOLOGGING"

A_INDEX_ROWID "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS NOLOGGING"

A_INDEX_SHAPE "PCTFREE 0 INITRANS 4
  TABLESPACE VEDITS NOLOGGING"

A_INDEX_STATEID "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS NOLOGGING"

A_INDEX_USER "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS NOLOGGING"

A_INDEX_XML "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS NOLOGGING"

A_STORAGE "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS"

D_INDEX_DELETED_AT "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS NOLOGGING"

D_INDEX_STATE_ROWID "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS NOLOGGING"

D_STORAGE "PCTFREE 0 INITRANS 4
 TABLESPACE VEDITS"

UI_TEXT ""
```

```
END
```

As you can see, there is no UNICODE_STRING parameter in the HIGH_EDITS configuration keyword. When a parameter is missing from a keyword, ArcSDE looks to the DEFAULTS keyword for those missing parameters. If the UNICODE_STRING parameter under DEFAULTS is set to TRUE, adding coverage data to the wstreets feature class will fail because wstreets was not created with UNICODE character storage (strings were stored as type SE_STRING) but you are trying to add new data and storing those characters as UNICODE (SE_NSTRING). To avoid this, you can add the UNICODE_STRING parameter to the HIGH_EDITS configuration keyword and set it to FALSE. Use the sdedbtune export operation to export the contents of the DBTUNE table to a file, edit the file to add UNICODE_STRING "FALSE" under the HIGH_EDITS keyword, then import the edited file back to the DBTUNE table using the sdedbtune import operation. See the sdedbtune command topic in this help for details.

If, instead, the wstreets feature class had been created using the DEFAULTS keyword, you would need to change the UNICODE_STRING parameter under DEFAULTS to "FALSE". Use the sdedbtune alter operation to do this. For example:

```
sdedbtune -o alter -k DEFAULTS -P UNICODE_STRING -v FALSE
```

Remember to change the UNICODE_STRING parameter value back to TRUE after you have finished adding coverage data to the existing feature class; if you do not, all new feature classes created with the DEFAULTS or HIGH_EDITS keywords will be created without UNICODE storage.

Examples

Append data to an existing feature class

The append operation is used to add features from a coverage to an existing feature class in the geodatabase.

```
$ cov2sde -o append -l wells,feature -V myversion -f wells,point
```

Create a feature class from a coverage

Use the create operation to create a new feature class in the geodatabase by loading data from a coverage, Map Librarian, or ArcStorm layer.

1. Loading the regions from a coverage

   ```
   $ cov2sde -o create -l country,shape -f country,region.cntry -
   a all -s tiffany -i esri_av -u andy -p passwrd -g
   GRID,3000,SPARSE
   ```

2. Loading polygons from an ArcStorm layer

   ```
   $ cov2sde -o create -l counties,feature -f
   worldas.worldlib.country,polygon -a all -s tiffany
   -i esri_av -u andy -p password -g GRID,FULL
   ```

3. Loading lines from a coverage with feature-associated annotation

   First, establish a relate between the feature attribute table (in this case, AAT) and the text attribute table (TAT). Save the relate to an INFO file named <cover.rel>:

   ```
   Arc: relate add
   Relation Name: cityrel
   Table Identifier: cities.tatname
   Database Name: info
   INFO Item: cities-id
   Relate Column: cities-id
   Relate Type: linear
   Relate Access: ro
   Relation Name:

   Arc: relate save cities.rel
   1 Relates saved to file cities.rel

   Arc: q
   ```

   Second, specify the name of the relate with -r <relate>. Cov2sde will load the relate from the INFO file called <cover.rel>:

   ```
   $ cov2sde -o create -l cities,feature -f cities,point -r
   cityrel -g AUTOMATIC -x -180,-90,10000 -e pA -k WORLD -a all -
   s stout -u world -p world
   ```

4. Loading points from a Map Librarian layer

   ```
   $ cov2sde -o create -l cities,feature -f maplib.cities,point -
   g SPARSE -x -180,90, 10000 -e slA -k WORLD -a all -s stout -u
   world -p world
   ```

5. Loading stand-alone annotation from an ArcStorm layer

   ```
   $ cov2sde -o create -l rivernames,feature -f
   worldas.worldlib.rivers, anno.riv -g AUTOMATIC -x -180,-
   90,10000 -e slA -k WORLD -a all -s stout -u world -p world
   ```

6. Loading feature-associated annotation from an ArcStorm layer

   First, establish a relate between the feature attribute table (in this case PAT) and the text attribute table (TAT). Save the relate to an INFO file named <database>.<library>.<layer.rel>:

   ```
   Arc: relate add
   Relation Name: cityrel
   Table Identifier: !cities.tatname
   Database Name: info
   INFO Item: cities-id
   Relate Column: cities-id
   Relate Type: linear
   Relate Access: ro
   Relation Name:
   ```

```
Arc: relate save worldas.worldlib.cities.rel
1 Relates saved to file worldas.worldlib.cities.rel

Arc: q
```

Second, specify the name of the relate with -r <relate>. Cov2sde will load the relate from an INFO file called <database>.<library>.<layer.rel>:

```
$ cov2sde -o create -l cityname,shape -f
worldas.worldlib.cities,point -r cityrel -e pA -g NONE -a all
-u world -p world
```

7. Creating a multiversioned feature class from a polygon coverage

```
$ sdeversion -o create -V version1 -S 0 -A public -d Base
version -u andy -p andy -s tiffany -i esri_av

$ cov2sde -o create -l country,shape -f country,polygon -a all
-s tiffany -i esri_av -u andy -p passwrd -g RTREE -V version1
```

8. Setting the row ID column and its initial value

The row ID column of a feature class's business table must be a unique integer column. The row ID column is always required if the feature class is stored in a DB2 or Informix DBMS or if the feature class contains an Oracle Spatial SDO_GEOMETRY column. Feature classes stored in the SQL Server DBMS and the Oracle DBMS (except those that have an SDO_GEOMETRY column) are not required to have a row ID defined unless they are registered with the geodatabase or they are to be registered as multiversioned.

For those feature classes required to have a row ID defined, cov2sde creates a unique integer column called objectid. The -C option allows you to control the name, type, and initial value of the row ID column. If not specified, the default objectid column is created.

If the -C option is specified, the column name can be any name that follows the DBMS naming convention. The default name is objectid. The type can either be maintained by ArcSDE (ArcSDE controls the sequence of values entered into the column, which is required by ArcGIS for any table registered with the geodatabase), or USER maintained, with which your application controls the sequence of values entered into the row ID. The default type is ArcSDE maintained. The initial value of the row ID can be a positive integer. The initial value defaults to 1.

The -C option is useful if the coverage attribute table contains a unique integer column that can serve as the row ID of the feature class to which you intend to convert the coverage. Keep in mind that if the feature class is registered with the geodatabase, ArcSDE maintains further entry of values into the row ID column.

The following is an example of a row ID column being created with the -C option. The row ID column is called cid, it is maintained by ArcSDE, and its initial value is set to 1. If the row ID column is required and the -C option is not specified, the row ID column is created with the name objectid, and it is maintained by ArcSDE with an initial value set to 1.

```
$ cov2sde -o create -l country,shape -f country,polygon -a all
-C cid,sde,1 -s tiffany -i esri_av -u andy -p passwrd -g
AUTOMATIC
```

Delete existing features and import new ones

Use the init operation to delete the existing features in a feature class in the geodatabase before importing new features from a coverage.

```
$ cov2sde -o init -l parcel,shape -f parcel,polygon
```

ArcSDE Administration Commands

sde2cov

The **sde2cov** command converts ArcSDE feature classes to ArcInfo coverages.

Usage syntax

```
sde2cov -o create -l <table,column> [-V <version_name>]
-f <coverage,feature_class> [-P {double | single}]
[-a {none | all | file=<file_name>}] [-w <"where_clause">]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>]

sde2cov -o create -L <log_file> [-V <version_name>]
-f <coverage,feature_class> [-P {double | single}]
[-a {none | all | file=<file_name>}] [-w <"where_clause">]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>]

sde2cov -h

sde2cov -?
```

Operations

Operation	**Description**
create	Creates a new feature class in a coverage An error is returned if the feature class exists.

Options

Options	**Description**

-a	**Attribute mode** *none*: Do not load any attributes. *all*: Load all attribute columns (default). *file=<file_name>*: File containing lines in the form <SDE_column> [INFO_item] The first part selects the SDE column; the second part (optional) allows a new item name to be specified.
-D	Database name (not supported on Oracle)
-f	Output feature class name of coverage
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-l	Input feature class table and spatial column name These must exist, and the executing user must have read access. If you are not the owner of the table, you must qualify the table name as owner.table.
-L	The name of the log file to extract from
-o	Operation
-p	DBMS user password
-P	ArcInfo coverage's precision (default: double)

-s	ArcSDE server host name (default: localhost)
-u	DBMS user name
-V	Version name; if specified, uses only the data that belongs to the version, if not specified, uses the default version (default: sde.DEFAULT) Version names are case sensitive; for example, SDE.DEFAULT is a different version from SDE.default.

Discussion

You must update the topology of any coverage produced by sde2cov. Run CLEAN on polygon and region coverages and BUILD or CLEAN on other feature classes. You must then run CREATELABELS on the output coverage to assign label points to each polygon. You can export annotation to new subclasses of existing coverages. You can't export annotation into an existing subclass of an existing coverage. All other feature types must be exported to new coverages.

Since the ArcSDE multipart area feature class equates to an ArcInfo coverage region feature, you must preserve the regions (and associated attribute data) in the resulting output coverage. Further reduction to composite polygon features could result in data loss and corruption.

Multipart lines with measures exported to route subclasses will produce coverages with an arc and a section at each vertex. You can reduce the number of arcs if the coverage is cleaned for lines using the CLEAN command. All section tables will remain after the CLEAN operation.

Note that annotation created with the geodatabase cannot be exported to a coverage with the sde2cov command. Use the tools provided in ArcToolbox and ArcCatalog to convert geodatabase annotation to a coverage.

Feature class to ArcSDE entity (feature) type mapping	
Coverage feature class	**ArcSDE entity (feature) type**
point, node	point, node
line	simple line, line (sl)

polygon	area (a)
region.<subclass>	multipart area (a+)
route.<subclass>	multipart lines with measures (slM+)
anno.<subclass>	point, line, simple line, anno (pslA)

Examples

Export multipart polygons to a region coverage

```
$ sde2cov -o create -l world.cntry94,feature -f country,region.cntry
-P single -a all -u world -p world
```

Export a subset of features with a WHERE clause

```
$ sde2cov -o create -l world.cities,feature -f big,point -w
`population 10000000' -a all -u world -p world
```

Export polygons from a multiversioned feature class (layer)

```
$ sde2cov -o create -l world.countries,feature -f country,polygon -V
version10 -P single -a all -u world -p world
```

sde2shp

The **sde2shp** command extracts features from an ArcSDE geodatabase feature class or log file and writes them to an ESRI shapefile.

Usage syntax

From a feature class:

```
sde2shp -o append -l <table,column>
[-V <version_name>]
-f <shape_file> -t <file_type>
[-a {all | file=<file_name>}]
[-w <"where_clause">] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]


sde2shp -o init -l <table,column>
[-V <version_name>]
-f <shape_file> -t <file_type>
[-a {all | file=<file_name>}]
[-w <"where_clause">] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
```

From a log file:

```
sde2shp -o append -L <log_file>
[-V <version_name>]
-f <shape_file> -t <file_type>
[-a {all | file=<file_name>}]
[-w <"where_clause">] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]


sde2shp -o init -L <log_file>
[-V <version_name>]
-f <shape_file> -t <file_type>
[-a {all | file=<file_name>}]
[-w <"where_clause">] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]

sde2shp -h

sde2shp -?
```

Operations

Operation	**Description**
append	Adds the features to existing features in the shapefile If the file does not already exist, it is created.
init	Deletes the shapefile if it exists, then creates a new one

Options

Options	**Description**
-a	**Attribute Modes** *all*: Loads all attribute columns If an attribute table exists, the incoming schema must be union compatible with the shapefile's dBASE table if using the append operation. *file=<file_name>*: File containing lines in the form <sde_column> [shape_column] [type] [width] [nDec] The sde_column selects the column to output. The remaining optional items determine the new attribute column definitions. You can export feature table columns such as spatial_column.FID and spatial_column.AREA. The type, width, and nDec arguments refer to the dBase column format as follows:

description	type	width

	<table><tr><td>Character</td><td>C</td><td>1-254</td></tr><tr><td>Date</td><td>D</td><td>8</td></tr><tr><td>Logical</td><td>L</td><td>1</td></tr><tr><td>Numeric</td><td>N</td><td>1-19</td></tr></table> The number of decimal places can be any positive integer value that is less than the width.
-D	Database name (not supported on Oracle)
-f	Path to and name of the shapefile to add to or create
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-l	Input feature class table and spatial column name These must exist, and the executing user must have read access. If you are not the owner of the table, you must qualify the table name as owner.table.
-L	Name of the log file to extract from
-o	Operation, such as append or init
-p	DBMS user password

-s	ArcSDE server host name (default: localhost)
-t	Shapefile types (names are not case sensitive): **point**: Accepts only point feature **pointZ**: Accepts points, z-values, and measure values (optional) **pointM**: Accepts points and measures **arc**: Accepts line, simple line, and area features. Area features with donuts are accepted as multipart shapefile arc features **arcZ**: Accepts linear features, z-values, and measure values (optional) **arcM**: Accepts linear features and measures **polygon**: Accepts area features **polygonZ**: Accepts area features, z-values, and measure values (optional) **polygonM**: Accepts area features and measures **multipoint**: Accepts points and multipart points **multipointZ**: Accepts points, multipoints, z-values, and measure values (optional) **multipointM**: Accepts points, multipoints, and measure values
-u	DBMS user name
-V	Version name; if specified, uses only the data that belongs to the version, if not specified, uses the default version (default: sde.DEFAULT) Version names are case sensitive; for example, SDE.DEFAULT is a different version from SDE.default.
-w	SQL WHERE clause to limit the features retrieved from the business table or log file

	WHERE clause must be enclosed in double quotes ("").

Conversion table

ArcSDE entity (feature)	Output shape
point	point (If the output shapefile is a multipoint type, each point feature is written out as a multipoint shape with one point.)
lines	arc
simple lines	arc
area	polygon
multipart area	multipart polygon

ArcSDE support for database file types

Type	Width	Description	Storage
B	10	Binary field	No
C	1-254	Character	Yes

D	8	Date Field specified as 8 ASCII characters in YYYYMMDD format	Yes
F	1-20	Numeric floating point field	See below
G	10	General field	No
L	1	Logical field	Yes
M	10	Memo field	No
N	1-19	Numeric fixed position field	Yes
P	10	Picture field	No
V	10	Variable field	No

The dBase F-type column will not be created in the resultant shapefile, but ArcSDE will write to an existing F-type column. Therefore, the sde2shp init operation will not create an F-type column, but the sde2shp append operation will write into a shapefile containing an existing F-type. Conversely, the shp2sde command will read an F-type column.

Discussion

The sde2shp command converts ArcSDE feature classes into shapefiles and can convert up to 15 digits of precision or the maximum precision imposed by the SE_DOUBLE data type.

The sde2shp command does not export rows with NULL geometries. Any row with a NULL geometry will be rejected. You will know if it was rejected because sde2shp reports the number of rows exported and number of rows rejected.

If you need to export the entire data source, you should use sdeexport. When using sdeexport, the NULL geometry and attributes are preserved and the sdeimport command will import the row with a NULL shape.

If the command encounters a feature class column name containing the characters # or -, it returns the error message:

```
Unable to create shape attribute table: Bad column definition
```

In such cases, use the DBMS SQL editor to rename the columns before converting them to a shapefile.

Note: If you have altered column definitions using a SQL interface while the data was stored in an ArcSDE geodatabase, exporting the data may fail. You could instead create a new column with an acceptable name that is of the same data type and length as the old column. Then, in an edit session in ArcMap, you can use the field calculator to copy the values from the old column to the new column. See the topic "Making field calculations" in the ArcGIS Desktop help for details on the field calculator.

Examples

Add features to a shapefile

Use the append operation to add features to an existing shapefile or to create a new shapefile and add features.

```
c:\> sde2shp -o append -l parks,shape -f parks -t polygon -u av
```

Delete an existing shapefile then re-create it

Use the init operation to delete an existing shapefile then convert a feature class to the shapefile. The example below deletes the shapefile census_data, then converts a feature class called blocks to the census_data shapefile.

```
c:\> sde2shp -o init -l blocks,shape -f census_data -t polygon -a
all -u av
```

Create a new shapefile, specifying which features should be extracted from the ArcSDE feature class

You can use the WHERE clause to choose specific features from the ArcSDE feature class you want in the new shapefile. Here, only primary schools are being chosen from an ArcSDE feature class to create a shapefile called elementary.

```
c:\> sde2shp -o append -l schools,shape -f elementary -t point -w
"type=primary" -u av -p secret
```

sde2tbl

The **sde2tbl** command converts ArcSDE geodatabase tables to INFO and dBASE tables. You can also use sde2tbl to selectively copy columns from one ArcSDE geodatabase table to a new ArcSDE geodatabase table.

Usage syntax

From a feature class:

```
sde2tbl -o append -t <table> -f <file_name> -T {dBASE | INFO | SDE}
[-I] [-a {all | file=<file_name>}] [-c <commit_interval>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-v]
[-w <"where_clause">]


sde2tbl -o create -t <table> -f <file_name> -T {dBASE | INFO | SDE}
[-I] [-a {all | file=<file_name>}] [-c <commit_interval>]
[-k <config_keyword>] [-w <"where_clause">]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-v]


sde2tbl -o init -t <table> -f <file_name> -T {dBASE | INFO | SDE}
[-I] [-a {all | file=<file_name>}] [-c <commit_interval>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-v]
[-w <"where_clause">]


sde2tbl -h

sde2tbl -?
```

Operations

Operation	Description
append	Adds records to an existing ArcSDE geodatabase, dBASE, or INFO table (the default)
create	Creates a new table and imports

	records into it; an error is returned if the table already exists.
init	Deletes all records in an existing DBMS table before importing new records

Options

Options	**Description**
-a	**Attribute modes** *all:* Loads all columns (the default) If the table exists, the incoming schema must be union compatible with the table if using the append or init option. *file=<file_name>:* File containing lines in the form <fr_colName> [to_colName] [type] [size] [nDecs] [NOT_NULL] The <fr_colName> selects the column to export, while the <to_colName> specifies the new column in which to load. The allowed type, size, and nDecs (number of decimal places) values will vary according to each DBMS.
-c	Commit rate (default is the AUTOCOMMIT value from giomgr.defs); only used when the output table type is SDE.
-D	Database name (not supported on Oracle)
-f	Output table name; define the table type with the -T option

-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-I	Disable buffered inserts (default: ON); only used when the output table type is SDE.
-k	Configuration keyword from DBTUNE table (default: DEFAULTS); only used when the table is output to an ArcSDE geodatabase
-o	Operation
-p	DBMS user password
-s	ArcSDE server host name (default: localhost)
-t	Input ArcSDE table name
-T	Output table type, either dBASE, INFO, or SDE
-u	DBMS user name
-v	Verbose option; reports records committed at the commit interval
-w	SQL WHERE clause

Discussion

This command converts tables in an ArcSDE geodatabase to INFO and dBase tables. The column types it converts to are as follows:

Type	Width	Description	Storage
B	10	Binary field	No

C	1-254	Character	Yes
D	8	Date Field specified as 8 ASCII characters in YYYYMMDD format	Yes
F	1-20	Numeric floating point field	See below
G	10	General field	No
L	1	Logical field	Yes
M	10	Memo field	No
N	1-19	Numeric fixed position field	Yes
P	10	Picture field	No
V	10	Variable field	No

The dBase F-type column will not be created in the resultant dBase table, but ArcSDE will write to an existing F-type column. The sde2tbl init or create operations will not create an F-type column, but the sde2tbl append operation will write into a dBase table containing an existing F-type. The tbl2sde command will read an F-type column.

The "from" column types vary by DBMS.

It also allows you to selectively copy columns from an ArcSDE geodatabase table into a new ArcSDE geodatabase table. To do this, you use a WHERE clause to specify which columns will be moved to the new table. When copying columns from an existing ArcSDE geodatabase table to another ArcSDE geodatabase table, you can specify a configuration keyword and commit rate, as well as disable buffered inserts.

Note: If you have altered column definitions using a SQL interface while the data was stored in an ArcSDE geodatabase, exporting the data may fail.

Examples

Add records to a table

To add more records to an existing ArcSDE geodatabase, INFO, or dBase table, use the append operation.

```
$ sde2tbl -o append -t hotels.pat -f hotels -T INFO -u av
```

Create a new table

Use the create operation to create a new table. Be sure to specify the type of table created: SDE, INFO, or dBASE. The following example converts an ArcSDE geodatabase table, block_attr, to a dBase table called census_data.

```
$ sde2tbl -o create -t block_attr -f census_data -T dBASE -a all -u
av -p mo
```

Remove the records from an existing table then load more records

Use the init operation to delete the records from an existing table, then import records to the table. The following example removes the records from the table customers then imports new ones from the file marketing for those records with a postal code of 91750.

```
$ sde2tbl -o init -t customers -f marketing -T SDE -w "zip=91750" -u
av
```

sdeexport

The **sdeexport** command creates an ArcSDE export file.

Usage syntax

```
sdeexport [-o create] -t <table> [-V <version_name>] [-O] [-q]
[-a {all | file=<file_name>}]
-f <{export_file | -}> [-X <volume_size>]
[-r <target_ArcSDE_version_number>]
[-w <"where_clause">] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_tser_name> [-p <DB_tser_password>]


sdeexport [-o create] -l <table,column> [-V <version_name>] [-O] [-
q]
[-a {all | file=<file_name>}]
-f <{export_file | -}> [-X <volume_size>]
[-r <target_ArcSDE_version_number>]
[-w <"where_clause">] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_tser_name> [-p <DB_tser_password>]


sdeexport -h

sdeexport -?
```

Operations

Operation	**Description**
create	Creates export format file

Options

Options	**Description**

-a	**Attribute modes** *all*: Loads all columns into the export file *file=<file_name>*: File containing a list of columns to be exported
-D	Database or data source name (not supported on Oracle)
-f	Export format file name; exports to standard output if set to -
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-l	The table and either the spatial column or raster column name
-o	Operation
-O	Export in spatial index order
-p	DBMS user password
-q	Quiet—all titles and warnings are suppressed

-r	The export version number
-s	ArcSDE server host name (default: localhost)
-t	Table to be exported
-u	DBMS user name
-V	Version name; if specified, uses only the data that belongs to the version, if not specified, uses the default version (default: sde.DEFAULT) Version names are case sensitive; for example, SDE.DEFAULT is a different version from SDE.default.
-w	SQL WHERE clause
-X	Maximum file size in bytes, kilobytes, megabytes, or gigabytes for each export file volume The upper limit is dependent on the O/S file size. The minimum volume size that may be specified with the -X option is 4096 bytes or 4K. When specified, the export files are created with the extensions .000 through .999.

Discussion

The sdeexport command creates an export format that can be imported by the sdeimport command or examined with the sdexinfo command.

If you intend to import the file into an earlier version of ArcSDE, you must use the -r option and specify the version. For example, if you want to export from ArcSDE 9.0 and then import to ArcSDE 8.3, you must specify -r 8.3.

The -f option specifies the output file to which you want to export. You can enable file volumes by specifying the -X option. When file volumes are enabled, the extension .000 is appended to the file name specified at the -f option. When this file volume fills to the limit specified by the -X option, a new file with the extension .001 is created and sdeexport continues to export data to this file. Up to 1,000 file volumes may be created with extensions from .000 through .999 given that enough space exists. The minimum volume size that may be specified with the -X option is 4,096 bytes, or 4K (kilobytes). The maximum volume size is determined by the limits imposed by the file system where the volume is being created. See your system documentation for the appropriate limits. The file volume sizes are specified in bytes, but they may also be specified in either kilobytes, megabytes, or gigabytes by appending the modifiers K, M, or G to the integer value.

The -V option specifies the version of the table or feature class to be exported. Versions can be created with either the ArcGIS Desktop applications or the sdeversion command. The base table or feature class and all edits unique to that version are exported.

The -O option allows you to create an export file in spatial index order. You should be aware that ArcSDE issues a SQL ORDER BY clause on the spatial index and, therefore, uses DBMS sort space to sort the records prior to writing the data to the export file. For this option to work, you may need to increase the size of the DBMS sort space.

Note: If you have altered column definitions using a SQL interface while the data was stored in an ArcSDE geodatabase, exporting the data may fail.

Examples

The following sdeexport command creates file volumes that are no larger than 500 megabytes, named roadex.000, roadex.001, and so on.

```
$ sdeexport -l roads,shape -f roadex -X 500M -u gisuser
```

Note: The create operation is optional.

You may direct sdeexport to export to standard output by specifying the - with the -f option. This is useful if you want to pipe standard output to a device such as a tape device. For example, you could pipe standard output to the UNIX dd command that is writing to the tape device.

```
$ sdeexport -l roads,shape -f - -u gisuser | dd of=/dev/rmt/Ocn
bs=16k
```

You may export either a business table by specifying the -t option or a feature class, raster catalog, or raster dataset by specifying the -l option. If the -t option is used to export a business table that contains a spatial column or a raster column, all the dependent tables are exported as well. In fact, if the -t option is used to export a business table containing a spatial column, the resulting export file is equivalent to one created by the -l option specifying the table and spatial column of a feature class. The following examples would create export files that contain the same data, assuming that the roads business table specified by the -t option contains the spatial column shape. Here, the feature class faults has a spatial column called shape, so it can be exported with -l or -t.

```
c:\> sdeexport -l faults,shape -f c:\GISData\faultex -u dataowner
```

or

```
c:\> sdeexport -t faults -f c:\GISData\faultex -u dataowner
Password:

ArcSDE 9.2 for SQL Server Build 721 Mon Aug 1 16:00:45 2005
SDEX File Export Administration Utility
----------------------------------------------------------------
Exporting ArcSDE object to "c:\GISData\faultex" in SDEX 9.0 export
format ...
Exporting table "faults".
Spatial column "SHAPE"
18239 features converted.
18239 features exported.
```

You may specify only the records of a specific version be exported by including the -V option. Only the records of the base table and the edits unique to the states of the specified version are included in the sdexport file.

```
c:\> sdeexport -t roads -f roadex_newpavement -u gisuser -V
newpavement
```

ArcSDE Administration Commands

sdegroup

The **sdegroup** command merges features by combining their geometries into multipart shapes. Features are grouped by tiles or by a business table attribute.

Usage syntax

```
sdegroup -o append -S <source_layer,spl_column>
[-s <server_name>] -T <target_layer,spl_column>
[-D <database>] [-c <commit_interval>]
[-i <service>] [-a {none | all | file=<file_name>}]
[-k <config_keyword>] [-w <"where_clause">]
-u <DB_user_name> [-p <DB_user_password>]
[-N] [-q] [-t <Tile size> | column=<col_name> ]


sdegroup -o create -S <source_layer,spl_column>
[-s <server_name>] -T <target_layer, spl_column>
-e <entity_type> [-a {none | all | file=<file_name>}]
[-D <database>] [-c <commit_interval>] [-i <service>]
[-k <config_keyword>] [-w <"where_clause">] [spatial index]
-u <DB_user_name> [-p <DB_user_password>] [-q]
[-t <Tile_size> | column=<col_name>]


sdegroup -o init -S <source_layer,spl_column> [-s <server_name>]
-T <target_layer,spl_column> [-D <database>]
[-c <commit_interval>] [-i <service>]
[-a {none | all | file=<file_name>}]
[-k <config_keyword>] [-w <"where_clause">]
-u <DB_user_name> [-p <DB_user_password>]
[-N] [-q] [-t <Tile_size> | column=<col_name>]
```

Where

```
[Spatial_Index] := [-g {<grid_size1> [,<grid_size2> [,<grid_size3>]]
|
GRID,<grid_1>[,<grid_2>[,<grid_3>]] |
NONE |
RTREE |
DBTUNE |
FIXED,<sdo_level> |
HYBRID,<sdo_level>,<sdo_num_tiles>}]


sdegroup -h
sdegroup -?
```

Operations

Operation	Description
append	Adds features from a source feature class to any existing target feature class
create	Creates a new feature class and imports features into it from a source feature class based on a grouping of shapes to a single feature
init	Deletes all features from an existing target feature class before importing the grouped features from the source feature class

Options

Options	Description
-a	**Attribute modes** *none*: Do not load any attributes (default). *all*: Load all attribute columns. If no attribute table exists for the feature, one will be created. If one exists, incoming schema must be union compatible with the existing table if the append option is used. *file=<file_name>*: File containing lines in the form <shape_column> [sde_column] The shape_column selects the column to be output; the

	sde_column is the new name to be specified.
-c	Commit rate
-D	Database or data source name (not supported Oracle databases)
-e	Entity types allowed (psla+) If your source feature class has multiple shape types but you restrict the entity type to one shape type, only that shape type is copied to the target feature class. Valid entity types are: p: Point shapes s: Line (spaghetti) shapes l: Simple line (line string) shapes a: Area shapes +: Multipart shapes For example, if the source feature class has lines and area shapes, but you use -e a+, only area shapes will be grouped and written to the new feature class.
-g	Spatial index type and parameters

<grid1,grid2,grid3> *or* GRID,<grid_sz1>[,<grid_sz2>[,<grid_sz3>]]	Creates multilevel grid index (Grid2 and grid3 are optional.)
DBTUNE	Uses DBTUNE parameters to index
NONE	No spatial index is created

RTREE	Creates Rtree index
FIXED,<level>	Creates fixed Quadtree index
HYBRID,<level, n_tiles>	Creates hybrid Quadtree index

-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-k	Configuration keyword present in DBTUNE table (default: DEFAULTS)
-N	No verification is performed; the operation begins immediately after being invoked.
-o	Operation
-p	DBMS user password
-q	Quiet—all titles and warnings are suppressed.
-s	ArcSDE server host name (default: localhost)
-S	Source feature class and spatial column names If you are not the owner of the table, you must qualify the table name as owner.table.
-t	Either the size of the tiles (default: 2) or the name of a column containing the values on which the features are to be grouped
-T	Target feature class and spatial column names If you are not the owner of the table, you must qualify the table

	name as owner.table.
-u	DBMS user name
-w	SQL WHERE clause The WHERE clause is used to limit the records/values from the source table that are brought into the target feature class.

Discussion

The sdegroup command uses either tiles or a column value to group shapes into new multipart shapes. Using sdegroup can improve performance and display speed by reducing the number of rows that must be fetched from the database. However, you will lose the attribute data. The new table will be populated with the attribute values of the first feature for each new feature.

If you are grouping on a tile size, use a common viewing or query area to determine the tile size. If clients usually want to see an area of 1,600 square miles (40 by 40 miles) on a feature class that's 40,000 square miles (200 by 200 miles), you might want to use a tile size of 40. This will group the original features into 25 new features. Remember to consider the unit of measure of the feature class. If the feature class's unit of measure is feet, you would use 211,200 for the previous example.

Grouping on a column is usually done to combine all the features that have the same value into a single feature. For instance, if a feature class containing roads contains roads that are segmented into many line strings, roads with the same name could be combined into a single multiline string by grouping on a NAME column.

If the spatial index of the target and source feature class is a grid, you should increase the size of the grid cells, because you're reducing the number of features in each grid. How much depends on the size of the resulting grouped features. Run sdelayer -o si_stats to obtain grid statistics of the resulting target feature class, and determine if an adjustment to the grid cell size is necessary.

Examples

Add grouped features from a source feature class to an existing feature class

Use the append operation to bring additional grouped features into an existing feature class.

```
$ sdegroup -o append -S arroyos,shape -T rivers,shape -u av
```

Create a new feature class containing grouped features

The create operation is used to make new grouped feature classes.

Polygon and multipolygon features are grouped into multiline string features. This is done to avoid the loss of topology that would occur if polygon or multipolygon features were grouped into multipolygon features. Topology would be lost because multipolygons cannot share a common boundary. The common boundaries of grouped polygons and multipolygons would be dissolved.

To create a new feature class, target_cities, from the original cities feature class, use the following command syntax:

```
$ sdegroup -o create -S cities,shape -T target_cities,shape -u av -p
mo -t 1 -e p+ -g 20,0,0 -i esri_81
```

The tile size is 1, and only points will be copied from the cities feature class.

In this example, the new feature class, target_roads, combines line strings from the source feature class roads into multiline strings based on their name:

```
$ sdegroup -o create -S roads,shape -T target_roads,shape -u av -p
mo -t column=name -e sl+ -g 20,0,0 -i esri_81
```

The next example creates the target feature class grp_cities with values from the source feature class cities. It uses a WHERE clause to restrict the values from the cities.shape column, only bringing in those with values greater than 2.

```
c:\> sdegroup -o create -S cities,shape -T grp_cities,shape -w
"shape > 2" -e p+ -a all -t 2
```

Delete the contents of an existing feature class then add new grouped features

The init operation will delete the features in an existing feature class, then repopulate the feature class with grouped features from the specified source file.

```
c:\> sdegroup -o init -S schools,shape -T elem_schools,shape -u av
```

ArcSDE Administration Commands

sdeimport

The **sdeimport** command imports data from an ArcSDE export file.

Usage syntax

```
sdeimport -o append {-l <table,column> | -t <table>} [-V
<version_name>]
-f <{export_file | -}> [-q] [-v] [-c <commit_interval>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>]
```

```
sdeimport -o create {-l <table,column> | -t <table>} [-V
<version_name>]
-f <{export_file | -}> [-v] [-L <ON | OFF> ] [-P <HIGH | LOW>]
[Spatial_Index] [-M <minimum_ID>] [-k <config_keyword>]
[-c <commit_interval>] [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]
```

```
sdeimport -o delete {-l <table,column> | -t <table>} -K
<key_columns>
[-V <version_name>] -f <{export_file | -}> [-q] [-v]
[-c <commit_interval>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
```

```
sdeimport -o init {-l <table,column> | -t <table>} -f <{export_file
| -}> [-v]
[-c <commit_interval>] [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>]
```

```
sdeimport -o update {-l <table,column> | -t <table>} -K
<key_columns>
[-V <version_name>] -f <{export_file | -}> [-q] [-v]
[-c <commit_interval>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
```

```
sdeimport -o update_else_insert {-l <table,column> | -t <table>} -K
<key_columns>
[-V <version_name>] -f <{export_file | -}> [-q] [-v]
[-c <commit_interval>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
```

Where

```
[Spatial_Index] := [-g [Grid_Options] | GRID,[Grid_Options] |
AUTOMATIC | NONE | RTREE ]

[Grid_Options] : = [<grid_sz0>[,<grid_sz1>[,<grid_sz2>]] [,][{FULL |
SPARSE}]


sdeimport -h

sdeimport -?
```

Operations

Operation	Description
append	Imports and appends records into an existing table, feature class, or raster layer (default)
init	Deletes all features of an existing table, feature class, or raster layer before importing new records
create	Creates a new table, feature class, or raster layer based on the definition stored in the export file and imports records into it; an error is returned if the dataset already exists.
delete	Deletes the records from the target table, feature class, or raster layer that match the key column values of the export file

update	Updates the records in the target table, feature class, or raster layer that match the key column values of the export file
update_else_insert	Updates the records of the target table, feature class, or raster layer that match the key column values of the export file and inserts into the target the records of the export file whose key values do not match

Options

Options	**Description**
-c	Commit rate (default = the AUTOCOMMIT parameter value in the SDE.SERVER_CONFIG table)
-D	Database or data source name (not supported on Oracle databases)
-f	Export format file name; if set to -, import is from standard input
-g	Spatial index type and parameters GRID,<grid_sz1>[,<grid_sz2>[,<grid_sz3>]],[FULL|SPARSE] Creates multilevel grid index (Grid2 and grid3 are optional.) If you don't

		enter grid values, it is the same as specifying an AUTOMATIC spatial index type. FULL grids create a spatial index grid on the entire feature envelope. This is the default option if GRID is specified and is the type of grid always created in ArcSDE 9.1 and lower. SPARSE grids create spatial index grids only where grids actually include parts of the feature. See the Discussion section for more details.
	AUTOMATIC	For Oracle Spatial and Informix, this is RTREE. For all other storage types, if the layer is in normal IO mode and there are 0 features, the grid type is NONE. If one

		or more features is present and the layer is in normal IO mode, grid sizes will be calculated and set based on the calculations below. If the layer is in load_only IO mode, grid sizes will be calculated and set based on the calculations below when the layer is returned to normal IO mode.
		
		If there are no valid envelopes detected, grid sizes are calculated as:
		
		grid_size1 = 1000000.0 / layerGrid->xyUnits
		grid_size2 = 1000000000.0 / layerGrid->xyUnits
		
		If it is a point layer, grid sizes are calculated as:
		
		grid_size1 = 1000.0 / layerGrid-

	<table><tr><td></td><td>>xyUnits</td></tr><tr><td>NONE</td><td>No spatial index is created</td></tr><tr><td>RTREE</td><td>Creates Rtree spatial index (Informix, Oracle Spatial)</td></tr></table>
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-k	Configuration keyword present in DBTUNE table (default: DEFAULTS); used with the create operation only.
-K	The list of columns used as a key to locate matching records for the delete, update, or update_else_insert operations
-l	Either the feature class business table and spatial column or raster layer's business table and raster column If you are not the owner of the table, you must qualify the table name as owner.table and have the necessary privileges for that table.
-L	Turns on or off autolocking (default: ON)
-M	Minimum feature ID New feature or raster IDs are assigned the larger of the minimum ID or the maximum assigned ID + one. (default: 1)
-o	Operation
-p	DBMS user password

-P	Layer coordinate precision: LOW (32-bit) or HIGH (64-bit) If the -P option isn't specified, sdeimport loads data in the precision specified by the SERVER_CONFIG DEFAULTPRECISION setting. The default setting of DEFAULTPRECISION is 64 (HIGH). If you attempt to place high precision data in a low precision, you will receive the following warning message: WARNING: Loading high precision data into a low precision layer. Might encounter invalid coordinate range error.
-s	ArcSDE server host name (default: localhost)
-t	Business table name
-u	DBMS user name
-v	Verbose option—reports records committed
-V	Version name (default: sde.DEFAULT) Inserts the data into the version specified or, if not specified, inserts the data into the default version. Not allowed for the init operation. Version names are case sensitive. For example, SDE.DEFAULT is a different version from SDE.default.

Discussion

ArcSDE export files created by the sdeexport command are imported using the sdeimport command. The sdeexport command can create exports of records from a table, feature class, or raster layer. If you are importing data from a later release of ArcSDE, you would have needed to specify a version using the -r option for the export file when you created it. For example, if you want to export data from ArcSDE 9.0 and then import it to an ArcSDE 8.3 geodatabase, you must specify -r 8.3 when you run sdeexport. The format version as well as other metadata is stored in the header of the export file.

Note: If the 3.0.1 format was specified when the export file was created, only the records of the feature class can be exported.

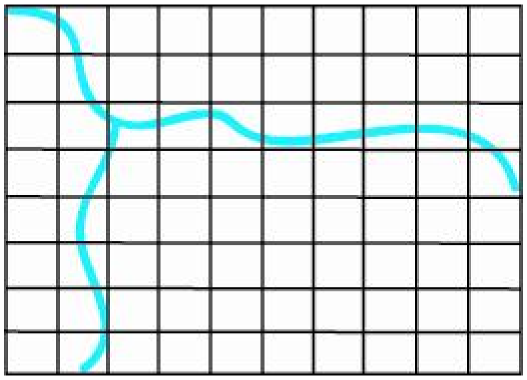
The create operation allows you to create a new table, feature class, or raster layer. When you use the create operation, if the object specified by the -t option already exists, the sdeimport command returns an error. A configuration keyword can be specified with the -k option, allowing you to control the storage within the DBMS. If the -k option is not specified, the DEFAULTS keyword is used.

The sdeimport command depends on the type of records the export file contains to create the appropriate object. If the export file contains feature class records and the

-t option is specified to create a table from the export file, a feature class will be imported. If the -l option is specified to create a feature class or raster layer from an export file that contains table records, the sdeimport command will return an error.

If the -l option is specified with any operation other than create, the spatial reference stored in the export files header must match the spatial reference of either the feature class or the raster layer.

Beginning with ArcGIS Server 9.2 geodatabases, you have the option to create sparse spatial index grids. Sparse grids populate the spatial index with information for only those grids that actually contain a portion of a feature. For example, if you created a spatial index on a river feature class, only those grids that the river crossed would be indexed for the river.



This could be more efficient for spatial queries, because grids that do not qualify as crossing the feature (in this case, rivers) will be eliminated from the query results during the primary filter process ArcSDE performs when executing spatial queries. Be aware that if you use third-party applications to perform envelope-on-envelope spatial queries, using a sparse index grid will affect query results.

If instead you used a full grid for the rivers, the nonqualifying grids would still be eliminated, but they wouldn't be eliminated until the secondary filter process.

Examples

Add records to an existing table, feature class, or raster layer

The append operation allows you to add records to an existing table, feature class, or raster layer. If the target does not exist, sdeimport returns an error.

In this example, the append operation adds the records of the usa export file to the world feature class.

```
$ sdeimport -o append -l world,shape -f usa -u admin -p passwd
```

Create a new table, feature class, or raster layer

The create operation creates a new table, feature class, or raster layer based on the definition stored in the export file then imports records into it from the source file. If a layer or table of the same name already exists, an error is returned.

In this example, the sdeimport command creates the table customer_names in the DBMS from the table description stored in the header of the custname export file. The records of the custname export file are loaded into the customer_names table.

```
$ sdeimport -o create -t customer_names -f custname -k smalltab -g
GRID,FULL -u admin -p passwd
```

The following example shows an import of an export set created in volumes:

```
$ sdeimport -o create -t countries.000 -f countries -g GRID,FULL -u
admin -p passwd
```

The example below shows the creation of a HIGH precision layer with a sparse index grid:

```
c:\> sdeimport -o create -t rivers -f rivers -g GRID,SPARSE -P HIGH
-u loader -p down
```

This example shows what happens when you create a LOW precision layer with a HIGH precision source file:

```
% sdeimport -o create -t markets_low -f compete.sdx -P LOW -v -u
editor -p changes

ArcSDE 9.2 for Oracle10g  Mon Mar 27 16:01:07 2006
SDEX File Import Administration Utility
-------------------------------------------------------
Importing SDEX from compete.sdx ...
Importing spatial column "SHAPE"
WARNING:
Loading high precision data into a low precision layer.
Might encounter invalid coordinate range error.

6 records processed
6 records read.
6 records stored.
```

Note that to receive this warning, you need to have specified the verbose (-v) option, as shown above.

Delete records

The delete operation allows you to remove records from an existing target. The records of the target whose key column values match those of the export file are deleted. The key columns are specified by the -K option. The key columns must exist in both the export file and the target; otherwise, an error is returned.

In this example, the delete operation removes those records from the usa feature class that have fips column values matching those of the california export file.

```
$ sdeimport -o delete -l usa,shape -f california -K fips -u admin -p
passwd
```

Delete existing records and reinsert records

The init operation allows you to delete the existing records of the target before inserting the records. If the target does not exist, an error is returned. The target business table definition must match the definition of the exported records.

In this example, the init operation deletes the records of the montana raster layer and inserts the records from the montana export file.

```
$ sdeimport -o init -l montana,image -f montana -u admin -p passwd
```

Update records

The update operation allows you to update the records of the target. The target records possessing key column values that match those of the export file are updated with the records stored in the export file.

In this example, the update operation replaces the records of the usa feature class with the records from the export file that match the fips key column value.

```
c:\> sdeimport -o update -l usa,shape -f nebraska -K fips -u admin -
p passwd
```

Update matching records and insert nonmatching records

The update_else_insert operation allows you to either update or insert the records of the target. If the target key columns match the values of the sdeexport files, the records are updated. For those records of the export file for which a match is not found, the records are inserted into the target.

In this example, the update_else_insert operation replaces the records of the usa feature class with the records of the texas export file, which has matching fips key column values. Those records of the texas export file that do not match on the fips key column are inserted in the usa feature class.

```
$ sdeimport -o update_else_insert -l usa,shape -f texas -K fips -u
admin -p passwd
```

sdelayer

The **sdelayer** command administers feature classes.

Usage syntax

```
sdelayer -o add -l <table,column> -e <entity_mask>
[Spatial_Index] [{-R <SRID> | [Spatial_Ref_Opts]}]
[-M <minimum_id>] [{-f <init_features,avg_points> | -k
<config_keyword>}]
[-E <{empty | xmin,ymin,xmax,ymax}>] [-t <storage_type>]
[-L {ON | OFF}] [-C <row_id_column>[,{SDE|USER}[,<min_ID>]]]
[-P {HIGH | BASIC}][-S <layer_description_str>] [-i <service>]
[-s <server_name>] [-D <database>] -u <DB_user_name>
-p <DB_user_password> [-q]
```

```
sdelayer -o alter -l <table,column> [-e <entity_mask>] [-M
<minimum_id>]
[-S <layer_description_str>]
[-k <config_keyword>] [-i <service>] [-s <server_name>]
[-D <database>] [Spatial_Index] [-L <ON | OFF>]
[-E <{empty | calc | xmin,ymin,xmax,ymax}>]
[-G {<projection_ID> | file=<proj_file_name}>]
[-P HIGH[{-R <SRID>|[Spatial_Ref_Opts]}]]
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]
```

```
sdelayer -o delete -l <table,column> [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]
```

```
sdelayer -o {describe | describe_long} [{-O <owner | -l
<table,column>}]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]
```

```
sdelayer -o feature_info -l <table,column> [-V <version>]
[-r {valid | all | invalid}] [-w <"where_clause">] [-c]
[Spatial_Index] [-S <layer_description_str>] [-q]
[-i <service>] [-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>] [-q]
```

```
sdelayer -o {grant | revoke} -l <table,column> -U <user>
-A <SELECT,UPDATE,INSERT,DELETE>
[-i <service>] [-s <server_name>] [-D <database>]
```

```
-u <DB_user_name> [-p <DB_user_password>] [-I] [-q]


sdelayer -o list -l <table,column> -v <shape_id> [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdelayer -o {load_only_io | normal_io} -l <table,column> [-i
<service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdelayer -o register -l <table,column> -e <entity_mask>
{[-C NONE] | [-C <row_id_column>[,{SDE|USER}[,<min_ID>]]]}
[Spatial_Index] [{-R <SRID> | [Spatial_Ref_Opts]}]
[-S <layer_description_str>] [-k <config_keyword>]
[-i <service>] [-s <server_name>]  [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdelayer -o {stats | si_stats} -l <table,column> [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> -p <DB_user_password> [-q]


sdelayer -o truncate -l <table,column> [-i <service>]
[-s <server_name>] [-D <database>]
-u < DB_user_name> [-p <DB_user_password>] [-N] [-q]
```

Where

```
[Spatial_Ref_Opts] := [-x <xoffset,yoffset,xyscale>]
[-z <zoffset,zscale>] [-m <moffset,mscale>]
[-G {<projection_ID> | file=<proj_file_name>}]

[Spatial_Index] := [-g [Grid_Options] | GRID,[Grid_Options] |
AUTOMATIC | NONE | RTREE ]

[Grid_Options] : = [<grid_sz0>[,<grid_sz1>[,<grid_sz2>]] [,][{FULL |
SPARSE}]


sdelayer -h


sdelayer -?
```

Operations

Operation	**Description**
add	Adds a spatial column to a business table, creating a feature class
alter	Modifies some characteristics of a feature class
delete	Deletes an entire feature class All features and the feature class definition are permanently removed. The business table is not deleted.
describe	Lists feature class definitions (short form) If a feature class is not specified, lists all feature classes to which the user has access.
describe_long	Lists feature class definitions (long form) If a feature class is not specified, lists all feature classes to which the user has access.
feature_info	Reports information about the feature, such as shape validity, measurements and extent information, the FID, the presence of annotation, whether the feature contains CAD data, the presence of inclusions or cojoined inner rings, the minimum precision of the layer, and the number of points, parts, and subparts contained in the feature

grant	Grants a user access to a specified feature class
list	Lists the fields of a feature including the point values for all points that define the shape of the feature
load_only_io	Sets the I/O mode of the feature class to load-only, allowing only store and replace I/O operations
normal_io	Sets the I/O mode of the feature class back to normal I/O mode
register	Registers a feature class having a spatial column defined using Oracle Spatial Geometry Type, Informix Spatial DataBlade, DB2 Spatial Extender, or Spatial Type for Oracle Registering a feature class with ArcSDE registers the business table in the TABLE_REGISTRY system table.
revoke	Revokes access to a feature class from a user
stats	Reports feature class statistics
si_stats	Reports spatial index statistics (Oracle and SQL Server)
truncate	Deletes all features in the feature class; the feature class definition remains.

Options

Options	**Description**

-A	Type of access: SELECT, UPDATE, INSERT, or DELETE
-c	Specifies whether area, length, and coordinate extrema should be calculated and output
-C	Specifies the name of the row ID column of the layer's business table and the optional minimum row ID value For the register operation, it also specifies whether the row ID is to be registered as user or ArcSDE maintained. If ArcSDE maintained, you can specify the starting value (min_ID) for the row ID. If no minimum ID is specified, it defaults to 1.
-D	Database name (not supported on Oracle)
-e	Entity types allowed (npslaAc3+M) n Nil p Point features s Line (spaghetti) features l Simple line (line string) features a Area features A Annotation c CAD data 3 Three-dimensional features, which can be added to the entity type mask only with the -o add operation + Multipart features M Measures on coordinates

	The -m option is required to store measure values on each coordinate. Measures can be added to the entity type mask only with the -o add operation. Adding or registering a new entity type allows users to store features of the added entity type. (Some applications, such as ArcGIS, do not recognize feature classes that contain a mixture of point, line, and polygon features.) Using sdelayer to remove a previously allowed entity type will permanently delete all feature rows of that entity type.
-E	Feature class envelope There are three options: empty Sets the feature class envelope to an empty envelope calc Calculates the feature class envelope xmin,ymin,xmax,ymax Sets the envelope to the specified values
-f	(Oracle only) Initial number of features and the average number of points per feature These are used to calculate the initial and next extent of the feature class's F and S tables. You cannot use this option with the -k option.
-g	Spatial index type and parameters GRID,<grid_sz1>[,<grid_sz2>[,<grid_sz3>]],[FULL|SPARSE] Creates multilevel grid index (Grid2 and grid3 are optional.) If you don't enter grid values, it is the same as specifying an AUTOMATIC spatial index type.

		FULL grids create a spatial index grid on the entire feature envelope. This is the default option if GRID is specified and is the type of grid always created in ArcSDE 9.1 and lower. SPARSE grids create spatial index grids only where grids actually include parts of the feature. See the Discussion section for more details.
	AUTOMATIC	For Oracle Spatial and Informix, this is RTREE. For all other storage types, if the layer is in normal IO mode and there are 0 features, the grid type is NONE. If one or more features is present and the layer is in normal IO mode, grid sizes will be calculated and

	set based on the calculations below. If the layer is in load_only IO mode, grid sizes will be calculated and set based on the calculations below when the layer is returned to normal IO mode.
	
	If there are no valid envelopes detected, grid sizes are calculated as:
	
	grid_size1 = 1000000.0 / layerGrid->xyUnits
	grid_size2 = 1000000000.0 / layerGrid->xyUnits
	
	If it is a point layer, grid sizes are calculated as:
	
	grid_size1 = 1000.0 / layerGrid->xyUnits

NONE	No spatial index is created
RTREE	Creates Rtree index

-G	Coordinate system specifier When used with the alter operation, it changes the feature class' metadata, not the data itself. <projection_id>coordinate system ID (see the pedef.h file for the integer codes) file=<proj_file_name>file containing coordinate system description string
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-I	Inherit grant privilege The grant option is included with the granted privilege. For example, if user A grants user B update privileges on a table, the -I option indicates that user A also wants to grant user B the ability to grant other users update privileges on a particular table.
-k	Configuration keyword present in DBTUNE table (default: DEFAULTS) The storage parameters specific to the feature class will be found under the specified keyword. You cannot use this option with the -f option.
-l	The business table and its spatial column If you are not the owner of the table, you must qualify the table name as owner.table.
-L	Controls the feature class's autolocking of features by area locks Set to ON, layer autolocking is enabled; set to OFF, layer autolocking is disabled. When enabled, layer autolocking attempts to lock the area encompassed by a feature's envelope. If the area is not already locked by another session, the area is locked and the feature can be edited (inserted, updated or deleted). If the area is already locked by another session, an error is returned. By default, feature class autolocking is enabled. Even if a feature class's autolocking property is enabled, the following conditions must be true before autolocking will occur. 1. The LAYERAUTOLOCKING server configuration parameter must be set to TRUE. 2. The feature class must be in NORMAL_IO mode.

	3. The feature class's business table must not be multiversioned.
-m	Measure offset and scale, separated by a comma
-M	Minimum feature ID New feature IDs are assigned the larger of the minimum ID or the maximum assigned ID + one.
-N	No verification is performed; the operation begins immediately after being invoked
-o	Operation
-O	The owner of the feature class
-p	DBMS user password
-P	Specifies whether the geometry of the feature class is to be stored with BASIC (32-bit) or HIGH (64-bit) precision (default: geometry is stored with HIGH precision) The precision cannot be altered after the feature class is created.
-q	Quiet—all titles and warnings are suppressed.
-r	Specifies if only valid shapes are read or all shapes are read If all shapes are read, an error is returned to indicate invalid shapes.
-R	Specifies the ArcSDE spatial reference ID (SRID) and the spatial reference options: -x <xoffset,yoffset,xyscale> -z <zoffset,zscale> -m <moffset,mscale> -G {<projection_ID> | file=<proj_file_name>}
-s	ArcSDE server host name (default: localhost)
-S	Map layer description string (up to 63 characters)
-t	Storage type, which, if included during the add operation, overrides the DBTUNE GEOMETRY_STORAGE parameter Valid values include: B: ESRI binary (Oracle Long Raw and SQL Server)

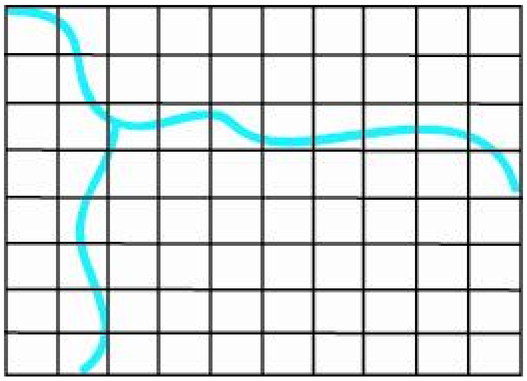
	S: Oracle Spatial (Oracle) Q: Spatial data type (DB2, Informix, and Spatial Type for Oracle) W: Well-Known Binary L: LOB (Oracle BLOB) N: Normalized (obsolete)
-u	DBMS user name
-U	The DBMS user for whom feature class access is granted or revoked
-v	Feature ID
-V	Version name (default: sde.DEFAULT) Version names are case sensitive. For example, SDE.DEFAULT and SDE.default are different versions.
-w	SQL WHERE clause
-x	The x- and y-offsets and scale values, separated by commas (default: 0.0, 0.0, 1.0)
-z	The z-offset and scale values, separated by a comma

Discussion

Before creating a new feature class, be sure there is enough space available on your DBMS.

Calculate the x-, y-, and z-scale values for the feature class by using the size of the service area and how much resolution is needed for the data. Coordinates are truncated if the scales are too small. For example, if the scale is set to 100 for coordinates in meters, the unit of resolution is centimeters. To set the unit of resolution to millimeters, increase the scale to 1,000.

Beginning with ArcGIS Server 9.2 geodatabases, you have the option to create sparse spatial index grids. Sparse grids populate the spatial index with information for only those grids that actually contain a portion of a feature. For example, if you created a spatial index on a river feature class, only those grids that the river crossed would be indexed for the river.



This could be more efficient for spatial queries, because grids that do not qualify as crossing the feature (in this case, rivers) will be eliminated from the query results during the primary filter process ArcSDE performs when executing spatial queries. Be aware that if you use third-party applications to perform envelope-on-envelope spatial queries, using a sparse index grid will affect query results.

If instead you used a full grid for the rivers, the nonqualifying grids would still be eliminated, but they wouldn't be eliminated until the secondary filter process.

Examples

Given this feature class specification:

DBMS table name: victoria
DBMS column name: parcels
Feature types: multipart polygons and annotation
Location: victoria
Spatial index grid: Level 1 - 1000 meters
xy offset: 0,0
xy scale: 200
z offset: 0
z scale: 200
precision: 32-bit (BASIC)
autolocking: disabled
Coordinate system: 28355 (GDA1994 zone 55) Geographic extent

Create a new feature class

Use the add operation to create a new feature class by adding a spatial column to a table. This must be done by the owner of the table.

```
   $ sdelayer -o add -l victoria,parcels -e aA3+ -g GRID,1000 -x
   0,0,200 -z 0,200 -P BASIC -L off -G 28355 -u av -p mo -i esri_80
```

The x,y offsets and scale are optional. If unspecified, they default to 0, 0, 1.

Alter properties of the feature class

Using the alter operation of sdelayer, you can modify the entity mask, spatial index grid cell sizes, coordinate system ID, description, envelope, or configuration keyword. The example below modifies the entity mask, configuration keyword, coordinate system ID, and description.

```
$ sdelayer -o alter -l victoria,parcels -e al+ -k vict2 -G 4326 -S
"Victoria Parcels" -u av -p mo -i esri_40
```

This example changes the grid size:

```
$ sdelayer -o alter -l victoria,parcels -g GRID,2000 -u av -p mo -i
esri_40 -N
```

The -N option suppresses the prompt to verify the change in grid size.

The ArcSDE for Informix product implemented an Rtree index as its spatial index. Changing the grid parameters with the -g option has no effect on a feature class in an Informix database since it does not have a grid index. The Rtree index can be created and dropped but not altered.

This example alters a layer's (blocks) precision to high precision and specifies the x,y, z-, and m- offsets and scales, and the projection file to use.

```
c:\> sdelayer -o alter -l blocks,shape -P HIGH -x 1,1,1 -z 0,0 -m
0,0 -G file=c:\blocks.prj
```

Delete a feature class

For feature classes that have been implemented using the binary schema (Oracle and SQL Server only), deleting a feature class drops the DBMS spatial tables (the F and S tables) and removes the definition of the feature class from the ArcSDE data dictionary tables.

Under the spatial types (UDT) and functions implementation, the spatial column is made null, and the feature class definition is removed from the ArcSDE data dictionary tables. The business table still exists and is registered. To completely remove the business table, use sdetable -o delete. You should also use sdetable -o delete to delete views, NOT sdelayer -o delete.

If a user has the feature class locked, sdelayer will return an error. You can use the sdemon info operation to make sure the feature class is not locked. The sdemon kill operation can remove the process that locked the feature class. This is an abrupt solution and should only be used when necessary. If circumstances permit, you should inform all users that the feature class is going to be removed and provide application programmers with enough time to remove references to the feature class from their programs. The sdelayer command is not geodatabase aware, therefore, references to feature classes that participate in feature datasets, networks, and topologies will not be removed. Such feature classes should be removed using ArcCatalog.

Use the delete operation to delete the feature class:

```
$ sdelayer -o delete -l victoria,parcels -u av -p mo -i esri_80
```

Describe a feature class

You can list a feature class definition by using the sdelayer command with the describe or describe_long operations. All fields that define a feature class are displayed on the screen. To list all available feature classes to which the current user has access, don't specify a feature class name, as shown below:

```
$ sdelayer -o describe -u av -p mo -i esri_80

Layer Administration Utility
-----------------------------------------------------

Table Owner : AV
Table Name : BORDERS
Spatial Column : FID
Layer id : 1
Entities : a
Layer Type : SDE
I/O Mode : NORMAL
User Privileges : SELECT, UPDATE
Layer Configuration : DEFAULTS
-----------------------------------------------------

Table Owner : AV
Table Name : BOUNDARIES
Spatial Column : BOUNDARY
Layer id : 2
Entities : a
Layer Type : SDE
I/O Mode : NORMAL
User Privileges : SELECT, UPDATE, INSERT
Layer Configuration : DEFAULTS
-----------------------------------------------------

Table Owner : AV
Table Name : MINOR_ROADS
Spatial Column : ROAD_LAYER
Layer id : 3
Entities : s
Layer Type : SDE
I/O Mode : NORMAL
User Privileges : SELECT
Layer Configuration : DEFAULTS
-----------------------------------------------------
```

To display the definition for a particular feature class, specify the table and column name with the -l option.

```
$ sdelayer -o describe -l borders,fid -u av -p mo -i esri_80

Layer Administration Utility
 -----------------------------------------------------
 Table Owner : AV
 Table Name : BORDERS
 Spatial Column : FID
 Layer id : 1
 Entities : a
 Layer Type : SDE
```

```
    I/O Mode : NORMAL
    User Privileges : SELECT, UPDATE
    Layer Configuration : DEFAULTS
    -----------------------------------------------------
```

The describe_long operation returns the same information as describe plus such information as feature class description, database, SRID, minimum shape id, false x and false y offset, system units, z offset and units, measure offset and units, spatial index information, layer envelope, creation date, autolocking state, precision, and coordinate system. An example of describe_long is below.

```
   c:\ sdelayer -o describe_long -u raster -p raster -i 12000 -s joe -l
   counties,feature

   Layer Administration Utility
    -----------------------------------------------------
    Layer Description : None
    Table Owner : RASTER
    Table Name : COUNTIES
    Spatial Column : FEATURE
    Layer Id : 39
    SRID : 2  Minimum shape Id : 1  Offset :
     falsex: -179.00000
     falsey: 18.00000
    System Units : 10000000.00000
    Z Offset : 0.00000
    Z Units : 1.00000
    Measure Offset : None
    Measure Units : None
    XY Cluster Tolerance : 0.00000
    Spatial Index :
     parameter: SPIDX_GRID,GRID0=1.6,GRID1-9.6,FULL
     exist: Yes
     array form: 1.6,9.6,0
    Layer Envelope :
     minx: -178.21760   miny: 18.91278
     maxx: -66.96927   maxy: 71.40624
    Entities : a+
    Layer Type : SDE
    I/O Mode : NORMAL
    Autolocking : Enabled
    Precision : BASIC
    User Privileges : SELECT, UPDATE, INSERT, DELETE
    Coordinate system : UNKNOWN
    Layer Configuration : DEFAULTS
    -----------------------------------------------------
```

Obtain information about a feature's geometry

You can get information about a feature's geometry by using the feature_info operation. This is mostly useful when working with an Oracle Spatial database.

The feature_info option of the sdelayer command returns up to 21 different characteristics of a feature, in a series of comma-delimited fields. Fields are returned in the following order:

1. Row Id (Integer)—The row ID of the table containing the spatial column

   If the table does not have a user or row ID column maintained by ArcSDE, the Feature ID (FID) column value is returned here instead. If it has neither row ID nor FID, the value returned is 0.

2. FID (Integer)—The Feature ID of the shape

   If there is no FID, then the row ID is returned instead. If there is neither row ID nor FID, 0 is returned.

3. Entity Type (1 Character)—A single character indicating the entity type, either N (Nil), P (Point), S (Simple), L (Line), or A (Area)
4. Annotation (Boolean)—Indicates whether or not the shape has annotation

   Values are either 'T' (true) or 'F' (false).

5. CAD Data (Boolean)—Indicates whether or not the shape contains CAD Data

   Returned values are either 'T' (true) or 'F' (false).

6. Number of Points (Integer)—The total number of points in the shape
7. Number of Parts (Integer)—The number of parts in the shape

   If an error is encountered when attempting to obtain the parts, that error code, from the sdeerno.h file, is supplied instead.

8. Number of Subparts (Integer)—The number of subparts in the shape

   If this is a type of shape that does not have subparts, the value is 0. If instead an error is encountered when attempting to obtain the subparts, that error code from sdeerno.h is supplied.

9. Self-Touching Rings (Boolean)—Indicates the presence ('T') or absence ('F') of inclusions or cojoined inner rings in the shape

   'T' is always returned for area shapes.

10. Minimum Precision (Char)—The minimum precision to contain this feature; either BASIC (32-bit) or HIGH (64-bit)
11. Verification (Integer)—Indicates whether or not ArcSDE considers a shape valid

    A value for this field is only returned if "all" is used with the -r argument. Possible return values are 0 if the shape is verified as correct or a negative error code from the sdeerno.h file if it is incorrect. You would use the -r argument with features stored in Oracle Spatial to verify the validity of the shapes.

12. Area (Floating Point)—The area of the shape, or 0.0 if this shape is not a polygon

13. Length (Floating Point)—The length or perimeter of the shape, or 0.0 if this shape is a point or multipoint
14. Minimum X (Floating Point)—The minimum x-coordinate of this shape
15. Maximum X (Floating Point)—The maximum x-coordinate of this shape
16. Minimum Y (Floating Point)—The minimum y-coordinate of this shape
17. Maximum Y (Floating Point)—The maximum y-coordinate of this shape
18. Minimum Z (Floating Point)—The minimum z-coordinate of this shape

    This field is only present if this layer has z-coordinates.

19. Maximum Z (Floating Point)—The maximum z-coordinate of this shape

    This field is only present if this feature class has z-coordinates.

20. Minimum Measure (Floating Point)—The minimum measure of this shape

    This field is only present if this feature class has measures.

21. Maximum Measure (Floating Point)—The maximum measure of this shape. This field is only present if this feature class has measures.

Note: Values for 12 through 21 are only returned if the –c argument is specified.

The following example gets the feature geometry information for a feature class called lakes on a server called myrtle. The -r option is specified with "all", so all shapes will be read for validity. (The list returned in the example below is only a partial list of the results.)

```
c:\> sdelayer -o feature_info -l world.lakes,feature -r all -i
sde_world -s myrtle -u him -p shade

ArcSDE 9.2 Oracle 9i  Fri Mar  3 18:54:58 PST 2006

Layer            Administration Utility
-------------------------------------------------------------------
---------
Row Id, FID, Entity Type, Annotation, Cad Data, Number of Points,
Number of Parts, Number of Subparts, Self-Touching Rings, Minimum
Precision, Verification

0,1,A,F,F,38,1,1,F,BASIC,0
0,2,A,F,F,29,1,1,F,BASIC,0
```

Grant and revoke access to a feature class

The grant and revoke operations control access to feature classes. The grant operation allows the owner of a feature class to provide either SELECT, INSERT, UPDATE, or DELETE privileges to other users or roles. The revoke operation allows the owner to rescind previously granted privileges.

The following two commands grant and then revoke select privileges from user bob:

```
c:\> sdelayer -o grant -l victoria,parcels -U bob -A SELECT -u av -p
mo -i esri_80

c:\> sdelayer -o revoke -l victoria,parcels -U bob -A SELECT -u av -
p mo -i esri_80
```

Granting privileges to roles is easier to maintain than repetitively granting the privileges to each user. Whenever possible, create roles representing privileges that can be granted to a group of users.

Get a list of a feature's fields

Use the list operation to get a list of the spatial fields of a feature including the point values for all points that define the shape of the feature.

Specifying a feature ID with the list operation returns the feature's fields and coordinates. Some features may have so many points that you can't view all the information as it's displayed. To control the display of the feature information, list the detailed feature information with a pagination program or redirect it to a file for later viewing or printing.

The following example displays a feature's detail and pipes the output through the more filter, allowing you to page through the information (Only a partial list of points is shown.):

```
c:\> sdelayer -o list -l states,shape -v 11 -i esri_40 -s K9 -D usa
-u me -p mo | more

ArcSDE 9.2 for SQL Server Build 723 Wed Aug 3 16:00:40 2005
Layer            Administration Utility
-----------------------------------------------
Map Layer                 41
Feature Entity Type       Area
Number of Points          179
Feature Envelope Min X   -124.55840
                 Min Y     41.98779
                 Max X   -116.46944
                 Max Y     46.23626
Polygon Perimeter          24.78668
Polygon Area               28.18703
------------------------------------------------
Point      X            Y
------------------------------------------------
1     -121.44041     41.99446
2     -120.87083     41.98779
3     -119.99240     41.98931
4     -119.35065     41.98896
5     -119.30991     41.98924
6     -118.18433     41.99674
7     -117.01791     41.99490  ...
```

Switch between load only and normal I/O mode

To modify a feature class's input/output mode, use the load_only_io and normal_io operations.

Use load-only mode when performing large inserts to avoid the continuous update of the feature class's indexes.

If the grid fields are updated while the feature class is in load-only I/O mode, the spatial index is rebuilt with the new grid sizes when you reset the feature class to normal I/O mode.

You cannot place a feature class registered as multiversioned in load only I/O mode.

You can change the grid sizes while the feature class is in normal or load-only I/O mode.

If the feature class is in normal I/O mode, the indexes on the spatial index table are dropped. While rebuilding the spatial index table, the feature class is set to load-only mode. The indexes are rebuilt when the feature class is reset to normal I/O mode.

There is an envelope (-E) option you can specify when using the load_only and normal_io_mode operations. This option has three possible keywords: empty, calc, and xmin,ymin,xmax,ymax.

Setting it to empty will force a client application to calculate the feature class's envelope on the fly. This can be time-consuming.

When the feature class is in normal I/O mode, the envelope is automatically updated whenever a feature that extends the current envelope is added. The envelope is not updated while the feature class is in load-only I/O mode but is recalculated to the full extent when the feature class is reset to normal I/O mode.

Use xmin,ymin,xmax,ymax to set the envelope explicitly (for example, a study area). However, since the envelope is dynamically maintained, it will be extended to include new features that are added outside the extent set by xmin,ymin,xmax,ymax.

The sdelayer command always creates a feature class in normal I/O mode. The load-only I/O mode is provided to make the bulk data loading processes more efficient.

These examples show the parcels feature class being moved into load only mode then back to normal I/O mode.

```
$ sdelayer -o load_only_io -l victoria,parcels -u av -p mo -i
esri_40

$ sdelayer -o normal_io -l victoria,parcels -u av -p mo -i esri_40
```

When the feature class is returned to normal I/O mode, the spatial index table and database indexes are rebuilt. If the operation does not complete successfully for any reason, the feature class is left in load-only mode.

When a feature class is in load-only I/O mode, the unique index is removed from the feature class's spatial column. When the index is absent, it is possible to enter nonunique values into the spatial column with an application not created with the ArcSDE C- or Java API. No applications besides ArcSDE should ever update the spatial column. Database administrators should be aware of the increased vulnerability of the spatial column when the feature class is in load-only I/O mode.

Register a table with a spatial column

The register operation allows you to create a feature class from a DBMS table that contains a spatial column defined as a user-defined data type (UDT). To date, five different implementations of DBMS spatial data types are supported in ArcSDE geodatabases: Oracle Spatial Geometry Type, Informix Spatial DataBlade, DB2 Spatial Extender, and Spatial Type for Oracle. Tables created with one of these spatial data types and populated using the DBMS SQL interface or some other third-party interface can be added to the ArcSDE geodatabase by registering the existing tables as feature classes. In the following example, the victoria table is registered with spatial column feature (-l), registered user-maintained row ID column of parcel_no (-C), and allowed entity types of area and measure (-e).

```
$ sdelayer -o register -l victoria,feature -e aM -C parcel_no,USER -
u av -p mo -i esri
```

There are certain requirements, specific to each DBMS, that must be met to register a table as a layer with ArcSDE. These include the following:

- Only the owner of the table can register it with ArcSDE. Therefore, the user and password you provide with the -u and -p options must be those for the table owner.
- It must have only one spatial column.
- It must have no other columns of a user-defined type.

Display feature class statistics

You can use the stats operation to display statistics about a feature class including feature entity type counts, total number of features, feature ID, the date of the last feature modification, the number of points in the largest feature, minimum and maximum linear feature lengths, minimum and maximum polygon areas, and the feature class envelope. The following command will display statistics for the parcels feature class:

```
$ sdelayer -o stats -l victoria,parcels -u av -p mo -i esri_40
```

The sdelayer command's spatial index statistics operation, si_stats, can help you determine optimum spatial index grid sizes in an Oracle or SQL Server database. Optimum grid cell sizes depend on the spatial size of all features, the variation in spatial feature size, and the types of searches to be performed on the feature class.

Below is a sample output generated by si_stats:

```
$ sdelayer -o si_stats -l victoria,parcels -u av -p mo -i esri_40

Layer Administration Utility
--------------------------------------------------------
Layer 1 Spatial Index Statistics:
Level 1, Grid Size 200 (Meters)
|------------------------------------------------------------------|
| Grid Records: 978341 |
| Feature Records: 627392 |
| Grids/Feature Ratio: 1.56 |
| Avg. Features per Grid: 18.26 |
| Max. Features per Grid: 166 |
| % of Features Wholly Inside 1 Grid: 59.71 |
|------------------------------------------------------------------|
|                  Spatial Index Record Count By Group             |
|Grids:    <=4      >4     >10    >25    >50   >100   >250   >500  |
|--------- ------ ------ ------ ------ ------ ------ ------ ------ |
|Features: 627392    0      0      0      0      0      0      0   |
|% Total:    100%    0%     0%     0%     0%     0%     0%     0%  |
|------------------------------------------------------------------|
| |
Level 2, Grid Size 1600 (Meters)
|------------------------------------------------------------------|
| Grid Records: 70532 |
| Shape Records: 36434 |
| Grids/Shape Ratio: 1.94 |
| Avg. Shapes per Grid: 18.21 |
| Max. Shapes per Grid: 82 |
| % of Shapes Wholly Inside 1 Grid: 45.35 |
|------------------------------------------------------------------|
|                   Spatial Index Record Count By Group            |
|Grids:     <=4     >4    >10    >25    >50    >100   >250   >500  |
|--------- ------ ------ ------ ------ ------ ------ ------ ------ |
|Features: 35682    752    87     17      3      0      0      0   |
|% Total:    97%      2%    0%     0%     0%     0%     0%     0%  |
|------------------------------------------------------------------|
```

As the output shows, for each defined spatial index level in the feature class definition, the following values and statistics are printed:

- Grid level and cell size
- Total spatial index records for the current grid level
- Total features (shapes) stored for the current grid level
- Ratio of spatial index records per feature
- Feature counts and percentages by group that indicate how features are grouped within the spatial index at this grid level. The column headings have the following meaning (where N is the number of grid cells):
    - <=N, the number of features and percentage of total features that fall within <= N grid cells
    - N, the number of features and percentage of total features that fall within N grid cells

  Notice that the "" groupings include count values from the next group. For instance, the "4" group count represents the number of features that require more than four grid records as well as more than 10, and so on.

- Average number of features per grid
- Maximum number of features indexed into a single grid
- Percentage of features wholly contained by one grid record

The output sample shows spatial index statistics for a feature class that uses two grid levels: one that specifies a grid size of 200 meters the other a grid size of 1,600 meters. When a feature requires more than four spatial index records, it is automatically promoted to the next grid level if one is defined. That is, in no case will a feature index generate more than four grid records if more than one grid level exists. If a higher grid level doesn't exist, a feature can have more than four grid records.

In the example above, 627,392 features are indexed through grid level 1. Because the system automatically promotes features that need more than four spatial index records to the next defined grid level, all 627,392 features for grid level 1 are indexed with four grid records or less. Grid level 2 is the last defined grid level, so features indexed at this level are allowed to be indexed with more than four grid records.

At grid level 2, there are a total of 36,434 features and 70,532 spatial index records: 35,682 features are indexed with four grid records or fewer, 752 features are indexed with more than four grid records, 87 features are indexed with more than 10, 17 features with more than 25, and three features with more than 50 grid records. Percentage values below each column show how the features are dispersed through the eight groups.

Note: To get spatial index statistic information in a DB2 database, use the gseidx command provided by DB2. Consult your DB2 documentation on its use.

Remove the records from a feature class

The truncate operation removes the records from the feature class but does not drop the tables. However, the records of the business table are not removed.

```
$ sdelayer -o truncate -l victoria,parcels -u av -p mo -i esri_80 -N
```

ArcSDE Administration Commands

sderaster

The **sderaster** command manages raster layers. For general concepts regarding the storage and manipulation of ArcGIS raster data and specific information regarding the storage of raster data in a geodatabase, refer to the ArcGIS help online at http://support.esri.com/index.cfm?fa=knowledgebase.webHelp.gateway.

Usage syntax

```
sderaster -o add -l <table,column> [<-M minimum_id>]
[-G {<projection id> | file=<projection file>}]
[-type={default | blob | georaster}]
[-k <config_keyword>] [-S <description_str>]
[-i <service> | <port#>] [-s <server_name>]
[-D <database>] [-u <DB_user_name>]
[-p <DB_user_password>]


sderaster -o alter -l <table,column> [-M <minimum_id>]
[-G {<projection id> | file=<projection file>}]
[-S <description_str>] [-i <service> | <port#>]
[-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>]


sderaster -o colormap -l <table,column> -v <raster_id>
{-d | {-f <image_file> | <database raster>}} [-i <service> |
<port#>]
[-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>]


sderaster -o copy -l <table,column> [-g][-N]
[-V <version_name>]
[-n <image_name>] [-M <minimum_id]
[-rasterbufsize=<size>]
[-G {<projection id> | file=<projection file>}]
[-k <config_keyword>] [-type={default | blob | georaster}]
[-P <description_str>]
[-i <service> | <port#>] [-s <server_name>]
[-D <database>] [-u <DB_user_name>]
[-p <DB_user_password>] <database raster>


sderaster -o delete -l <table,column>
{-v <raster_id>|-W <whereclause>}
[-y] [-V <version_name>]
```

```
[-i <service> | <port#>] [-s <server_name>]
[-D <database>] [-u <DB_user_name>]
[-p <DB_user_password>]


sderaster -o describe [-l <table,column> | -t <table>]
[-verbose] [-i <service> | <port#>] [-s <server_name>]
[-D <database>] [-u <DB_user_name>] [-p <DB_user_password>]


sderaster -o drop {-l <table,column> | -t <table> } [-y]
[-i <service> | <port#> ] [-s <server_name> ]
[-D <database> ] [-u <DB_user_name> ]
[-p <DB_user_password>]


sderaster -o export -l <table,column> -v <raster_id>
[-I] [-V <version_name>]
[{-w | -e} <minx,miny,maxx,maxy>]
[-c {lzw | packbits | g3 | g4}]
[-b <band_number>] [-L <pyramid_level>]
[-i <service> | <port#>] [-s <server_name>]
[-D <database>] [-u <DB_user_name>}]
[-p <DB_user_password>] <image file>


sderaster -o import -l <table,column> [-g] [-N]
[-Align] [-log=<file_name>]
[-interleave={separate | contiguous}
[-rasterbufsize=<size>]
[-c {lz77 | jpeg | jp2}]
[-q {<quality> | bitrate = <bit_rate>}]
[-C {rgb | grayscale}]
[{-R <border>:<tolerance> | -a <NoData>}]
[-n <image_name>] [-M <minimum_id]
[-G {<projection id> | file=<projection file>}]
[-O <minx,maxy> | file=<list file>}]
[-type={default | blob | georaster}]
[-k <config_keyword>] [-P <description_str>]
[-L <pyramid_level>] [-I {nearest | bilinear | bicubic}]
[-t <tile_width,tile_height>] [-i <service> | <port#>]
[-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>]
{-f <image_file> | <database raster>}


sderaster -o insert -l <table,column>
[-N] [-Align] [-log=<file name>]
[-interleave={separate | contiguous}]
[-rasterbufsize=<size>]
[-c {lz77 | jpeg | jp2}]
[-q {<quality> | bitrate=<bit_rate>}]
[-V <version_name>] [-C {rgb | grayscale}]
[{-R <border>:<tolerance> | -a <NoData>}]
[-n <image_name>] [-L <pyramid_level>]
[-I {nearest | bilinear | bicubic}]
```

```
[-t <tile_width, tile_height>]
[-G {<projection_ID> | file=<proj_file>}]
[-O <minx,maxy> | file=<list file>]
[-i <service> | <port#>]
[-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>]
{-f <image_file> | <ArcSDE raster>}


sderaster -o list -l <table,column> [-verbose] [-storage]
[-v <raster_id> [-L <pyramid_level>] | -W <where>}]
[-V <version_name>] [-i <service> | <port#>]
[-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>]


sderaster -o mosaic -l <table,column> -v <raster_id>
[-N][-Align] [-norebuild] [-log=<file name>]
[-rasterbufsize=<size>]
[-V <version_name>] [-C {rgb | grayscale}]
[-q {<quality> | bitrate = <bit_rate>}] [-T]
[{-R <border>:<tolerance> | -a <NoData>}]
[-m {merge | delete}] [-n <image_name>]
[-L <pyramid_level>] [-I {nearest | bilinear | bicubic}]
[-i <service> | <port#>] [-s <server_name>]
[-D <database>] [-u <DB_user_name>]
[-p <DB_user_password>]
{-f <image_file> | <database raster>}


sderaster -o pyramid -l <table,column> -v <raster_id>
[-log=<file name>] [-V <version_name>]
[-L <pyramid_level>,[skipLevel1]] [-I {nearest | bilinear |
bicubic}]
[-q {<quality> | bitrate=<bit_rate>}] [-i <service> | <port#>]
[-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>]


sderaster -o stats -l <table,column> -v <raster_id> [-d]
[-L <pyramid_level>] [-i <service> | <port#>]
[-s <server_name>] [-D <database>]
[-u <DB_user_name>] [-p <DB_user_password>]


sderaster -o truncate -l <table,column> [-y]
[-i <service> | <port#>] [-s <server_name> ]
[-D <database>] [-u <DB_user_name>]
[-p <DB_user_password>]


sderaster -o update -l <table,column> -v <raster_id>
[-N] [-Align] [-log=<file name>]
[-interleave={separate | contiguous}
[-rasterbufsize=<size>]
[-V <version_name>] [-c {lz77 | jpeg | jp2}]
```

```
[-q {<quality> | bitrate = <bit_rate>}]
[-C {rgb | grayscale}] [{-R <border>:<tolerance> | -a <NoData>}]
[-n <image_name>] [-L <pyramid_level>]
[-I {nearest | bilinear | bicubic}]
[-t <tile_width,tile_height>] [-G {<projection_ID> |
file=<proj_file>}]
[-O <minx,maxy>]
[-i <service> | <port#>] [-s <server_name>]
[-D <database>] [-u <DB_user_name>]
[-p <DB_user_password>] {-f <image file> | <database raster>}


sderaster -h
```

Operations

Operation	Description
add	Creates a raster layer by adding a raster column to an empty business table
alter	Alters either the coordinate reference, minimum ID, or raster column description Use the alter operation to change the coordinate system, minimum ID value, or description string. The coordinate system can be changed if the image was created with one. If the image was created without a coordinate system, you will receive the error "Error: No coordref defined on the original raster layer." Images created without coordinate reference systems do not have an extent, and, for that reason, a coordinate system cannot be applied after the fact. Usually, coordinate reference systems are changed for images that have a coordinate reference system originally set to UNKNOWN. Setting the coordinate

	reference system to UNKNOWN requires you to enter the extent of the image, which you can collectively obtain from the source images. Later, you can alter the coordinate reference.
colormap	Updates a raster's color map
copy	Duplicates a raster, including pyramid, image statistics, and color map data The copy operation provides a complete duplicate of another geodatabase raster dataset or raster catalog. The image pyramid, the image statistics, and color map are copied with the rasters. The copy operation is unlike the import operation since the import operation only duplicates the pixel data without the pyramid. You can limit the rasters that are copied by applying a WHERE clause.
delete	Deletes a raster from a raster layer The delete operation allows you to remove a raster from a raster layer.
describe	Describes one or all raster layers owned by a user To obtain the properties of the raster layer, use the describe operation. With the describe operation, if you do not specify a raster layer, the properties of all raster columns are returned. Note, the raster layer ID property listed by the describe operation is not the argument to be entered with the operations that require you to enter a raster ID following a -v option. The raster layer ID identifies the raster layer. The

	raster ID identifies an individual raster within the column. The raster ID of a raster can be obtained from the list operation by listing all the rasters of a raster layer.
drop	Drops a raster column or business table Use the drop operation to drop a raster layer. Dropping a raster layer drops the supporting raster tables that store the raster data and deletes the raster layer's reference from the sde.raster_columns table.
export	Exports raster from a layer A raster can be exported to a file with the export operation. If the raster is compressed, it will expand when it is exported to a TIFF or BSQ file and you may inadvertently exhaust the disk space to which the file is written. The sderaster command will return the following error if it runs out of disk space: "TIFFAppendToStrip: <file name>.tif: Write error at scanline <n>." To avoid running out of disk space, use sderaster -o list to determine if the raster has been compressed and, as a general rule of thumb, multiply the stored size by 2 for LZ77 and 5 for JPEG and JPEG 2000. To obtain the stored size of a raster, use the -o list operation and include the -storage option. Depending on the size of the rasters, this option may require a significant amount of time to complete. You can also apply a compression if the output format is TIFF. By specifying the -c option, you can compress the TIFF file with either LZW, packbits, g3,

	or g4 compression. The maximum output size of a BSQ image file is limited by the current operating system's capability. For AIX, BSQ image files must be smaller than 2 GB. The maximum size of a TIFF file is limited to 4 GB.
import	Imports a raster layer Use the import operation to convert an image file to a raster layer. Be sure to include the -c option to compress the data and the -g option if you want to access the raster layer as a raster dataset from ArcCatalog or ArcMap. Do not use the -g option if you intend to add rows to the raster layer and treat the raster layer as an ArcGIS raster catalog. The import operation supports the standard baseline TIFF and band sequential (BSQ) image file formats. The GeoTIFF format is not supported. To be georeferenced, the image files must be accompanied by a world file. Files that do not include a world file will be registered at the 0,0 coordinate and mosaics will not be allowed following the import operation. The sderaster command also does not support YCbCr TIFF images. ArcCatalog does support these formats, however. Use the -O option to specify a virtual origin if you intend to follow the import operation with the mosaic operations to create a large raster. Use the -Align option if the virtual origin is not cell aligned with the image origin as specified by the coordinates with the image's world file. The -Align option will shift image origin

	within half a pixel cell to agree with the virtual origin you specify. Cell alignment must agree within 14 decimal degrees of freedom to be considered cell aligned. The following BSQ header file standard tags are recognized by sderaster: BYTEORDER NROWS NCOLS NBANDS NBITS BANDROWBYTES Extended tags: COLORMAP [value] Where [value] is the number of entries in the color map. PIXELTYPE [type] Where [type] is one of: 1bit 4bit uint8 int8 uint16 int16 uint32 int32 float double MASK [value] Where [value] indicates whether or not a bitmask is present (1 if it is present, 0 if it is not). You can manipulate the bitmask of a raster by creating a bitmask file to accompany the image file during an import operation. See the 'Discussion' section below for details.

insert	Inserts a raster into a raster layer The insert operation allows you to insert images into a raster layer. Include the -c option to compress the data upon entry. See the discussion on the import operation for supported image formats. When executing the insert operation, sderaster checks the source and destination coordinate references (including unknown and NULL) and returns an error if the source and destination coordinate references do not match. You can manipulate the bitmask of a raster by creating a bitmask file to accompany the image file during an insert operation. See the 'Discussion' section below for details.
list	Lists one or all rasters in a raster layer The list operation returns information about the individual rows of a raster column.
mosaic	Performs piece-wise updates on a raster The mosaic operation allows you to integrate data of an image with the data of an existing raster. The mosaic operation requires that the raster column have a coordinate system, the images have the same cell size and alignment, and the raster not have a color map. To mosaic images with color maps, first create a raster without a color map, mosaic the images, and reapply the color map with the colormap operation. The colormap operation can also be used to remove a color map by specifying the delete option. If color maps are going to be

	reapplied following the mosaic operations, the nearest neighbor pyramid interpolation method should be used to construct a pyramid, since it will not generate new values but use existing neighboring values. If the color maps of the source images are different, third-party tools can be used to create a color map for the entire mosaicked image. An alternative to preserving the color map is to convert the 8-bit, color-mapped images into a 24-bit Trucolor image using the -C RGB option. Although this option triples the storage space requirement for the raster, it alleviates the need for you to manage the color map. In addition, more complex pyramid interpolation resampling methods (such as bilinear and bicubic) may be applied, resulting in smoother-looking pyramid levels. Combining color maps for large areas of imagery may not be possible given the fact that you are restricted to 256 color combinations in an 8-bit, single-band image. You may require the 16,777,216 (256*256*256) color combinations available under a 24-bit Trucolor image. It is recommended that the mosaic operation be applied serially from a single session rather than in parallel from multiple sessions. Parallel applications of the mosaic operation can result in a deadlock. The mosaic operation will automatically update the pyramid. However, it should be pointed out that the pyramid will be completely rebuilt if an image is mosaicked to the left or above the raster's current image origin. A pyramid rebuild can be avoided by presetting a virtual image origin to the upper-left most, coordinate

	when the raster is created. The import and insert operations both include the -O option allowing you to do just that. The -Align option can be supplied with the mosaic operation to automatically cell align the input image with the existing rasters' cell alignment. If you create a raster with a virtual image origin, all subsequent images must be aligned with the virtual origin. The -norebuild option can also be included with the mosaic operation. This option specifies that the pyramid is not to be rebuilt if you attempt to mosaic an image with an origin that is to the left or above the current raster image origin. In this case, an error will be reported rather than allow the pyramid to be automatically rebuilt. When executing the mosaic operation, sderaster checks the source and destination coordinate references (including unknown and NULL) and returns an error if the source and destination coordinate references do not match. You can manipulate the bitmask of a raster by creating a bitmask file to accompany the image file during the mosaic operation. See the 'Discussion' section below for details.
pyramid	Updates a raster's image pyramid Use the pyramid operation to create or recreate the raster pyramid. If you do not specify a pyramid level, the 0 level is assumed and the pyramid is deleted. Specifying a pyramid level of -1 allows sderaster to create all the levels required to reach the apex. You can skip

	building the first level pyramid by specifying skipLevel1.
stats	Calculates a raster's image statistics & histogram The stats operation builds statistics for the specified raster. Certain ArcGIS Desktop tasks, such as applying a contrast stretch or classifying your data, require statistics. ArcGIS Desktop will calculate statistics when it requires them and detects when they are not present. As an alternative to having ArcGIS Desktop calculate image statistics, you can use the sderaster stats operation to perform this task ahead of time. For some raster data, image statistics do improve ArcGIS Desktop applications' default display quality. The data capture of some image data uses only a portion of the bit depth. If statistics are present, ArcGIS Desktop stretches the image data by two deviations, thus artificially extending the bit depth of the image and rendering it with a greater brightness and contrast. For more information on the ArcGIS Desktop default display of rasters, see 'Initial display properties'. For more information on improving raster display speed, see 'Improving raster display speed'. Both of these topics can be found in the raster section of ArcGIS Desktop Help.
truncate	Truncates a raster layer Use the truncate operation to delete the rasters of a raster layer but leave the raster layer schema intact. This operation is useful if

	you want to reload the rasters.
update	Updates a raster With the update operation, you can replace one raster of a raster layer with another. This operation is useful for reentering an edited image file into a raster layer. When executing the update operation, sderaster checks the source and destination coordinate references (including unknown and NULL) and returns an error if the source and destination coordinate references do not match. You can manipulate the bitmask of a raster by creating a bitmask file to accompany the image file during an sderaster update operation. See the 'Discussion' section below for details.

Options

Options	**Description**
-a	Sets pixels with specified value as NoData pixels
-Align	Directs insert, update, mosaic, and import operations to automatically align the pixels of the input image file with the origin of the raster
-c	For the import, insert, and update operations, it enables data compression on the raster that is generated, specifying the

	type of compression as either LZ77, JPEG, or JP2. For the export operation, it enables data compression on the TIFF files (not BSQ), specifying the type of compression as either LZW, packbits, g3, or g4. For JPEG and JP2 compressions, an additional -q quality argument may be supplied. The LZ77 compression technique is a lossless compression. The JPEG compression technique—a lossy compression technique—can only be applied to continuous tone images, which includes 8-bit grayscale and 24-bit (8 bit, three band) RGB images. The quality defaults to 75 but can be altered by supplying the -q argument with a quality value that may range between 0 and 100. The JP2 (JPEG 2000-based) compression can be applied to 8-bit, 16-bit, and 24-bit (8 bit, three band) RGB images. The compression ratio can be expressed as a quality ranging in value between 0 and 255. Values greater than 0 compute a lossy compression, whereas a quality value set to 0 computes a lossless compression. Thus, lower values produce smaller but visually poorer output and low-quality images, whereas higher values result in larger, higher-quality images. For JP2, the quality can also be specified as a fixed bitrate by supplying bitrate=<bit_rate> after the -q option; for example: -q bitrate=1. The fixed bitrate method specifies the actual

	compression ratio that will be applied, whereas a quality value applies a relative compression ratio. The default for JP2 compression is 0, lossless compression. LZW is a lossless compression that tends to perform better than packbits. Packbits—A simple byte-oriented run-length compression scheme g3—Group III compression, also known as CCITT T.4 bilevel encoding g4—Group IV compression, also known as CCITT T.6 bilevel encoding
-C	rgb: This option expands a single-band, color-mapped image into a three-band RGB image, preserving the color of the original image. Useful for mosaicking color-mapped images for which the color maps vary for each image grayscale: This option can only be applied to 1-bit imagery, since it is used to convert single-bit, black-and-white imagery to 8-bit, grayscale imagery.
-D	Database or data source name (not supported on Oracle)
-e	Extraction window in world coordinates
-f	The source raster specified as either a BSQ, TIFF image file, or a quoted argument string that specifies a raster dataset as the source

	If a quoted argument string is specified, the first quote must be followed immediately by the raster layer flag -l. No intervening white space characters are allowed between the -l flag and the first quote. At 9.2, the -f option is not required.
-g	Directs the import operation to register the raster layer with the geodatabase
-G	Coordinate system specifier <projection_id>: coordinate system ID (See the pedef.h file for the integer codes.) file=<proj_file_name>: file containing coordinate system description string
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?". For sderaster, you must specify an operation with -h or -?. For example, sderaster -h -o mosaic will return the syntax and options for the mosaic operation.
-H	Use this option to see the usage and options for a specific operation. For example, sderaster -H mosaic returns the syntax and options for the mosaic operation.
-i	ArcSDE service name, port number, or direct connect information
-I	For the export operation, inverts

	1-bit pixel data For the update, mosaic, pyramid, and import operations, specifies the pyramid resampling interpolation method The following are the three supported interpolation methods listed in order of complexity. As the interpolation methods become more complex, the quality of the resulting pyramid level improves at a cost of a longer execution time. Generally, the nearest neighbor interpolation level is used for single-bit pixel depths, while the bilinear interpolation is used for at least 8-bit pixel depths. The bicubic interpolation method is generally reserved for 16-bit depths and higher. nearest: The nearest neighbor method selects the closest pixel. bilinear: The bilinear method interpolates four adjacent pixels. bicubic: The bicubic method interpolates 16 adjacent pixels. For more information on pixel resampling, refer to the tutorial *Using ArcGIS Spatial Analyst*, which you can download from http://support.esri.com.
-interleave	The type of interleave that will be used to store the data If separate is specified, it indicates the data shall be stored in BSQ format. The contiguous setting applies only to three-band, 8-bit data and indicates that this data shall be stored in band interleave pixel (BIP) format. Using BIP format to store three-band, 8-

	bit data results in higher compressed storage when JPEG compression is applied.
-k	Configuration keyword present in DBTUNE table (default: DEFAULTS) The storage parameters specific to the raster column will be found under the specified keyword.
-l	The raster layer's business table and raster column If you are not the owner of the table, you must qualify the table name as owner.table. If you are copying from one server to another, you must specify the information for the source file and the destination using two separate -l arguments; the first -l argument is the destination and the -l argument at the end of the command string is the source.
-L	The pyramid level If set to a number greater than 0, ArcSDE creates the levels specified. If set to -1, ArcSDE calculates the pyramid until the apex is obtained. If set to 0, the pyramid is deleted. If you specify the optional skipLevel1, the pyramid is built without the first level. Not building the first level can save storage space and time when building the pyramid at a marginal cost to display performance.
-log	The output message log file specifying the file that will

	receive the sderaster command output
-m	Specifies the mosaic policy (merge or delete) Merge is the default value for the -m option. Using merge replaces the overlapping values of the target raster with the those of the source raster. The nodata bitmask is not affected. Specifying delete for the -m option only toggles the nodata bitmask between 0 and 1. The following scenarios apply: • If the overlapping cell value of a source raster is set to 0, specifying delete with the -m option will cause the corresponding target raster nodata value to be set to 0, effectively converting the cell to nodata. • If the overlapping cell value of a source raster is set to anything other than 0, -m delete will cause the corresponding target raster nodata value to be set to 1, converting the cell to data. Whatever value the cell had while it was in a nodata state will now be visible. • If a target cell is already set to nodata, a source raster value of 0 will not change its nodata

	state. • If a target cell is not set to nodata, a non-zero source raster value will not change the nodata state.
-M	Minimum feature ID New raster IDs are assigned the larger of either the minimum ID or the maximum assigned ID + one.
-n	Image name
-N	Ignores color map in data source and enters a grayscale image
-norebuild	Specifies that the pyramid is not to be rebuilt on the dataset if you attempt to mosaic an image with an origin that is to the left or above the current raster image origin Used only with the mosaic operation
-O	The virtual origin of the raster in world coordinates If not specified, the minimum X and maximum Y values obtained from the image world file are applied. Setting the virtual origin is useful if more images are going to be mosaicked to the left or above the current image extent. Setting the virtual origin to the left and above all other images that are going to be mosaicked avoids the geodatabase having to move the virtual origin. Each time the origin is moved, any existing pyramid must be reconstructed. The virtual origin

	can be obtained by examining the world files of the images to be mosaicked together.
-p	DBMS user password
-q	Compression quality for JPEG (0–100) and JP2 (JPEG 2000 based 0–255). Lower values create lower-quality rasters requiring less disk space to store. The default quality value is 75, which should be sufficient for most applications. In most cases, the quality value can be lowered to 40 without a noticeable change in the visual display of the image. The compression quality parameter for JPEG 2000-based compression has a theoretical range of between 0 and 255. In practice, the recommended setting is between 50 and 150. You can also set JPEG 2000 by specifying a bitrate. For 8-bit rasters, the bitrate range is between 0 and 8. For 16-bit rasters, the bitrate range is between 0 and 16. For other pixel depths, a bitrate of 1 is commonly used.
-rasterbufsize	Dynamically sets the RASTERBUFSIZE storage parameter during the execution of the sderaster command without changing the storage parameter for the entire server Increasing the RASTERBUFSIZE parameter for a raster loading session can be useful but is wasteful and unnecessary for sessions that query raster data.
-R	Removes areas surrounding the

	raster that contain no data; for example, removes pixels with background color in a rotated image
-s	ArcSDE server host name (default: localhost)
-S	The raster description (quoted string)
-storage	Lists the storage space consumed by the raster **Note:** The size given by the -storage option is the physical storage size of the raster in the database including image pyramids. This is not the same as the uncompressed size returned in ArcCatalog in the Raster Dataset Properties window, which is the uncompressed image size calculated by image dimension and pixel depth. It has nothing to do with actual storage size. **Note2:** Calculating storage on extremely large raster datasets (larger than 500MB) may take several hours to complete.
-t	The raster tile width and height measured in pixels Each tile is stored as a separate raster block.
-T	Enables single-user transaction The internal commits are disabled, requiring the mosaic operation to respect the commit that occurs at the end of the mosaic operation. If the mosaic operation encounters any database management system (DBMS) issues, the entire user transaction is rolled back. The DBMS must have enough undo

	space to hold the changes of the entire transaction when this option is used. Alternatively, when this option is not used, internal commits are enabled, reducing the amount of required undo space. However, a mosaic operation that fails due to a DBMS problem will only roll back to the last commit. This can result in an incomplete raster dataset.
-type	The data type in which the raster data is stored Only ArcSDE for Oracle currently provides multiple raster storage types. For all other implementations of ArcSDE, the default for that DBMS applies. The default type for ArcSDE for Oracle is LONGRAW, while Binary Large Object (BLOB) and sdo_georaster are the other two possibilites. This setting overrides the value of the DBTUNE RASTER_STORAGE parameter for ArcSDE for Oracle.
-u	DBMS user name
-v	The raster ID
-V	Version name; if specified, uses only the raster data that belongs to the version. (default: sde.DEFAULT) Version names are case sensitive. For example, SDE.DEFAULT and SDE.default are different versions.
-verbose	Describes all properties
-w	Extraction window in pixel coordinates

-W	A valid WHERE clause Because WHERE clauses may contain spaces, the -W option must be last in the command list.

Discussion

The sderaster command is a command line utility for managing raster layers. The supported external raster formats are ESRI BSQ and TIFF.

Supported TIFF types are:

- Bilevel, 1-bit single band
- Grayscale, 4- or 8-bit single band
- RGB, 8-bit unsigned three band
- 16- or 32-bit signed or unsigned data, single band
- 32- or 64-bit floating-point data, single band

The GeoTIFF file format is not directly supported by the sderaster administration command. You should use ArcCatalog to import this format, because it is aware of the embedded project and extent information. If you must use sderaster to import a GeoTIFF image, you will first have to create the world file from which sderaster will read the extent information. World files can be created with the Visual Basic sample. You will also need to interrogate the image file and retrieve its projection information, which must be entered with the -G option.

The typical scenario for using the sderaster command is to create a raster layer (a business table and associated raster tables) with the import operation followed by subsequent executions of the mosaic operation to input additional image files. Finally, the pyramid operation is applied with the level set to -1, instructing sderaster to create a full pyramid.

Be sure to include the -g option during the import operation if you want the raster to appear in the table of contents of either ArcCatalog or ArcMap. Specifying -g creates a raster with an image name of ESRI_SDERASTERDATASET, which is detectable by ArcCatalog and ArcMap. The other operations of the sderaster command are used to make adjustments to the raster layer.

Note: When loading rasters with a color map, be sure to use lossless (LZ77) compression. Using lossy compression would cause the indexing from your raster file to your color map to break.

ArcSDE uses a bitmask to determine if a pixel cell contains data or not. A bitmask value of 1 for a particular pixel cell indicates that it contains data; conversely, a bitmask value of 0 indicates that the pixel cell contains no data.

It is possible to manipulate the bitmask of an ArcSDE raster by creating a bitmask file to accompany the image file during an sderaster input operation of insert, mosaic, update, or import.

The bitmask file must be an 8-bit, singleband BSQ representation of your source image file. Therefore, if your source image file is a 3-band, 32-bit floating point, you will need to copy that file to a downsampled 8-bit, single band BSQ file.

The BSQ bit mask file must then be edited so that any pixel value that is 0 will indicate no data when it is read by the sderaster command. All other non-zero values will indicate data is present.

Edit the header file of the source image file. Add the keyword MASK and set it to 1. This will direct sderaster to look for and load the bitmask file. The header file will have a .hdr suffix. For example:

```
    MASK     1
```

Rename the mask file to have the same root name as the image file but with the suffix .msk. For instance, if your image file is myimage.tif, rename the mask file to be myimage.msk.

Use sderaster with either the import, insert, update, or mosaic operations to bring the image file with its mask file into your geodatabase.

JPEG 2000 compression

The default JPEG 2000 compression used by ArcSDE is the variable bitrate compression algorithm. The variable bitrate algorithm operates on a scale of 1–255.

The resulting image following the application of the variable bitrate algorithm strives to maintain consistent image quality but at the expense of storage space. For instance, for image areas with high variablity in pixel values, such as an urban area or forest, more space is devoted to encoding the pixel data. Conversely, areas that typically show less variability, such as water bodies, use less space for encoding pixel data. Therefore, when you use the variable bitrate compression algorithm, lower compression ratios are achieved for areas of high pixel value variability and higher compression ratios are achieved for areas of low pixel value variability.

Another option for JPEG 2000 compression is fixed bitrate compression. The fixed bitrate compression algorithm operates on a scale that depends on the bit depth of the raster dataset. For 8-bit raster datasets, the scale is greater than 0 but less than 8 bits. For 16-bit raster datasets, the scale is greater than 0 but less than 16 bits.

To achieve a 20:1 compression ratio on an 8-bit raster dataset, divide 8 by 20 to obtain a fixed bitrate of 0.4.
To achieve a 20:1 compression ratio on a 16-bit raster dataset, divide 16 by 20 to obtain a fixed bitrate of 0.8.

To apply the fixed bitrate JPEG 2000 compression with sderaster instead of variable bitrate compression, use the -q option to specify the fixed bitrate.

For example, to achieve a 20:1 compression ratio on an 8-bit raster, specify 0.4 for the bitrate, as shown below:

```
sderaster -o import -l testrd,raster -c jp2 -q bitrate=0.4
OP2006_12436_NC.tif
```

Note: Applying the -q bitrate=<N> option during the mosaic operation ensures that the pyramids are constructed with the same bitrate.

The fixed bitrate algorithm strives to obtain a consistency in storage space usage at the expense of quality. An area of highly variable pixel values, such as an urban area, is be treated the same as a less variable area, such as a lake, to achieve a uniform compression ratio across the entire image.

When using fixed bitrate compression, using tile sizes greater than the default 128 by 128 pixels provides a higher quality output. A tile size of 256 by 256 pixels should be adequate. If you use 128 by 128, you will probably see artifacts at the tile boundary.

Examples

Add a raster column to a table

In the example below, the raster column is added to the empty table topo. The DBTUNE configuration keyword contains the storage configuration for the tables and indexes ArcSDE will create to support the raster column. For more information about DBTUNE configuration keywords, consult the topic "DBTUNE configuration keywords" in the ArcGIS Desktop help, which is available from the ESRI support site on the Knowledge Base tab.

```
$ sderaster -o add -l topo,image -k topo -S "topographic series 11"
-u gis1

Connecting to server ultra, port 7000, as user gis1

Creating raster layer: topo.image

Complete...
```

Alter the properties of a raster layer

In this example, the coordinate system is changed to North American Datum 1983.

```
$ sderaster -o alter -l topo,image -u gis1 -G 4269


Connecting to server ultra, port 7000, as user gis1

Altered.

Complete...
```

Create a color map on a raster

Update the color map of a raster with the colormap operation.

```
c:\> sderaster -o colormap -l topo,image -v 763 -f topo_049.tif -u
gis1
```

Specifying the -d option deletes the raster's color map, while the -f option updates the color map with the contents of an image file or another raster.

Copy a raster layer

In this example, a copy is made of the raster dataset carto_copy to a new raster dataset called topo. The pyramid, statistics, and color map of topo are copied, but only rasters topo_128 and topo_129 are copied over (as specified in the WHERE clause).

To copy a raster from a multiuser geodatabase on one server to a multiuser geodatabase on another server, you need to specify both the source file information and the destination location. The raster in the following example is being copied from a multiuser geodatabase on the server taciturn and is copied to another multiuser geodatabase on the server ultra.

```
$ sderaster -o copy -l topo,image -c lz77 -g -u gis1 -n carto_copy
"-l carto,raster -u gis2 -p gis -i lands -s taciturn -W name =
'topo_128' or name = 'topo_129'"

Connecting to server ultra, port 7000, as user gis1

Image Dimension..............: 500, 1108, 1
Pixel Type...................: uchar

Connecting to server taciturn, port 9000, as user gis2

Creating user table: topo
Creating raster layer: topo.image
Raster ID : 1

Total Time: 00:00:02

Complete...
```

Delete a raster

In this example, one raster with Raster ID 295 is being removed from the topo layer.

```
$ sderaster -o delete -l topo,image -v 295 -u gis1

Connecting to server ultra, port 7000, as user gis1

Raster ID : 295
Deleted.

Complete...
```

Describe raster columns

In this example, no specific layer is specified, so all raster columns are described.

```
$ sderaster -o describe -u gis1


Connecting to server ultra, port 7000, as user gis1

Raster Layer Description ....: <None>
Table Name ..................: GIS1.TOPO
Raster Column ...............: IMAGE
Raster Layer ID .............: 2
Minimum Raster ID ...........: 1
Creation Date ...............: Wed Jun 13 12:43:35 2001
User Privileges .............: SELECT, INSERT, DELETE, UPDATE
Raster Layer Configuration ..: DEFAULTS
Coordinate System ...........: <None&gbr


Raster Layer Description ....: <None>
Table Name ..................: GIS1.LANDMARKS
Raster Column ...............: IMAGE
Raster Layer ID .............: 3
Minimum Raster ID ...........: 1
Creation Date ...............: Wed Jun 13 14:01:22 2001
User Privileges .............: SELECT, INSERT, DELETE, UPDATE
Raster Layer Configuration ..: DEFAULTS
Coordinate System ...........: <None>

2 raster(s).

Complete...
```

Here, the topo layer and image column are specified.

```
$ sderaster -o describe -l topo,image -u gis1

Connecting to server ultra, port 7000, as user gis1

Raster Layer Description ....: <None>
Table Name ..................: GIS1.TOPO
Raster Column ...............: IMAGE
Raster Layer ID .............: 2
Minimum Raster ID ...........: 1
Creation Date ...............: Tue Jun 12 16:05:47 2001
User Privileges .............: SELECT, INSERT, DELETE, UPDATE
Raster Layer Configuration ..: DEFAULTS
Coordinate System ...........: <None>
```

Drop a raster layer

In this example, the topo raster layer is dropped.

```
$ sderaster -o drop -l topo,image -u gis1

Connecting to server ultra, port 7000, as user gis1

Deleting raster layer topo.image...
```

```
   Complete...
```

Export a raster

Below, raster (ID 821) is being exported from the topo layer to file topo_938.tif.

```
   $ sderaster -o export -l topo,image -v 821 -f topo_938.tif -u gis1

   Connecting to server ultra, port 7000, as user gis1

   Exporting...
   Raster ID : 821

   Total Time: 00:00:00

   Complete...
```

Import an image

In this example, the image topo_562.tif is being converted to a raster layer named topo, which will be accessed from ArcGIS Desktop applications as a dataset.

```
   $ sderaster -o import -l topo,image -c lz77 -g -v 232 -f
   topo_562.tif -u gis1

   Connecting to server ultra, port 7000, as user gis1

   Creating user table: topo
   Creating raster layer: topo.image

   Image Dimension..............: 500, 1108, 1
   Pixel Type...................: uchar
   Raster ID : 1

   Total Time: 00:00:02

   Complete...
```

Rasters from other raster layers can be used as the source of the import by specifying the raster in a quoted string after the -f option. For instance, rather than specifying the image file topo_137.tif in the above example, a raster from another geodatabase is used. Note that if you use this method, the -l argument specifying the source ArcSDE raster must immediately follow the double quote. Any other argument or white space will result in an invalid raster error.

```
   $ sderaster -o import -l topo,image -c lz77 -g -v 562 -f "-l
   carto,raster -v 102 -i lands -s taciturn -u gis2 -p gis" -u gis1

   Connecting to server ultra, port 7000, as user gis1

   Creating user table: topo
   Creating raster layer: topo.image

   Image Dimension..............: 500, 1108, 1
```

```
Pixel Type...................: uchar
Raster ID : 1

Total Time: 00:00:02

Complete...
```

The import operation can import multiple files into the same raster column. This is done by listing the files at the end of the command and not specifying the -f option.

```
$ sderaster -o import -l europe,raster -c jpeg -Q 50 -u gis1
france.bsq spain.bsq

france.bsq:
Opening file france.bsq...
Connecting to server jolex, port 9000, as user mark

Creating user table: europe
Creating raster layer: europe.raster

Image Dimension..............: 238, 134, 3
Pixel Type...................: uchar
Raster ID : 1
Time : 00:00:01

spain.bsq:
Opening file spain.bsq...
Image Dimension..............: 210, 127, 3
Pixel Type...................: uchar
Raster ID : 2
Time : 00:00:00

Total Time: 00:00:02
2 raster(s).

Complete...
```

Insert images

The insert operation allows you to insert images into a raster layer. Here, one image, topo_0192.tif, is being inserted into the topo layer. The -c option is specifying that LZ77 compression for the data be done as it is being inserted into the table.

```
$ sderaster -o insert -l topo,image -f topo_0192.tif -c lz77 -u gis1

Connecting to server ultra, port 7000, as user gis1

Image Dimension..............: 500, 1108, 1
Pixel Type...................: uchar
Raster ID : 104

Total Time: 00:00:02

Complete...
```

The insert operation can insert multiple files into the same raster column. This is done by listing the files at the end of the command and not specifying the -f option.

```
$ sderaster -o insert -l topo,image -c lz77 -u gis1 topo_0192.tif
topo_0193.tif topo_0194.tif

topo_0192.tif:
Opening file topo_0912.tif...
Connecting to server ultra, port 7000, as user gis1

Image Dimension.............: 500, 1108, 1
Pixel Type..................: uchar
Raster ID : 104
Time: 00:00:02

topo_0193.tif:
Opening file topo_0913.tif...
Image Dimension.............: 510, 1000, 1
Pixel Type..................: uchar
Raster ID : 105
Time: 00:00:02

topo_0194.tif:
Opening file topo_0914.tif...
Image Dimension.............: 520, 950, 1
Pixel Type..................: uchar
Raster ID : 106
Time: 00:00:02

Total Time: 00:00:07

Complete...
```

You could also insert multiple files into the same raster column using a wildcard. In the example below, all the .tif files in the d:\rasters and f:\bsq directories are being inserted to the orthos file.

```
$ sderaster -o insert -l orthos,image -c jp2 -u raster -p raster -i
12000 d:\rasters\*.tif f:\bsq\*.tif
```

Rasters from other raster layers can be used as the source of the insert by specifying the raster in a quoted string after the -f option. For instance, rather than specifying the image file topo_0192.tif in the above example, a raster from another geodatabase is used.

```
$ sderaster -o insert -l topo,image -f "-l carto,rasters -v 102 -u
gis2 -p gis -i esri_sde -s taciturn" -c lz77 -u gis1

Connecting to server ultra, port 7000, as user gis1

Image Dimension.............: 500, 1108, 1
Pixel Type..................: uchar
Raster ID : 104

Total Time: 00:00:02
```

```
   Complete...
```

List information

The list operation returns information about each row in a raster column. This example shows only two rows; your list may be much longer.

```
   $ sderaster -o list -l topo,image -u gis1

   Connecting to server ultra, port 7000, as user gis1

   Raster ID ..................: 204
   Raster Dimension ...........: 500, 1108, 1
   Raster Tile Dimension .......: 128, 128
   Pixel Type .................: uchar
   Compression ................: none
   Image Pyramid ..............: 0, false, none


   Raster ID ..................: 205
   Raster Dimension ...........: 500, 1108, 1
   Raster Tile Dimension .......: 128, 128
   Pixel Type .................: uchar
   Compression ................: none
   Image Pyramid ..............: 0, false, none

   Raster ID ..................: 206
   Raster Dimension ...........: 1001, 2001+24, 1
   Raster Tile Dimension .......: 128, 128
   Pixel Type .................: uchar
   Compression ................: none
   Image Pyramid ..............: 0, false, none

   3 raster(s).

   Complete...
```

The raster ID uniquely identifies the raster within the raster column. The raster dimension lists three values: the pixel width, the pixel height, and the number of bands. Depending on how the data was loaded, the pixel width or pixel height may include a pixel offset from the nearest tile boundary. An offset will be indicated by a plus sign "+" followed by a value. The value indicates the number of pixels the nearest tile boundary is to the left of the image for the x dimension or the number of pixels the nearest tile boundary is above the image for the y dimension. The offset information is only useful for advanced application developers who need to know where the image begins in relation to the underlying tile structure. The raster tile dimension lists the pixel width and pixel height of a tile. The pixel type lists the bit depth of the pixel data. The pixel types are listed as:

- 1bit—Single bit or black and white
- 4bit—4-bit
- uchar—8-bit unsigned character

- char—8-bit signed character
- ushort—16-bit unsigned
- short—16-bit signed
- ulong—32-bit unsigned
- long—32-bit signed
- float—32-bit floating point
- double—64-bit floating point

The compression is the type of compression in which the raster was stored. Possible compression types include the following:

- none—No compression was applied.
- lz77—Lossless LZ77 compression was applied.
- jpeg—Lossy JPEG compression was applied.
- jp2—Lossy JPEG 2000 compression was applied.

You can specify that the values of a single row should be returned by including the -v option.

```
$ sderaster -o list -l topo,image -v 205 -u gis1 -storage

Connecting to server ultra, port 7000, as user gis1

Raster ID ..................: 205
Raster Dimension ...........: 500, 1108, 1
Raster Tile Dimension .......: 128, 128
Pixel Type .................: uchar
Compression ................: none
Image Pyramid ..............: 0, false, none

1 raster(s).

Complete...
```

The -storage option includes additional storage information about each band and pyramid level. The -storage option will examine each pixel block record of a raster and, therefore, requires a significant amount of time to process depending on the size of the raster. For each band of each pyramid level, the minimum and maximum tile sizes, as well as the mean size and standard deviation, are provided. Also, the number of records in the raster blocks table is displayed along with storage space occupied by those block table records.

```
$ sderaster -o list -u gis1 -l province,raster -v 1 -storage

Connecting to server ultra, port 7000, as user gis1

Raster ID ..................: 1
Raster Dimension ...........: 88383, 41428, 1
Raster Tile Dimension .......: 320, 256
```

```
Pixel Type ..................: uchar
Compression .................: lz77
Image Pyramid ...............: 8, false, nearest
Raster Tile Storage .........:
min max mean std dev count total
Level 0:
Band 1 103 4889 350.99 514.74 23355 8,005 KB
Level 1:
Band 1 103 5787 524.76 688.27 6256 3,206 KB
Level 2:
Band 1 103 6201 788.03 872.64 1726 1,328 KB
Level 3:
Band 1 103 6695 1178.07 1014.13 475 546 KB
Level 4:
Band 1 103 4859 1650.67 1077.74 134 216 KB
Level 5:
Band 1 208 4268 2042.66 1089.10 41 82 KB
Level 6:
Band 1 222 5358 2223.50 1544.52 14 30 KB
Level 7:
Band 1 268 5568 1945.00 2167.35 6 11 KB
Level 8:
Band 1 359 3978 2168.50 2559.02 2 4 KB
13,430 KB

13,752,096 bytes

1 raster(s).

Complete...
```

You can get additional information about the image extent of the raster and the cell size in world coordinates by using the -verbose option. The raster statistics are also displayed with the -verbose option listing the minimum and maximum values as well the mean and standard deviation.

```
$ sderaster -o list -u gis1 -l province,raster -verbose -v 1

Connecting to server ultra, port 7000, as user gis1

Raster ID ...................: 1
Raster Dimension ............: 88383, 41428, 1
Raster Tile Dimension .......: 320, 256
Pixel Type ..................: uchar
Compression .................: lz77
Image Pyramid ...............: 8, false, nearest
Extent ......................:
minx : -141.00161818181800
miny : 41.68847272727270
maxx : -52.61961818181820
maxy : 83.11547272727270
Cell Size....................:
x : 0.00100000000000
y : 0.00100000000000
Statistics ..................:
min : 1.0000000000
```

```
max : 13.0000000000
mean : 6.1075254726
std dev : 3.6373814396


1 raster(s).

Complete...
```

Mosaic

This mosaic operation is integrating topo_137.tif with raster (ID 382) in the topo layer. Remember, the mosaic operation requires raster columns to have a coordinate system, both the existing and new images must have the same cell size and alignment, and the raster cannot have a color map. If the raster already has a color map, you will have to remove it before you mosaic, then reapply the color map.

```
$ sderaster -o mosaic -l topo,image -v 382 -f topo_137.tif -u gis1

Connecting to server ultra, port 7000, as user gis1

Image Dimension..............: 500, 1108, 1
Pixel Type...................: uchar
Image Extent.................:
minx : 234917.94185799997649
miny : 4198842.21638599969447
maxx : 239987.78185799997300
maxy : 4210089.33638599980623
Raster ID : 382

Total Time: 00:00:04

Complete...
```

Rasters from other raster layers can be used as the source of the mosaic by specifying the raster in a quoted string with the -l option. For instance, rather than specifying the image file topo_137.tif in the above example, a raster from another geodatabase is used.

```
$ sderaster -o mosiac -l topo,image -v 382 "-l carto,raster -v 102 -
i lands -s taciturn -u gis2 -p gis" -u gis1

Connecting to server ultra, port 7000, as user gis1

Image Dimension..............: 500, 1108, 1
Pixel Type...................: uchar
Image Extent.................:
minx : 234917.94185799997649
miny : 4198842.21638599969447
maxx : 239987.78185799997300
maxy : 4210089.33638599980623
Raster ID : 382

Total Time: 00:00:04

Complete...
```

The mosaic operation can mosaic multiple files into the same raster column. This is done by listing the files at the end of the command and not specifying the -l option.

```
$ sderaster -o mosiac -l topo,image -v 382 -u gis1 france.bsq
spain.bsq

Connecting to server ultra, port 7000, as user gis1

france.bsq
Opening file france.bsq...
Image Dimension..............: 238, 134, 3
Pixel Type...................: uchar
Image Extent.................:
minx : -6.24520815168458
miny : 41.23423565568964
maxx : 10.60812558971746
maxy : 50.69201366247066
Time : 00:00:01

spain.bsq:
Opening file spain.bsq...
Image Dimension..............: 210, 127, 3
Pixel Type...................: uchar
Image Extent.................:
minx : -9.65854156766455
miny : 35.26090217772306
maxx : 5.20368101441573
maxy : 44.22090239467350
Time : 00:00:01

Raster ID : 382

Total Time: 00:00:03

Complete...
```

You could also mosaic multiple files into the same raster column using a wild card. In the following example, all the bsq files in the specified directory are used.

```
c:\> sderaster -o mosaic -l land,image -v 407 -u editor2 -p change
c:\raster_temp\*.bsq
```

Create a pyramid

A pyramid is created on the topo layer with a pyramid level of -1, meaning all the levels required to reach the apex will be created. "skipLevel1" is also specified, though, so the first level of the pyramid will not be built.

```
$ sderaster -o pyramid -l topo,image -L -1,skipLevel1 -v 701 -u gis1

Connecting to server ultra, port 7000, as user gis1

Updating pyramid...
Raster ID : 701

Total Time: 00:00:00
```

```
Complete...
```

Calculate statistics on a raster

Statistics are calculated below on the raster (ID 847).

```
$ sderaster -o stats -l topo,image -v 847 -f topo_291.tif -u gis1

Connecting to server ultra, port 7000, as user gis1

Calculating statistics...
min : 27.0000000000
max : 248.0000000000
mean : 99.4931516245
std dev : 55.8325332694


Raster ID : 847

Total Time: 00:00:01

Complete...
```

The -L option allows you to build image statistics based on a reduced resolution pyramid level. For large rasters, this can provide a fast way to create image statistics for a fractional loss of accuracy. You can direct the sderaster command to build statistics on a particular level by specifying that level with the -L option, or you can enter -1 to specify that the top level of the pyramid should be used to build image statistics.

```
$ sderaster -o stats -l topo,image -v 847 -u gis1 -L -1

Connecting to server ultra, port 7000, as user gis1

Calculating statistics...
min : 27.0000000000
max : 248.0000000000
mean : 99.4931516245
std dev : 55.8325332694


Raster ID : 847

Total Time: 00:00:01

Complete...
```

The -d option allows you to remove the statistics from the raster.

```
$ sderaster -o stats -l topo,image -v 847 -u gis1 -d

Connecting to server ultra, port 7000, as user gis1

Statistics & histogram updated.
```

```
   Complete...
```

Remove rasters from a layer

If you want to remove the rasters from a layer then reload different rasters, you can use the truncate operation. Here, the truncate operation is used to delete the rasters of a raster layer, topo. The layer's schema, though, will be unaffected.

```
   $ sderaster -o truncate -l topo,image -u gis1


   Connecting to server ultra, port 7000, as user test

   Truncating raster layer topo.image...

   Complete...
```

Update a raster

Here, the raster (Raster ID 495) is being replaced with image topo_982.tif.

```
   $ sderaster -o update -l topo,image -v 495 -f topo_982.tif -c jp2 -q
   bitrate=1 -u gis1

   Connecting to server ultra, port 7000, as user gis1

   Image Dimension.............: 500, 1108, 1
   Pixel Type..................: uchar
   Raster ID : 495

   Total Time: 00:00:03

   Complete...
```

Rasters from other raster layers can be used as the source of the update by specifying the raster in a quoted string after the -f option. For instance, rather than specifying the image file topo_982.tif in the above example, a raster from another geodatabase is used.

```
   $ sderaster -o update -l topo,image -v 495 -f "-l carto,raster -v
   102 -i lands -s taciturn -u gis2 -p gis" -c lz77 -u gis1

   Connecting to server ultra, port 7000, as user gis1

   Image Dimension.............: 500, 1108, 1
   Pixel Type..................: uchar
   Raster ID : 495

   Total Time: 00:00:03

   Complete...
```

ArcSDE Administration Commands

sdetable

The **sdetable** command administers business tables and the data stored in them, as well as their indexes and views.

Usage syntax

```
sdetable -o add_uuid_column -t <table> -c <column>
-u <DB_User_name> [-D <database>] [-p <DB_User_password>]
[-i <service>] [-s <server_name>] [-q]


sdetable -o alter_column -t <table> -c <column>
[-x {<ALLOW_NULLS | DISALLOW_NULLS}]
[-z <sde_column_type>] [-S <column_description>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o alter_reg -t <table>
[{-c <row_id_column> -C {SDE|USER} } | {-C NONE} ]
[-L {OFF | ON}] [-M <minimum_id>] [-S <table_description>]
[-V {SINGLE,version_name | MULTI | HYBRID}] [-F]
[-k <config_keyword>] [-H {VISIBLE | HIDDEN}]
[-R {MANY | SINGLE}] [-y {OFF|<history_table,<history_rowid_column>,
<from_date_column>,<to_date_column>}]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o create -t <table> -d <column_definition>
[-k <config_keyword>] [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-q]


sdetable -o create_index -t <table> -n <index>
-c <column> [-k <config_keyword>]
[-Q] [-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-q]


sdetable -o create_mv_view -T <view_name> -t <table_name>
[-i <service>] [-s <server_name>] [-D <database>]
```

```
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o create_view -T <view_name> -t <table1,table2...tablen>
-c <table_col1,table_col2...table_coln>
[-a <view_col1,view_col2...view_coln>] [-w <"where_clause">]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o delete -t <table>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o delete_index -n <index> [-t <table>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o delete_mv_view -t <table_name>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o describe -t <table> [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_User_name> [-p <DB_User_password>] [-q]


sdetable -o describe_long -t <table> [-i <service>]
[-s <server_name>] [-D <database>] -u <DB_User_name>
[-p <DB_User_password>] [-q]


sdetable -o describe_reg [{-U <user> | -t <table>}] [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-q]


sdetable -o {grant | revoke} -t <table> -U <user>
-A <SELECT,UPDATE,INSERT,DELETE>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-I] [-q]


sdetable -o list -t <table> -c <column> -v <column_value>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o {load_only_io | normal_io} -t <table>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]
```

```
sdetable -o populate_uuid_column -t <table> -c <column> -u
<DB_User_name>
[-D <database>] [-p <DB_User_password>]
[-i <service>] [-s <server_name>] [-q]


sdetable -o rebuild_index -t <table>
[-x {RASTER | SPATIAL | VERSIONED | XML | ALL}]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o register -t <table>
[{-c <row_id_column> -C {SDE|USER} } | {-C NONE} ]
[-L {OFF | ON}] [-M <minimum_id>] [-S <table_description>]
[-V {SINGLE | MULTI | HYBRID}] [-k <config_keyword>]
[-H {VISIBLE | HIDDEN}] [-R {MANY | SINGLE}]
[-y {<history_table,<history_rowid_column>,
<from_date_column>,<to_date_column>}]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-q]


sdetable -o rename -t <table> -T <new_name>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o truncate -t <table>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]


sdetable -o unregister -t <table> [-F]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]
```

sdetable -o update_dbms_stats has database specific usage:

Oracle

```
sdetable -o update_dbms_stats -t <table_name> [ -K <keyword>]
[-n <{ALL | <index_name>}>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_User_name> [-p <DB_User_password>] [-N] [-q]
```

SQL Server

```
sdetable -o update_dbms_stats -t <table_name> [ -K <keyword>]
-m <"{with fullscan | with sample %% percent} [index]">
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_User_name> [-p <DB_User_password>] [-N] [-q]
```

Informix

```
sdetable -o update_dbms_stats -t <table_name>
[-K <keyword>] -m <{low | medium | high}>
[-i <service>] [-s <server_name>][-D <database>]
-u <DB_User_name> [-p <DB_User_password>]
[-N] [-q]
```

DB2

```
sdetable -o update_dbms_stats -t <table_name> [-K <keyword>]
[-n {ALL | <index_name>}][-i <service>] [-s <server_name>]
[-D <database>] -u <DB_User_name> [-p <DB_User_password>]
[-N] [-q]


sdetable -h

sdetable -?
```

Operations

Operation	**Description**
add_uuid_column	Adds a universal unique ID column to the table
alter_column	Alters the definition of a column
alter_reg	Modifies a registration entry such as converting the table from single to multiversioned, single to hybrid, changing the row_id to be maintained by ArcSDE rather than by the user, or enabling/disabling archiving **Note:** You cannot alter the registration of the row_id column

	on a view.
create	Creates a new table
create_mv_view	Creates a multiversioned view
create_index	Creates an index on a table
create_view	Creates a DBMS view
delete	Deletes a table; can be used to completely delete a layer (including business table) and a view
delete_mv_view	Deletes a multiversioned view
delete_index	Deletes an index on a table
describe	Displays a table definition
describe_long	Displays the table definition in detail
describe_reg	Displays table registration entries
grant	Grants access to a table for a user
list	Lists column data for a given field value
load_only_io	Puts table in load-only I/O mode, which drops the spatial index and makes the table unavailable to ArcSDE clients Used primarily to improve performance when bulk loading

	data
normal_io	Puts table in normal I/O mode This recreates the indexes and makes the table available to ArcSDE clients.
populate_uuid_column	Populates the universal unique ID column
rebuild_index	Rebuilds the indexes of the specified objects
register	Registers a table in the ArcSDE table registry (TABLE_REGISTRY or SDE_table_registry) Is also used to register a table as versioned and enable it for archiving
rename	Renames a table Do not use sdetable -o rename to rename a business table you created in ArcGIS Desktop.
revoke	Removes access to a table for a user
truncate	Deletes all records from a table
unregister	Removes a table from the ArcSDE table registry To unregister a table, it cannot have been

	registered as versioned, registered with the geodatabase, nor contain a spatial column.
update_dbms_stats	Updates RDBMS table and/or index statistics

Options

Options	**Description**
-a	A comma-separated list of a view's column names Use this option if you want the columns in the view to have different names than they did in the source table. The order in which you list these should correspond to the columns specified with the -c option.
-A	Type of access: SELECT, UPDATE, INSERT, or DELETE
-c	For the alter_reg or the registration operation, specifies the row ID column name This option requires using -C. If using the create_view operation, use a comma-separated list of column names. For the add_uuid_column and the populate_uuid_column operations, specifies the uuid column to be added or populated
-C	Row ID column type: SDE: The Row_ID column is maintained by ArcSDE. Tables that are registered with the geodatabase must be maintained by ArcSDE. Your

	application must not alter the values of the row_ID column. USER: The Row_ID column is user-maintained. Your application will maintain the row_ID. NONE: No row_ID column created. The table_registry does not have a reference to a row_ID column. You can't use the -c option with this choice.
-d	Column definition
-D	Database or data source name (not supported on Oracle)
-F	Forces a multiversioned table with edits to be converted to a nonversioned table, with loss of edits The records stored in the delta tables are lost when they are dropped.
-h	Prints usage and options
-H	Registers the table as either visible or hidden
-i	ArcSDE service name or direct connect information
-I	Inherit privilege to grant The grant option is included with the granted privilege. For example, if user A grants user B update privileges on a table, the -I option indicates that user A also wants to grant user B the ability to grant other users update privileges on a particular table.
-k	DBTUNE configuration keyword (default: DEFAULTS)

-K	Update DBMS statistics keyword: a—UPDATE_ADDS_TABLE_STATS b— UPDATE_BUSINESS_TABLE_STATS d—UPDATE_DELETES_TABLE_STATS f—UPDATE_FEATURE_TABLE_STATS r—UPDATE_RASTER_TABLE_STATS x—UPDATE_XML_TABLE_STATS
-L	Enables (on) or disables (off) a ROW_LOCK on a registered table
-m	Update DBMS statistics mode Specify if running against a SQL Server or Informix database; options are: <u>SQL Server</u> with fullscan with sample <#> percent with sample <#> rows with resample <u>Informix</u> low medium high Consult your DBMS documentation for details on update modes. There is no need to specify -m option if running update_dbms_stats against a DB2 database. Beginning with ArcSDE 9.2, do not use the -m option when executing update_dbms_stats on Oracle databases.
-M	Minimum row ID
-n	Either ALL for all indexes or a particular index name If you specify this option, only indexes are updated.
-N	No verification is performed—the operation begins immediately after

	being invoked.
-o	Operation
-p	DBMS user password
-q	Quiet—All titles and warnings are suppressed.
-Q	Index is unique.
-R	[{SINGLE | MANY}]: Sets the row ID allocation method for a row ID column maintained by ArcSDE Set the method to SINGLE if records are infrequently inserted into the table. Metadata tables are examples of these types of tables. Setting the method to SINGLE causes the session to get a single ID value each time a record is inserted into the table. Setting the method to MANY (the default) causes the session to fetch a block of ID values each time an insert occurs. The next block is not fetched until enough inserts have occurred to exhaust the current block of IDs. Use MANY for regular business tables that have many rows inserted during a session. This option is currently only implemented for Oracle to avoid gaps in the metadata tables' row ID columns maintained by ArcSDE.
-s	ArcSDE server host name (default: localhost)
-S	For the registration or alter_reg operations, specifies the registration description string For the alter_column operation, specifies the column description
-t	Table name

	If you do not own the table, qualify the table name as "owner.table". If you are using the create_view operation, use a comma-separated list of table names.
-T	New table name
-u	DBMS user name
-U	The user to whom table access is granted or revoked or the user who owns a table
-v	Column value
-V	Specifies if the table is registered as single versioned, multiversioned, or hybrid Registering a table as single versioned means you can perform nonversioned edits (save to base transactions); registering as multiversioned lets you do versioned editing and allows you to enable archiving and geodatabase replication. Registering as hybrid allows you to do either versioned or nonversioned edits, but does not allow for archiving or replication. Specifying the version name when reregistering a table as single versioned allows you to specify which version's delta tables will be moved into the table.
-w	Structured Query Language (SQL) WHERE clause
-x	When used with the alter_column operation, indicates whether a column allows or disallows NULL values (ALLOW_NULLS or DISALLOW_NULLS) When used with the rebuild_index operation, specifies the type of index

	to be rebuilt, either RASTER, SPATIAL, XML, VERSIONED or ALL RASTER rebuilds the indexes for the Raster tables. SPATIAL rebuilds the spatial index table. XML rebuilds the XML indexes. VERSIONED rebuilds the indexes for the Adds and Deletes tables. ALL rebuilds all valid indexes on the table; if -x is not specified, ALL is assumed
-y	Enables or disables archiving To enable archiving with the register or alter_reg operations, you must specify the name of the history table, and the names of the history row ID, from date, and to date columns in the history table. Archiving can only be disabled with the alter_reg operation.
-z	Used with the alter_column operation; specifies the data type you want the column to be, either INT16, INT32, FLOAT32, or FLOAT64. Do not alter the data type of an ArcSDE maintained column. Allowed data type conversions are dependent on the DBMS used to store the data. Consult your DBMS documentation for specific information on data type conversions.
-?	Prints usage and options (Use "-\?" on C shell.)

Discussion

To create a business table, supply the table name and field specifications. The field specification uses standard SQL syntax to specify a field list, so you cannot use any SQL-reserved words as field names. The field specification may use any data type defined by ANSI SQL89. The field specification of sdetable accepts standard ArcSDE data types, which it maps to DBMS-native data types. The standard ArcSDE data types are:

smallint(n) where: 0 < n <= 4
int16
int32
int64
float32
float64
integer(n) where: 4 < n <= 10
float(n,m) where: 0 < n <= 6, 0 < m <= DBMS limit
double(n,m) where: 6 < n <= DBMS limit, 0 < m < DBMS limit
string(n) where: 0 < n <= DBMS limit
blob
clob
uuid
nstring
nclob
date

In the case of ArcSDE for Informix, ArcSDE for DB2, and ArcSDE for Oracle using the Spatial Type for Oracle, the spatial data types are also accepted. Creating a business table with a spatial type does not create a feature class. To create a feature class from an empty business table containing a spatial type, use the sdelayer -o add command. If the table contains records, use the sdelayer -o register command to register the spatial table as a feature class.

Spatial data types:

st_point
st_curve
st_linestring
st_polygon
st_multipoint
st_multicurve
st_multilinestring
st_multipolygon
st_multisurface
st_surface
st_geometry
st_geometrycollection

Examples

Add a universal unique ID column

To add a universal unique ID column to a table, use the add_uuid_column operation. In the example below, a universal unique ID column, parceluuid, is being added to the table victoria.

```
   $ sdetable -o add_uuid_column -t victoria -c parceluuid -u av -p mo
```

Alter the definition of a column

Use the alter_column operation to alter a column's definition. You can change the NULL constraint or the data type if the DBMS allows it. This is limited to the following data types:

- INT16
- INT32
- FLOAT32
- FLOAT64

You cannot alter the definition of a column that has been registered as maintained by ArcSDE.

The following example alters a column, nest_size, to allow null values and changes the data type to FLOAT64.

```
$ sdetable -o alter_column -t birds -c nest_size -x ALLOW_NULLS -z
float64 -u av -p mo -N -q
```

Alter the registration of a table

You can alter the registration of a table from and to one of three options for editing: single, multiversioned, or hybrid. If registered as multiversioned, you can enable or disable archiving. In the example below, the registration of the table victoria is being changed to multiversioned (-V) and enabled for archiving (-y).

```
c:\> sdetable -o alter_reg -t victoria -c objectid -C sde -V multi -
y h_victoria,h_rowID,h_from,h_to -u av -p mo -N
```

You can also alter the registration of a table to change how the row_id column is maintained. In the following example, the row_id column (object_id) is altered to be user-maintained.

```
c:\> sdetable -o alter_reg -t birds -c object_id -C user -u av
```

Note: You cannot alter the registration of the row_id column of a view if the row_id column for the view has already been defined. You can use the sdetable -o describe_reg command (see example under 'Display table registration entries' to determine if the row_id column is already defined on the view.

Create a table

You can create a new table in the database using the create operation.

```
$ sdetable -o create -t victoria -d "name string(20), tot_pop
integer(9)" -k vict -u av -p mo -s ultra -i esri_80
```

The table created as a result of executing the above command contains two columns: a string column, name, and an integer column called tot_pop.

Tables are created as single versioned. Use the alter_reg operation to reregister them as multiversioned or hybrid if you want to edit the table with the ArcGIS Desktop applications. A row ID column that will store the unique object ID to the edited rows must be either added to the table or an existing one must be named. The row ID column must be managed by the sde user. The table victoria is reregistered as multiversioned. The column objectid is added as the row ID column and is populated by ArcSDE with unique ID values.

Create a multiversioned view

To access the attribute information of a multiversioned feature class from a client application that does not recognize versioning, you can create a multiversioned view. Multiversioned views should only be used with simple geodatabase objects (those not involved in networks or topologies), can only be applied to one versioned table or feature class, and cannot include the spatial column of the table. You cannot create a multiversioned view on a view. The following example creates a multiversioned view, world_imv_view, on the base table of the feature class world.

```
c:\> sdetable -o create_mv_view -T world_imv_view -t world -u av -p
morti26
```

Unlike a standard ArcSDE view, you do not choose columns or define a WHERE clause. The schema of a multiversioned view is identical to that of the business table on which it is based.

When querying a multiversioned view, you need to specify which version. Using the DBMS SQL interface, set the current version by executing the stored procedure sde.version_util.set_current_version. (This stored procedure is not supported on all DBMSs.)

For example, if the current version of the database you want to access is called working, set the database to that version during the SQL session.

Create an index on a table

To improve query performance, you can create an index on a table. The create_index operation creates an index on a business table column. This example creates an index (index1) on column parcel_no.

```
$ sdetable -o create_index -t victoria -c parcel_no -n index1 -u av
-p morti26 -i esri_40
```

You can create aggregate indexes by specifying a comma-separated list of columns. The next example creates an aggregate index on the parcel_no and zone_no columns.

```
$ sdetable -o create_index -t victoria -c parcel_no,zone_no -n
index2 -u av -p morti26
```

Create a database view

You can define views on a single feature class or table (for instance, to limit a user's access to the columns in a feature class), between two feature classes, between a feature class and a table, or create more complex views containing subqueries. This example creates a view, view_victoria, which includes the parcel_no column from the victoria feature class and the address from the province table, for all records where the province is 10 and the zone_no value in the victoria feature class equals the zone_no value in the province table.

```
$ sdetable -o create_view -T view_victoria -t victoria -c
victoria.parcel_no,province.address -w "province=10 and
victoria.zone_no = province.zone_no" -u av -p morti26
```

You must list the columns you want included in the view. If you are creating a view using multiple tables as shown above, you must also include a WHERE clause.

In addition to creating the view, sdetable –o create_view registers the view with ArcSDE. This means that ArcSDE places an entry for the view in the ArcSDE system table TABLE_REGISTRY (SDE_table_registry in SQL Server databases) and an entry for each of the view's columns in COLUMN_REGISTRY (SDE_column_registry in SQL Server databases). You could also create a view using SQL, then register it with ArcSDE using the sdetable -o register command.

Note: If you are going to use a view in an Informix database with an ArcSDE client, views should only be created with the sdetable -o create command—not using SQL.

You can also create spatial views. That means you include the spatial column (for example, the SHAPE column) in the view. These views can only reference one table that contains a spatial column. If you want to include attribute information from another table containing a spatial column in your spatial view, you must first create a nonspatial view of that second table. You can then create your spatial view using one table with a spatial column and attribute information from the nonspatial view. Spatial views should only be created on nonversioned tables.

Delete a table

To delete a business table or view, use the delete operation. The associated feature class (layer), if one exists, is deleted as well.

```
c:\> sdetable -o delete -t victoria -u av -p morti26 -s ultra -i
esri_40
```

Delete a multiversioned view

Use the delete_mv_view operation to remove a multiversioned view. This example drops the multiversioned view world.

```
$ sdetable -o delete_mv_view -t world -u av -p morti26
```

The view can also be dropped using the ArcSDE C API function SE_table_drop_mv_view.

Delete a table's index

To drop a table's index, use the delete_index operation.

```
$ sdetable -o delete_index -n index1 -u har -p sugar
```

Not all DBMS maintain unique index names, in which case you must qualify the index name with the table name (for example, victoria.index1).

Describe a table

Use the describe operation to display a table's definition, as shown in the following example:

```
$ sdetable -o describe -t victoria -u av -p morti26 -i esri_40
```

To list rows of a business table, you must specify the table, a column name, and a value of the specified column. An implicit query is performed on the table to fetch and display the rows that have a column equal to the value.

Obtain a detailed description of a table

To list the business table definition in column detail, use the describe_long operation.

```
$ sdetable -o describe_long -t victoria -u sasha -p polarbear -i
esri_40
```

The describe_long operation lists the following properties of each column:

Column Owner
Column Table
Column Name
SDE Column Type
Column Size
Decimal Digits
Null Allowed?
Layer ID

Display table registration entries

Use the describe_reg operation to obtain information about a table's registration status. Use -t to return information for one table or -U to return information for all tables owned by a specific user. Otherwise, describe_reg returns all tables. In the example below, the registration entry for one table, landmarks, is returned.

```
c:\> sdetable –o describe_reg –t dbo.landmarks

Attribute        Administration Utility

Table Database          : SDE
Table Owner             : DBO
Table Name              : LANDMARKS
Registration Id         : 47
Row ID Column           : OID
Row ID Column Type      : SDE Maintained
Row ID Allocation       : Many
Row Lock                : Not Enable
Minimum Row ID          : 1
Dependent Objects       : None
Dependent Object Names  : 147
```

```
   Registration Date       : 01/24/06 09:41:00
   Config. Keyword         : DEFAULTS
   User Privileges         : SELECT, UPDATE, INSERT, DELETE
   Visibility              : Visible
```

Grant a user access to a table

Use the grant operation to allow a user specific types of access to a table. (SELECT, INSERT, UPDATE, DELETE). In the following example, user hp is given select permission to the table hydrants owned by RJP.

```
   c:\> sdetable -o grant -t sde.RJP.hydrants -U hp -A
   SELECT,INSERT,UPDATE,DELETE -u RJP

   Enter Database User password:

   ArcSDE 9.2 for SQL Server Build 723 Wed Aug  3 16:02:07  2005
   Attribute      Administration Utility
   ------------------------------------------------------------
   Permissions successfully granted on table sde.RJP.hydrants
```

List column data

You can list column data for a given field value using the list operation.

```
   c:\> sdetable -o list -t sde.cat.ACCIDENTS -c ACCTYPE -v 11 -u cat -
   p zoom

   ArcSDE 9.2 for SQL Server Build 723 Wed Aug  3 16:02:07  2005
   Attribute          Administration Utility
   -------------------------------------------------------------
   OBJECTID: 12
   MILE: 120
   DAY_NIGHT: D
   ALCOHOL: 0
   HITRUN: 0
   ACCTYPE: 11
   WEATHER: RAIN
   MEANS: 7
   SPEED: 38

   OBJECTID: 23
   MILE: 1432
   DAY_NIGHT: N
   ALCOHOL: 1
   HITRUN: 0
   ACCTYPE: 11
   WEATHER: CLEAR
   MEANS: 7
   SPEED: 40
```

Switch a table to load-only I/O mode, then back to normal I/O mode

You can put a table in load-only I/O mode when you want to load large amounts of data at one time. When finished, you should switch it back to normal I/O mode.

```
c:\> sdetable -o normal_io -t sde.RJP.hydrants -u RJP -p windowpane

ArcSDE 9.2 for SQL Server Build 723 Wed Aug  3 16:02:07  2005
Attribute     Administration Utility
----------------------------------------------------------------
Successfully changed mode for sde.RJP.hydrants
```

Populate a universal unique ID column of a table

Use the populate_uuid_column operation to populate the uuid of a table.

```
$ sdetable -o populate_uuid_column -t victoria -c parceluuid -u
rocket -p bubby
```

Rebuild an index

To rebuild the index of a specified object, use the rebuild_index operation. Here, all the indexes on the table landmarks are rebuilt.

```
$ sdetable -o rebuild_index -t landmarks -u squeaky
```

In this example, the indexes on the raster tables associated with the business table vegetation are rebuilt.

```
sdetable -o rebuild_index -t vegetation -x RASTER -u vijay
```

Register a table

Use the register operation to register a table in the ArcSDE table registry. The following table, world, is registered with a row ID column (ID) maintained by ArcSDE.

```
$ sdetable -o register -t world -c ID -C SDE -u dart -p wolfy
```

You will need to register the table with an ArcSDE maintained row ID to use it with an ArcSDE client application such as ArcGIS Desktop.

Rename a table

The rename operation can be used to change the name of a table. In the following example, the table world is renamed to world2000.

```
c:\> sdetable -o rename -t world -T world2000 -u dart -p wolfy
```

Note: Not all DBMSs support table renaming.

Remove access to a table

You can take away access to a table from a user with the revoke operation. In the example below, user doc's UPDATE privilege to table victoria is revoked.

```
$ sdetable -o revoke -t victoria -U doc -A UPDATE -u av
```

Delete all records from a table

To remove all records from a table, use the truncate operation. Truncating deletes all the records of the business table and the associated feature class, leaving the definition of the table and feature class intact. In the following example, all records from the table victoria are removed but the table definition will remain in the database.

```
$ sdetable -o truncate -t victoria -u av -p mo -s ultra -i esri_40
```

Remove a table from the ArcSDE table registry

Use unregister to remove a table from the TABLE_REGISTRY system table.

```
$ sdetable -o unregister -t world -u av -p mo
```

Note: Tables cannot be unregistered if they have been registered as multiversioned, they are part of a geodatabase object, or they have a spatial column. In all cases, the dependencies must be removed first or you must use the -F option.

To manually remove the dependencies, unregister the table from the geodatabase using ArcCatalog (if it has been registered with the geodatabase) and remove the spatial column (if it has one) using the sdelayer -o delete operation.

Alternatively, you can use the -F option to force the unregistering of the table. However, all edits stored in the delta tables will be lost. The table will not be unregistered with the geodatabase.

Update database statistics

The update_dbms_stats updates RDBMS table and/or index statistics. These are specific to the type of DBMS you are using.

Here, statistics on all indexes are being updated for the table dist_centers in an Oracle database:

```
$ sdetable -o update_dbms_stats -t dist_centers -n ALL
```

Note: Beginning with ArcSDE 9.2 for Oracle, update_dbms_stats uses the Oracle DBMS_STATS package not ANALYZE. Consult your Oracle 9*i* or 10*g* documentation for details on what the DBMS_STATS package does.

On SQL Server databases, the syntax used with the -m option corresponds to that used with the Transact-SQL statement UPDATE STATISTICS. Include "with" in the -m option as shown in this example, in which statistics are being updated on table students using full scan:

```
c:\> sdetable -o update_dbms_stats -t students -m "with fullscan"
```

Fullscan specifies that all the rows in the table are used to gather the statistics. If you are updating statistics on a very large table, other options you might use are as follows:

- sample <#> percent, which uses the specified percent of the table to use when collecting statistics. For example:

```
c:\> sdetable -o update_dbms_stats -t books -m "with sample
50 percent"
```

- sample <#> rows, which uses the specified number of rows to collect statistics
- resample, which uses a sampling ratio

In the following example, statistics are updated in medium mode on the table wells in an Informix database:

```
$ sdetable -o update_dbms_stats -t wells -m medium
```

For tips on when to use low, medium, or high, consult your IBM Informix documentation on updating statistics.

In the next example, the index statistics for table orchards in a DB2 database are being updated:

```
c:\> sdetable -o update_dbms_stats -t orchards -n orchard_idx -u pp
-p llama
```

sdeversion

The **sdeversion** command allows the ArcSDE administrator to manage versions.

Usage syntax

```
sdeversion -o alter -V <version_name> [-d <description>]
[-A {public | protected | private}] [-N]
-u <DB_user_name> [-p <DB_user_password>] [-q]
[-i <service>] [-s <server_name>] [-D <database>]


sdeversion -o create -V <version_name> -P <parent_version> [-U]
[-A {public | protected | private}]
[-d <description>]
-u <DB_user_name> [-p <DB_user_password>] [-q]
[-i <service>] [-s <server_name>] [-D <database>]


sdeversion -o compress [-N] -u <ArcSDE_Admin_user_name> [-p
<ArcSDE_Admin_user_password>] [-q]
[-i <service>] [-s <server_name>] [-D <database>]


sdeversion -o delete -V <version_name> [-N]
-u <DB_user_name> [-p <DB_user_password>] [-q]
[-i <service>] [-s <server_name>] [-D <database>]


sdeversion -o describe [-V <version_name>] [-w <"where_clause">]
-u <DB_user_name> [-p <DB_user_password>] [-q]
[-i <service>] [-s <server_name>] [-D <database>]


sdeversion -o rename -V <old_version_name> -n <new_version_name> [-
N]
-u <DB_user_name> [-p <DB_user_password>] [-q]
[-i <service>] [-s <server_name>] [-D <database>]


sdeversion -h

sdeversion -?
```

Operations

Operation	Description
alter	Allows you to alter the definition of a version
compress	Compresses the database's state tree
create	Creates a new version
delete	Deletes an existing version
describe	Shows version descriptions and lists the current definition of an existing version
rename	Renames an existing version

Options

Options	Description
-A	Access: public, protected, or private
-d	Version description
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".

-i	ArcSDE service name or direct connect information
-n	New version name
-N	No verification is performed; the operation begins immediately after being invoked
-o	Operation
-p	DBMS user password For the compress operation, this is the DBMS user password of the ArcSDE administrator
-P	Parent version name
-q	Quiet—all titles and warnings are suppressed.
-s	ArcSDE server host name (default: localhost)
-U	Generate unique version name by adding a suffix if necessary.
-u	DBMS user name For the compress operation, this is the DBMS user of the ArcSDE administrator
-V	Version name Version names are case sensitive. For example, SDE.DEFAULT and SDE.default are different versions.
-w	SQL WHERE clause

Discussion

The sdeversion command manages versions of a geodatabase. For more information on versioning, see the ArcGIS Desktop Help, which can be accessed within ArcMap or ArcCatalog or from the ESRI support site. (http://support.esri.com)

Examples

Alter a version's definition

The alter operation allows you to alter the definition of a version. For example, you may want to make the new version available to some coworkers but with read-only access permissions. Only the version owner can alter a version. In the following example, version MY_VERSION, the access level is being changed to protected.

```
$ sdeversion -o alter -V MY_VERSION -A protected -u sde -p sde
```

Compress the database's state tree

Versioned databases need to be compressed to maintain performance. To do this, you can use the compress operation. You must be the ArcSDE administrator to perform a compress operation. Note that if you are using geodatabases stored in a user's schema with an Oracle database, the ArcSDE administrator is the schema owner. Also note, that if you are using SQL Server 2005, your ArcSDE administrator may be dbo.

```
$ sdeversion -o compress -N -u sde -p jackaninny
```

The database's state tree is compressed by removing all states that are no longer referenced by a version. Branches not referenced by a version are removed, and nonbranching sequences between versions are trimmed.

Create a new version

In the following example, a new private version is created—JIMS_VERSION—based on version SDE.DEFAULT. Since this will be a private version, only the person who creates it will have access to it.

```
$ sdeversion -o create -V JIMS_VERSION -P SDE.DEFAULT -A private -d
"Overpass Design- Phase II" -u sde -p sde
```

The -d option allows you to add some descriptive text when creating the new version. This provides useful documentation as to the reasons for creating the new version.

Delete a version

The delete operation allows you to delete an existing version. Only the version owner can delete a version. Here, the ArcSDE administrator has decided to delete the version he just created above.

```
$ sdeversion -o delete -V JIMS_VERSION -u jim -p sde
```

Get the description and definition of a version

The describe operation shows a description and lists the current definition of an existing version. This example shows the description and definition of an SDE.DEFAULT version.

```
c:\> sdeversion -o describe -V SDE.DEFAULT -u sde -p sde129

ArcSDE 9.2 Build 625 Mon Oct 2 22:33:27 PDT 2005
Version Administration Utility
-----------------------------------------------------
Instance default version.
-----------------------------------------------------
Version Name: SDE.DEFAULT
Version ID: 1
Parent Version Name:
Parent Version ID: 0
State ID: 3422
Access: Public
Creation Time: 09/01/99 11:16:28
```

Rename a version

Use the rename operation to change the name of an existing version. Only the version owner can rename a version.

```
c:\> sdeversion -o rename -V MY_VERSION -n PATS_VERSION -u sde -p
sde
```

ArcSDE Administration Commands

sdexinfo

The **sdexinfo** command describes an ArcSDE export file.

Usage syntax

```
sdexinfo [-o describe] -f <{export_file | -}>

sdexinfo -o list -f <{export_file | -}> [-s] [-a]

sdexinfo -o stats -f <{export_file | -}>

sdexinfo -h

sdexinfo -?
```

Operations

Operation	**Description**
describe	Lists the export file header and attribute column definitions (the default)
list	Performs the stats operation and lists partial or complete feature and attribute data values
stats	Performs the describe operation and lists statistical information

Options

Options	Description
-a	Lists the attribute data values
-f	ArcSDE export file name; if "-", read from standard input
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-o	Operation
-s	Displays a detailed shape feature definition Displays all fields of the feature definition including the point values

Discussion

The sdexinfo command allows you to obtain information about an ArcSDE export file.

Examples

Each of the operations performed with the sdexinfo command provides a description of the export file.

Describe the export file

Use the describe operation to get a list of the header and column definitions.

```
c:\> sdexinfo -o describe -f c:\GISData\xfaults
---------------------------------------------------
SDEX File:                    c:\GISData\xfaults
Version:                      SDEX 9.0.0.2
Type:                         Final
Volume Type:                  Single
Layer ID:                     3
```

```
   Description:
   Database Name:               SDE
   Table Owner:                 RJP
   Table Name:                  ORFAULTS
   Spatial Column Name:         SHAPE
   Minimum Shape ID:            1
   Grid Size 0, 1, 2:           6790.25,       0,       0
   Entity Type:                 nslc+
   I/O Mode:                    Normal
   Layer Config. Keyword:       DEFAULTS
   Layer Precision:             64-bit
   User Privileges:             SELECT, UPDATE, INSERT, DELETE
   Creation Date:               08/08/05 07:19:43
   Layer Envelope:
     minx: 480039.87500, miny: 5454508.50000
     maxx: 856992.56300, maxy: 5937025.00000
   XY False Origin:             479999.999, 3759999.99
   XY System Units:             1000.0
   XY Half SysUnit:             0.5
   XY Round:                    1.0
   Z Offset:                    0.0
   Z Units:                     1.0
   Z Half SysUnit:              0.5
   Z Round:                     1.0
   Measure Offset:              0.0
   Measure Units:               1.0
   Measure Half SysUnit:        0.5
   Measure Round:               1.0
   Spatial Referencd ID:        3
   Coordinate System:           UNKNOWN

   SDE Attribute Columns: 9
   Name           Type      Width   numDecimal   Null?    RowID
   -------------------------------------------------------------
   OBJECTID     SE_INT32     10         0      NOT NULL    SDE
   FNODE_       SE_FLOAT64   11         0
   TNODE_       SE_FLOAT64   11         0
   LPOLY_       SE_FLOAT64   11         0
   RPOLY_       SE_FLOAT64   11         0
   LENGTH       SE_FLOAT64   15         3
   ORFAULT_     SE_FLOAT64   11         0
   ORFAULT_ID   SE_FLOAT64   11         0
   SHAPE        SE_SHAPE      0         0
```

List feature and attribute values

The list operation returns the description list, statistics, and feature and attribute data values. The feature and attribute value list will be detailed if you use the -s option. Otherwise, it returns the same information as the stats operation.

```
   $ sdexinfo -o list -f c:\GISData\xfaults -s
```

Note: This list can be quite long.

Obtain statistics on the export file

Use the stats operation to get statistical information on the export file in addition to a description list.

```
c:\> sdexinfo -o stats -f c:\GISData\xfaults
```

Returns the same description as in the last example but with added information:

```
Spatial Column Statistics:

Total Linear Features:            18239

Total Features:                   18239

Minimum Feature Number:               1
Maximum Feature Number:           18239
Largest Feature:                     30 Points
Smallest Feature:                     2 Points
Average Feature:                      3.56 Points
Minimum Line String Length:           2.756804
Maximum Line String Length:       22454.355594
Average Line String Length:        1404.028506
Layer Envelope:
                     minx:480039.87500, miny:5454508.50000
                     maxx:856992.56300, maxy:5937025.00000
```

sdexml

The **sdexml** command administers XML columns.

Usage syntax

```
sdexml -o add -l <table,column> [-M <mininum_id>] [-k
<config_keyword>]
[-n <index_name> [-f <xml_index_def>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdexml -o alter -l <table,column> [-M <mininum_id>] [-k
<config_keyword>]
[-n <index_name> [-f <xml_index_def>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]


sdexml -o alter_template -f <xml_index_def> -n <template_name>
[-i <service>] [-s <server_name>] [-D <database>]|
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]


sdexml -o create_template -f <xml_index_def> -n <template_name>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdexml -o delete -l <table,column> [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]


sdexml -o delete_template -n <template_name>
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-N] [-q]


sdexml -o {describe | describe_long} [{-O <owner> | -l
<table,column>}]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]


sdexml -o list -l <table,column> -v <sde_row_id> [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]
```

```
sdexml -o optimize -l <table,column> [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]
```

```
sdexml -o {stats | xi_stats} -l <table,column> [-i <service>]
[-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-q]
```

```
sdexml -h
```

```
sdexml -?
```

Operations

Operation	**Description**
add	Adds a new XML column to an existing table
alter	Modifies some properties of the XML column
alter_template	Modifies the XML index template
create_template	Creates an XML index template
delete	Deletes an XML index column
delete_template	Deletes an XML index template
describe	Describes the properties of an XML column

describe_long	Describes the properties of the fields of an XML column
list	Lists the contents of an XML document
optimize	Optimizes the text index on the documents in the XML column
stats	Lists the XML column statistics
xi_stats	Lists the XML column index statistics

Options

Options	Description
-D	Database name (not supported on Oracle)
-f	Name of the XML index definition file
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information

-k	Configuration keyword present in DBTUNE table (default: DEFAULTS)
-l	The business table and its XML column If you are not the owner of the table, you must qualify the table name as owner.table.
-M	Minimum row ID
-n	Index name
-N	No verification is performed; the operation begins immediately after being invoked.
-o	Operation
-O	Owner name
-p	DBMS user password
-q	Quiet—all titles and warnings are suppressed.
-s	ArcSDE server host name (default: localhost)
-u	DBMS user name
-v	The value of the ArcSDE registered row ID column

Discussion

The sdexml command administers XML columns in tables and feature classes in ArcSDE geodatabases. XML columns are designed to store entire XML documents, one per row, in tables or layers managed by ArcSDE. You can use the sdexml command to add or alter XML columns and indexes on the documents stored in those columns, create and maintain XML index templates, generate statistics about XML documents and indexes, and display the contents of individual XML documents. ArcSDE supports multiple XML columns per table or feature class. For definitions of XML terms and more information about XML documents, visit http://www.w3.org.

Indexing XML columns

A full text index is always created on all XML columns. In addition, you can enable searches within specific tags in XML documents by indexing the contents of those tags. This is done with either an XML index definition or an XML index template.

You can create an XML index definition specifically for tags in a single XML column. This is done by identifying the tags you want to index in an index definition file, and using that file with the add or alter operations. (The format of the index definition file is found below.) The index definition must have a name when it is created. This index name is stored in the ArcSDE system tables along with its ID number, index type, text description, and list of tags to index.

You can also use sdexml to create XML index templates and associate them with one or more XML columns. XML index templates are similar to XML index definitions, except they are stored once and used to index one or more XML columns containing similar information. XML index templates often include a list of tags commonly found in standardized XML documents. You can create your own templates using the create_template operation along with an index definition file. XML index templates are stored in the ArcSDE system tables and are associated with XML columns as needed. XML templates need not be associated with any XML columns, so you can create them before creating any XML columns that use them. Like index definitions, each index template must have a name.

Index definition file

An XML index definition file is a text file used to define the new index or index template. It contains a text description of an index or template plus a list of parameters for each tag to be indexed. Only one index or template may be defined in an index definition file.

Index parameters are specified with a series of lines beginning with keywords. Some keywords are optional. Each group of parameters for each tag must start with the ##TAG keyword and end with the keyword END. For missing tag parameters, default values are assigned. The keywords (such as DESCRIPTION) must be in capital letters. Keywords must start in the first column of the line of text. Following is the format of the file:

```
DESCRIPTION: <index description>
##TAG DESCRIPTION: <tag description>
LOCATION PATH: <location path>
DATA TYPE: <STRING or DOUBLE>
ALIAS: <alias>
EXCLUDED: <TRUE or FALSE>
END
{repeat from ##TAG to END for each tag}
```

- The first DESCRIPTION keyword in the file is a short text description of the index. Subsequent DESCRIPTION keywords are for each indexed tag. Each description can be up to 63 characters. All text on the line following the keyword is assumed to be part of the description, including any trailing white space characters. The default values for descriptions are blank.
- LOCATION PATH is the path to the tag to be indexed. This is a required parameter, and there is no default value. The path should not be in

quotes. All text on the line following the keyword is assumed to be part of the location path.

- DATA TYPE is either STRING or DOUBLE, depending on whether text or numeric values are expected to be in the tag. The default value is STRING.
- ALIAS is a numeric value used as an alias for a tag. A value of zero means that there is no alias, which is the default value.
- EXCLUDED determines whether this tag will be indexed. This is useful for temporarily excluding a tag without removing its definition from the file.

Examples

Add an XML column

The add operation adds a new XML column to an existing table. You can specify a tag index when creating the column or add one later with the alter operation.

There are several variations on how to specify an index definition or index template to use with the XML column. In the first example, an XML column called myxmlcol is added to a table called mytable. No index will be created for the tags in the XML column.

```
$ sdexml -o add -l mytable,myxmlcol -u me -p mypw
```

The second example creates an XML column and an index definition on that column. You must specify a name for the index using the -n option.

```
$ sdexml -o add -l mytable,myxmlcol -f index_paths.txt -n
mytbl_myxmlcol_i -u me -p mypw
```

The third example associates an existing XML index template to be used to index the tags in the XML column.

```
$ sdexml -o add -l mytable,myxmlcol -n template1 -u me -p mypw
```

Alter properties of an XML column

Use the alter operation to change the index or the minimum ID values for documents in an existing XML column.

The following example replaces the existing index definition (if any) on the XML column myxmlcol with an index template called template1. The XML index template template1 must already exist.

```
$ sdexml -o alter -l mytable,myxmlcol -n template1 -u me -p mypw
```

If you alter the index using an index definition file, the file must include the complete new index definition.

Modify an XML index template

Use the alter_template operation to modify an XML index template. You must provide a complete definition for the template in the index definition file. You cannot alter a template that is currently used by any XML columns.

```
$ sdexml -o alter_template -f template1_paths_new.txt -n template1 -
u me -p mypw
```

Create an XML index template

Creating an XML index template is very similar to creating an XML index definition for a column you are adding or altering. XML index templates must have a name, just as XML indexes do. Provide a complete definition for the template in an index definition file. This example shows the creation of template1 with the index definition file template1_paths.txt:

```
$ sdexml -o create_template -f template1_paths.txt -n template1 -u
me -p mypw
```

Delete an XML index column

The delete operation drops the XML column and its index. All XML documents in the column are deleted.

```
c:\> sdexml -o delete -l mytable,myxmlcol -u me -p mypw
```

Delete an XML index template

The delete_template operation drops the XML index template from the ArcSDE system tables. You cannot delete a template that is currently used by any XML columns.

```
$ sdexml -o delete_template -n template1 -u me -p mypw
```

Describe an XML column and its index

The describe and describe_long operations display information about the XML columns and their indexes. If the table and column name are specified with the -l option, the specified XML column is described. If, instead, you specify a table owner with the -O option, this operation describes all XML columns in all tables owned by that owner and accessible by the connecting user. If neither -l nor -O options are specified, all XML columns accessible by the connecting user are described.

The describe operation displays:

- Table owner, table name, and XML column name
- Database name (if applicable)
- Privileges that user has on the XML column
- The configuration keyword (DBTUNE) used when the XML column was created
- The minimum XML ID for the column

- Whether the column has an associated XML index and, if so, the name and type of the index (index definition or index template)

  ```
  c:\> sdexml -o describe -l sde.2005,xml -i 5151 -s circus -u
  sde -p sdepass

  ArcSDE for SQL Server  Wed Aug 10  16:00:46  2005
  XML         Administration Utility
  -------------------------------------------------
  Database                    : SDE
  Table Owner                 : SDE
  Table Name                  : 2005
  XML Column                  : XML
  User Privileges             : SELECT, UPDATE, INSERT, DELETE
  XML Configuration           : DEFAULTS
  XML column is indexed.
  index name                  : SDE.ims_xml134
  index type                  : Index Definition
  ```

The describe_long operation displays all the information that the describe operation displays. It also displays:

- Index description (text)
- For every indexed tag:
  - Tag name (path)
  - Data type
  - Description (if any)
  - >Alias (if any)
  - A note if the tag is excluded

List the content of an XML document

The list operation displays the formatted contents of the XML document that is in the specified column and row. Specify the row using the value in the registered row ID column for the table. The sdexml command displays a message—XML document for row nnn:—followed by a formatted list of the XML document. If the content of the column in the specified row is NULL, this message—XML document is NULL—is displayed.

```
c:\> sdexml -o list -l sde.demo,xml -v 403 -i 5151 -s circus -u sde
-p sdepass
```

Optimize the text index

The optimize operation optimizes the tag indexes on the documents in the XML column. You can continue querying from XML columns while optimization is running. This operation will help maintain the best possible search performance for users of Oracle databases by solving fragmentation issues due to frequent inserts/updates to the XML column. Users with other databases need not use this command. Using this command with other database types will return this message: XML column text index optimization not supported on this RDBMS.

Run the optimize operation after loading XML documents. You should also optimize the tag indexes when the count of new or changed XML documents in the database is more than 20 percent of the total number of XML documents.

```
$ sdexml -o optimize -l counties,xml -i esri_sde -s egan -u sde -p
sigh
```

List XML column statistics

The stats operation displays statistics about all documents in the specified XML column. The document lengths are expressed in bytes.

```
$ sdexml -o stats -l mytable2,myxmlcol -u me -p mypw


Number of XML documents found : 323

Number of NULL documents found : 1

Total number of rows : 324

Average length of documents : 2309.25

Minimum document length : 190

Maximum document length : 45678
```

List XML column index statistics

The xi_stats operation displays statistics about the indexes on all documents in the specified XML column. For each indexed tag, sdexml displays the total number of times the tag appears in all documents, the number of documents it appears in, and the number of unique values contained by the tag. A summary is also displayed. If we ran the example above with the xi_stats operation instead of the stats operation, we would get the same results plus this summary:

```
tag count doc count unique values tag name

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

2 2 2 /metadata/county/name

Number of unique tag names : 1

Total number of tags indexed : 2

Number of XML documents indexed : 2
```

shp2sde

The **shp2sde** command converts an ESRI shapefile to an ArcSDE geodatabase feature class. It will convert both the attributes and the geometry.

Usage syntax

```
shp2sde -o append -l <table,column> [-V <version_name>]
-f <shape_file> [-I] [-a {none | all | file=<file_name>}]
[-r <reject_shpfile>] [-c <commit_interval>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-v]


shp2sde -o create -l <table,column> -f <shape_file> [-I]
[Spatial_Index] [{-R <SRID> | [Spatial_Ref_Opts]}]
[-S <layer_description_str>] [-v] [-L {ON | OFF}] [-P {HIGH | LOW}]
[-e <entity_mask>] [-k <config_keyword>] [-M <minimum_ID>]
[-a {none | all | file=<file_name>}] [-r <reject_shpfile>]
[-V <version_name>][-C <row_id_column>[,{SDE | USER},<min_ID>]]
[-c <commit_interval>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]


shp2sde -o init -l <table,column> -f <shape_file> [-I] [-v]
[-a {none | all | file=<file_name>}] [-r <reject_shpfile>]
[-c <commit_interval>] [-i <service>] [-s <server_name>]
[-D <database>] -u <DB_user_name> [-p <DB_user_password>]
```

Where

```
[Spatial_Ref_Opts] := [-x <xoffset,yoffset,xyscale>]
[-z <zoffset,zscale>] [-m <moffset,mscale>]
[-G {<projection_ID> | file=<proj_file_name>}]

[Spatial_Index] := [-g [Grid_Options] | GRID,[Grid_Options] |
AUTOMATIC | NONE | RTREE ]

[Grid_Options] : = [<grid_sz0>[,<grid_sz1>[,<grid_sz2>]] [,][{FULL |
SPARSE}]


shp2sde -h

shp2sde -?
```

Conversion table

Shape to ArcSDE entity (feature) type mapping	
Shape	**ArcSDE entity (feature)**
point	point
multipoint	multipart points
arc	1:n lines (spaghetti) or simple lines (line strings) **Note:** If the feature class allows both lines and simple line features, all features that pass validation are stored as simple line features, while the rest are stored as lines. If the feature class supports only lines, all features are stored as lines. If the feature class supports only simple line features, all features not able to pass verification are discarded.
polygon	1:n area features
nil	nil

Operations

Operation	Description
append	Adds features to an existing feature class (the default)
create	Creates a new feature class and imports features from the shapefile into it; an error is returned if the feature class already exists.

init	Deletes all features of an existing feature class before importing new features (not allowed on versioned data)

Options

Options	Description
-a	**Attribute modes** *none:* Does not load any attributes (the default) The business table of the feature class is populated with the shape records, and the spatial column is populated with a sequential number. This option allows import of data created with the -a none option of the sde2shp command in versions 2.1 or 2.1.1. *all*: Loads all attribute columns If an attribute table doesn't exist for the feature class, one is created. Otherwise, the incoming schema must be union compatible with the table if using the append or init option. *file=<file_name>:* File containing lines in the form <shape_column [sde_column] The shape_column selects the column to be output, while sde_column specifies the column to load to. File containing lines of the form: <shpCol> [sdeCol] [type] [size] [nDecs] [NOT_NULL] The shpCol selects the column to be output, while the sdeCol is the new name of the column. The type specifies a legal type conversion. The size is the maximum size of the column and nDecs the number of digits to the right of the decimal point for floating point data types. NOT NULL, if specified, requires that the column must have a non-NULL value.
-c	Commit rate (default: AUTOCOMMIT value from giomgr.defs)
-C	Row ID's column_name, column_type, and minimum_id
-D	Database name (not supported on Oracle)
-e	Entity types allowed (npsla3+M)

	n - Nil p - Point shapes s - Line (spaghetti) shapes l - Simple line (line string) shapes a - Area shapes 3 - Three-dimensional shapes, which can be added to the entity type mask with the -o add operation only + - Multipart shapes M - Measures on coordinates. The -m option is required if measures on coordinates is a defined shape type.
-f	Path to and name of the shapefile
-g	Spatial index type and parameters

GRID,<grid_sz1>[,<grid_sz2>[,<grid_sz3>]],[FULL|SPARSE]	Creates multilevel grid index (Grid2 and grid3 are optional.) If you don't enter grid values, it is the same as specifying an AUTOMATIC spatial index type. FULL grids create a spatial index grid on the entire feature envelope. This is the default option if GRID is specified and is the type of grid always created in ArcSDE 9.1 and lower. SPARSE grids create spatial

		index grids only where grids actually include parts of the feature. See the Discussion section for more details.
	AUTOMATIC	For Oracle Spatial and Informix, this is RTREE. For all other storage types, if the feature class is in normal IO mode and there are 0 features, the grid type is NONE. If one or more features is present and the feature class is in normal IO mode, grid sizes will be calculated and set based on the calculations below. If the feature class is in load_only IO mode, grid sizes will be calculated and set based on the calculations below when the feature class is

		returned to normal IO mode.
		If there are no valid envelopes detected, grid sizes are calculated as:
		
		grid_size1 = 1000000.0 / layerGrid->xyUnits
		grid_size2 = 1000000000.0 / layerGrid->xyUnits
		
		If it is a point feature class, grid sizes are calculated as:
		
		grid_size1 = 1000.0 / layerGrid->xyUnits

	NONE	No spatial index is created
	RTREE	Creates Rtree index
-G	Coordinate system specifier: <projection_id>coordinate system ID (See the pedef.h file for the integer codes.) file=<proj_file_name>file containing coordinate system description string	
-h or -?	Use either of these options to see the usage and options for the command. Note: If using a C shell, use -h or "-\?".	
-i	ArcSDE service name or direct connect information	

-I	Disable buffered inserts (default: ON).
-k	Configuration keyword present in DBTUNE table (default: DEFAULTS); used with the create operation only.
-l	Feature class and spatial column to load They must exist, and the user must either own the table or have INSERT access to it. If you do not own the table, qualify the table name as owner.table.
-L	Enables or disables autolocking Set to ON, autolocking is enabled; set to OFF autolocking is disabled. When autolocking is enabled, the session attempts to lock the area defined by a feature envelope before the feature can be edited. If the area is already locked, an error is returned. (default: ON)
-m	Measure offset and scale, separated by a comma If the -m is not specified, the measure offset is calculated from the shapefile's minimum measure value, and the measure scale is calculated from the shapefile's measure delta. If these values cannot be obtained, the measure offset and scale is set to 0.0 and 1.0.
-M	Minimum ID New shape IDs are assigned the larger of the minimum ID or the maximum assigned ID + one. (default: 1)
-o	Operation
-p	DBMS user password
-P	The internal storage of the feature geometry, either LOW (32bit) or HIGH (64bit) (default: HIGH)
-r	Rejects shapefile name for rejected shapes Shapes are only written to the rejects file if they can be. If the originating shapefile was not created according to the ESRI shapefile specification, it may not be possible to write a rejected shape record to the rejects shapefile. For instance, if the shapefile contains a numeric field that exceeds a width of 19, the rejects shapefile cannot be created.
-R	Spatial reference ID (SRID)
-s	ArcSDE server host name (default: localhost)

-S	Feature class description string
-u	DBMS user name
-v	Verbose option—reports records committed at the commit interval
-V	Geodatabase version name Version names are case sensitive. For example, SDE.DEFAULT and SDE.default are different versions.
-x	The x-offset, y-offset, and x,y scale values If the -x option is not specified, the x- and y-offset is calculated from the minimum x- and y-coordinate values of the shapefile, and the x,y scale is calculated from the maximum x or y delta value. If the x and y delta values cannot be obtained from the shapefile, the default is 0.0,0.0,1000.
-z	The z-offset and scale values separated by a comma If the -z option is not specified, the z-offset is calculated as the shapefile's minimum z-value and the z-scale is calculated from the shapefile's z delta. If the values cannot be obtained from the shapefile, the default is 0.0, 1.0.

ArcSDE support for database file types

Type	**Width**	**Description**	**Storage**
B	10	Binary field	No
C	1-254	Character	Yes
D	8	Date field specified as 8 ASCII characters in YYYYMMDD format	Yes

F	1-20	Numeric floating point field	See below
G	10	General field	No
L	1	Logical field	Yes
M	10	Memo field	No
N	1-19	Numeric fixed position field	Yes
P	10	Picture field	No
V	10	Variable field	No

The dBASE F-type column will not be created in the geodatabase, but the geodatabase will write into an existing F-type column. Therefore, the sde2shp init operation will not create an F-type column, but the sde2shp append operation will write to a shapefile containing an existing F-type. The shp2sde command will read an F-type column.

Discussion

The shp2sde command converts shapefiles to multiusre geodatabase feature classes.

The shp2sde command can convert up to 15 digits of precision or the maximum precision imposed by the SE_DOUBLE data type.

The row ID column of a business table must be a unique integer column. The row ID column is always required if the feature class is stored in a DB2 or Informix DBMS, an Oracle DBMS using the ST_Geometry spatial type, or if the feature class contains an Oracle Spatial SDO_GEOMETRY column. Feature classes stored in the SQL Server DBMS and the Oracle DBMS (except those that have an SDO_GEOMETRY column or ST_Geometry column) are not required to have a row ID defined unless they are registered with the geodatabase or they are to be registered as multiversioned.

For those feature classes required to have a row ID defined, by default, shp2sde creates a unique integer column, maintained by ArcSDE, called objectid with an initial value set to 1. The -C option allows you to control the name, type, and initial value of the row ID column.

If the -C option is specified, the column name can be any name that follows the DBMS naming convention. The name defaults to objectid. The type can be either maintained by ArcSDE (ArcSDE controls the sequence of values entered into the column, which ArcGIS requires of any table registered with the geodatabase), or user maintained (your application controls the sequence of values entered into the row ID.) The type defaults to having ArcSDE maintain the row ID. The initial value of the row ID can be a positive integer. The initial value defaults to 1.

In ArcSDE 9.1 or lower, if you did not use the -C option to specify a column to use for the row ID, shp2sde would search for an existing column named objectid and use that for the row ID. If such a column didn't exist, it would create one. Beginning with ArcGIS Server 9.2, shp2sde will not search for and use an existing column named objectid. If a column named objectid already exists in the shapefile, shp2sde will add a column named objectid1 to use as the row ID. Therefore, if your shapefile contains a unique integer column named objectid that you want to use for the row ID, you **must** use the -C option to specify it as your row ID.

The following is an example of a row ID column being created with the -C option. The row ID column is called wid; it is maintained by ArcSDE, and its initial value is set at 1. If the -C option is not specified in a situation in which the row ID column is required, the row ID column would be named objectid, maintained by ArcSDE, and its initial value would be set to 1.

```
$ shp2sde -o create -l borders,feature -f world -C wid,SDE,1 -e -u
world
```

In this example, an existing objectid column is specified with the -C option.

```
c:\> shp2sde -o create -l studyareas,feature -f c:\data\sites -C
objecid,SDE,1 -e -u prof
```

The shp2sde command reports the status of the shapefile to feature class transfer in the form of features converted and features stored. The meaning of these statistics is as follows:

- Features converted: The number of shapefile records processed and converted, partially or wholly, into an ArcSDE feature class
- Features stored: The number of features actually stored in the feature class

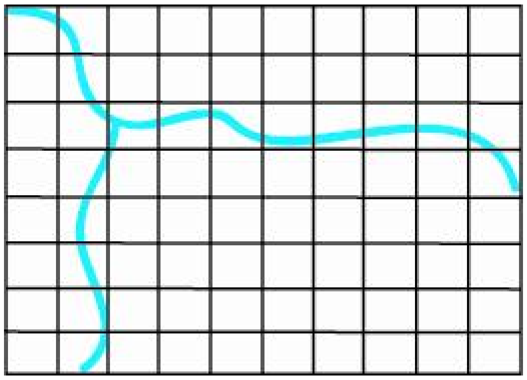
The two values can differ if the shapes within a multipart shapefile are separated and stored as several features. This occurs if the feature class entity type does not allow multipart features. To allow multipart features to be stored in the feature class, change the feature class entity type with either the sdelayer command or by adjusting the properties of the feature class with the ArcCatalog application.

Note: The metadata for a feature class in an ArcSDE geodatabase is stored in a table called GDB_USERMETADATA. Only applications written with ArcObjects—such as ArcCatalog—can write to the geodatabase (GDB) tables, including the

GDB_USERMETADATA table. The shp2sde command is not an ArcObjects application, is not geodatabase aware, and, therefore, will not write the metadata contained in the XML file associated with the shapefile to the geodatabase.

Spatial index grids

Beginning with ArcGIS Server 9.2, you have the option to create sparse spatial index grids. Sparse grids populate the spatial index with information for only those grids that actually contain a portion of a feature. For example, if you created a spatial index on a river feature class, only those grids that the river crossed would be indexed for the river.



This could be more efficient for spatial queries, because grids that do not qualify as crossing the feature (in this case, rivers) will be eliminated from the query results during the primary filter process ArcSDE performs when executing spatial queries. Be aware that if you use third-party applications to perform envelope-on-envelope spatial queries, using a sparse index grid will affect query results.

If instead you used a full grid for the rivers, the nonqualifying grids would still be eliminated, but they wouldn't be eliminated until the secondary filter process.

Examples

Append features to a feature class

The append operation adds features to an existing feature class. In the following example, the features from the shapefile vireo are appended to the nests table. The -a option specifies that none of the attributes from the vireo shapefile will be brought into the nests' business table; only the spatial information will be populated. This might be useful if you plan to collect more up-to-date field data about the vireos after adding the spatial information to the nests feature class.

```
c:\> shp2sde -o append -l nests,shape -f vireo -a none -u av -v
```

By specifying the -v option, you will see a report of the records committed at the commit interval. Since -c isn't specified, the commit rate defaults to the AUTOCOMMIT value in the SERVER_CONFIG table (SDE_server_config table in SQL Server databases).

Create a new feature class and import features from a shapefile

The create operation creates a new feature class and imports features from the specified shapefile. If the feature class you are trying to create already exists in the geodatabase, an error is returned.

This example creates a feature class called blocks and imports features from a shapefile called census_data in geographic coordinates:

```
$ shp2sde -o create -l blocks,shape -f census_data -a all -x -200,-
100,100000 -g GRID,1000,SPARSE -G 4269 -e a -k block_attr -u av
```

This example converts the world shapefile to a feature class called borders. The data is small scale, so the scale (in the -x option) is set to 10,000. The example also uses the -r option to write any rejected shapes to a new shapefile called rejects.

```
$ shp2sde -o create -l borders,feature -f world -g AUTOMATIC -x -
180,-90,10000 -e a -k WORLD -a all -r rejects -s stout -u world
```

Delete features of an existing feature class before importing new features

The init operation deletes all features of an existing feature class before importing new features. This operation cannot be done on versioned data.

```
c:\> shp2sde -o init -l stations,feature -f trains -a all -u av
```

In the previous example, the -a option is specifying that all attribute columns be imported to the stations feature class. Remember, the incoming schema of the trains shapefile must be union compatible with the stations' attribute table for this to work.

tbl2sde

The **tbl2sde** command converts a table to an ArcSDE geodatabase table. The input table format may be a geodatabase table.

Usage syntax

```
tbl2sde -o append -t <table> -f <file_name> -T {dBASE | INFO | SDE}
[-I] [-a {all | file=<file_name>}]
[-c <commit_interval>] [-i <service>]
[-s <server_name>] [-D <database>] -u <DB_user_name>
[-p <DB_user_password>] [-v] [-w <"where_clause">]


tbl2sde -o create -t <table> -f <file_name> -T {dBASE | INFO | SDE}
[-I] [-a {all | file=<file_name>}] [-c <commit_interval>]
[-k <config_keyword>] [-w <"where_clause">]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-v]


tbl2sde -o init -t <table> -f <file_name> -T {dBASE | INFO | SDE}
[-I] [-a {all | file=<file_name>}] [-c <commit_interval>]
[-i <service>] [-s <server_name>] [-D <database>]
-u <DB_user_name> [-p <DB_user_password>] [-v]
[-w <"where_clause">]


tbl2sde -h

tbl2sde -?
```

Operations

Operation	**Description**
append	Adds records to an existing DBMS table (the default)
create	Creates a new table and imports records into it

	An error is returned if the table already exists.
init	Deletes all records in an existing DBMS table before importing new records

Options

Options	**Description**
-a	Attribute modes: *all*: Loads all columns (the default) If the table exists, the incoming schema must be union compatible with the table if using the append or init option. *file=<file_name>*: File containing lines of the form <fr_colName> [to_colName] [type] [size] [nDecs] [NOT_NULL] The <fr_colName> selects the column to export, while the <to_colName> specifies the new column to load to. The allowed type, size, and nDecs (number of decimal places) values will vary according to each DBMS.
-c	Commit rate (default: AUTOCOMMIT value from SERVER_CONFIG table)
-D	Database name (not supported on Oracle)
-f	Input table name Define the table type with the -T option.
-h or -?	Use either of these options to see

	the usage and options for the command. Note: If using a C shell, use -h or "-\?".
-i	ArcSDE service name or direct connect information
-I	Disable buffered inserts (default: ON).
-k	Configuration keyword present in DBTUNE table (default: DEFAULTS)
-o	Operation
-p	DBMS user password
-s	ArcSDE server host name (default: localhost)
-t	Output table name
-T	Input table type, either dBASE, INFO, or SDE
-u	DBMS user name
-v	Verbose option—reports records committed at the commit interval
-w	SQL WHERE clause—only used if the input table type is SDE.

Discussion and examples

The tbl2sde command converts INFO and dBASE tables to ArcSDE geodatabase tables. You can also use tbl2sde to selectively copy columns from an ArcSDE geodatabase table to another ArcSDE geodatabase table. The example below converts a dBASE table called census_data into a table called block_attr.

```
$ tbl2sde -o create -t block_attr -f census_data -T dBASE -a all -k
block_attr -u abc -p mo
```

If the table already exists, you can add more records with the append operation. You can also remove the records from an existing table with the init operation before loading more records. In both cases, the incoming schema must be union compatible with the business table to which it is being imported.

```
c:\> tbl2sde -o append -t sherds -f site3 -T INFO -u abc -p mo

c:\> tbl2sde -o init -t catchbasin -f storms -T SDE -u abc -p mo
```