CHAPTER

7

The GIS Procedure

Introduction

The GIS procedure creates and maintains the spatial databases that are used by SAS/GIS software. A SAS/GIS spatial database consists of

- SAS data sets that contain the coordinates and identifying information for the spatial features.
- A spatial entry, a SAS catalog entry of type GISSPA that identifies which SAS data sets contain spatial information. The spatial entry also stores
  - Composites that define how the variables in the spatial data are used
  - Names of the polygonal indexes that define the boundaries of area layers for the map

- A lattice hierarchy that defines which features in the spatial data enclose or are enclosed by other features (the relationships among the polygonal variables)
- Information about the projection method that is used for the stored spatial data.

A spatial entry alternatively can contain references to two or more other spatial entries that have been merged into a single spatial database.

- A coverage entry, a SAS catalog entry of type GISCOVER that selects a subset of the spatial data that are available for display in a map.
- One or more layer entries, SAS catalog entries of type GISLAYER that identify features that have common characteristics and specify how they are displayed as layers in the map.
- A map entry, a SAS catalog entry of type GISMAP that specifies which layers from a particular coverage are included in a map. The map entry also stores
  - The names of attribute data sets that are associated with the map, along with definitions of how the attribute data are linked to the spatial data
  - The name of a SAS data set that contains labels for map features
  - Definitions of GIS actions that can be performed when map features are selected
  - Definitions for map legends
  - Values for display and projection options.

Note: The task of creating new SAS/GIS spatial databases from spatial data in other formats can also be performed interactively by using the GIS Spatial Data Importing window. △

PROC GIS Syntax Overview

PROC GIS <CATALOG=<*libref.*>*catalog*> ;
CATALOG <*libref.*>*catalog*;
SPATIAL <*operation*> <*libref.catalog.*>*spatial-entry* </ *options*>;
COMPOSITE *operation composite* </ *options*>;
POLYGONAL INDEX *operation polygonal-index* </ *options*>;
LATTICE *outer-composite-name-1* ENCLOSES *inner-composite-name-1*
<...*outer-composite-name-n* ENCLOSES *inner-composite-name-n*>
<_UNIVERSE_ ENCLOSES *inner-composite-name*>;
COVERAGE *operation* <*libref.catalog.*>*coverage-entry* </ *options*>;
LAYER *operation* <*libref.catalog.*>*layer-entry* </ *options*>;
MAP *operation* <*libref.catalog.*>*map-entry* </ *options*>;
COPY <*libref.catalog.*>*entry*<.*type*> </ *options*> ;
MOVE <*libref.catalog.*>*entry*<.*type*> </ *options*>;
SYNC <*libref.catalog.*>*entry*<.*type*> </ *options*>;

Note: Optional arguments are enclosed in angle brackets, for example, </ *options*>. △

How Statements Are Processed

The GIS procedure supports RUN-group processing. RUN-group processing enables you to invoke the procedure, then submit additional procedure statements without submitting the PROC statement again.

In other SAS procedures that do not support RUN-group processing, a RUN statement that follows a block of submitted statements terminates the procedure. With RUN-group processing, a RUN statement executes the preceding block of statements, but the procedure remains active. You can continue to submit additional statements for the active procedure without resubmitting the PROC statement. For example, the following code invokes the GIS procedure, assigns a default catalog, and identifies the current spatial entry:

```
proc gis catalog=mymaps.region;
   spatial norwest;
```

Note: The SPATIAL, CATALOG, LATTICE, COPY, MOVE, and SYNC statements are immediate statements for the GIS procedure. That is, they are always processed immediately and do not require a RUN statement (although including a RUN statement does not do any harm). △

After you invoke the GIS procedure, suppose that you also want to define composites. You can submit additional GIS procedure statements to define the composites without submitting a new PROC statement, as shown in the following example:

```
   composite create state /
      class=state
      var=(left=statel,right=stater);
   composite create county /
      class=area
      var=(left=countyl,right=countyr);
   composite create lat /
      class=y var=y;
   composite create lon /
      class=x var=x;
run;
```

You can end RUN-group processing and terminate the GIS procedure by submitting a QUIT statement:

```
quit;
```

Submitting another PROC step, a DATA step, or an ENDSAS statement also ends RUN-group processing and terminates the GIS procedure.

Note: Certain error conditions may also terminate the GIS procedure. If this occurs, a message is printed in the SAS log. △

Data Set Names

You can specify a data set by its complete two-level name as in *libref.data-set*. If you omit *libref*, the data set is assumed to be in the library specified in the CATALOG= option in the PROC GIS statement or in the catalog that was specified by the most recent CATALOG statement.

Note: If a one-level catalog name was used in the CATALOG= option or CATALOG statement, or if no default catalog has been named, the default library is WORK, for example, *WORK.data-set*. △

Catalog Entry Names

You can specify a GIS catalog entry by its complete three-level name, *libref.catalog.entry*. If you use only the one-level *entry*, the entry is assumed to be in the catalog that is specified in the CATALOG= option in the PROC GIS statement or in the catalog specified by the most recent CATALOG statement.

Note: If the *libref* was omitted from the CATALOG= option or catalog statement, the default library is WORK. If no default catalog has been declared, and a one-level *entry* name is used, an error is written to the log because of insufficient information to identify the entry. △

PROC GIS Statement

The PROC GIS statement invokes the GIS procedure and, optionally, specifies the default SAS catalog in which the spatial, coverage, layer, and map entries are stored.

PROC GIS Statement Syntax

PROC GIS <CATALOG=<*libref.*>*catalog-name*>;

where the CATALOG= option can take one of the following forms

CATALOG=<*libref.*>*catalog-name*
CAT=<*libref.*>*catalog-name*
C=<*libref.*>*catalog-name*

Description

Each of the statement forms specifies the default SAS catalog in which are stored GIS spatial, coverage, layer, and map entries referred to in subsequent statements in the PROC GIS step.

If the specified catalog does not already exist, it is created when a subsequent SPATIAL, COVERAGE, LAYER, or MAP statement is executed. If you omit the *libref* argument, the default SAS data library, WORK, is used.

The CATALOG= argument is overridden when you perform one of the following:

- Issue a CATALOG statement in conjunction with the PROC GIS statement. Subsequent statements in the GIS procedure will refer to the catalog that was named in the most recent CATALOG statement rather than to the one that is specified in the CATALOG= option in the PROC GIS statement.
- Specify fully qualified (three-level) entry names in SPATIAL, COVERAGE, LAYER, and MAP statements. This *temporarily* overrides the default catalog for the current statement only. It does not reset any catalog that is specified with the CATALOG= option. See the descriptions of these statements for more information.

CATALOG Statement

CATALOG <CONTENTS> <*libref.*>*catalog-name*;

Description

The CATALOG statement identifies the default SAS catalog in which GIS spatial, coverage, layer, and map entries are stored when you specify one-level catalog entry names in subsequent statements in the PROC step.

Note: The CATALOG statement permanently replaces the CATALOG= option that was specified in the PROC GIS statement. If you use the CATALOG= option in the PROC GIS statement and then submit a CATALOG statement, subsequent statements in the GIS procedure refer to the catalog that was named in the most recent CATALOG statement. △

CONTENTS Operation

CONTENTS displays information about the entries in the specified catalog to the SAS Output window. If a catalog is not specified, CONTENTS displays the entries in the current catalog.

Libref and Catalog-Name Arguments

The arguments <*libref*.>*catalog-name* specify the SAS catalog in which are stored the GIS spatial, coverage, layer, and map entries that are referred to in subsequent statements in the PROC GIS step.

If the specified catalog does not already exist, it is created when a subsequent SPATIAL, COVERAGE, LAYER, or MAP statement is executed. If you omit *libref*, the default SAS data library, WORK, is used.

You can temporarily override the CATALOG statement by specifying fully qualified (three-level) entry names in the SPATIAL, COVERAGE, LAYER, and MAP statements. This does not reset the current default catalog.

SPATIAL Statement

SPATIAL <*operation*> <*libref.catalog.*>*spatial-entry* </ *options*>;

The SPATIAL statement

- Selects the spatial entry on which subsequent statements operate
- Displays information about the contents of a spatial entry
- Creates a new spatial entry, replaces an existing spatial entry, or modifies the characteristics of an existing spatial entry
- Deletes a spatial entry.

Description

A spatial entry is a SAS catalog entry of type GISSPA that defines the components of a SAS/GIS spatial database. The definition specifies which SAS data sets contain spatial information, how the data sets are related, and what roles the variables play.

Any composites, polygonal indexes, and lattice hierarchies that are created or updated during an invocation of the GIS procedure are stored in the current spatial entry. Any subsequent COVERAGE statements that are issued within the PROC GIS step subset the data in the current spatial entry.

No additional arguments (other than the spatial entry name) are used when the *operation* keyword is omitted. An error occurs if there is no existing spatial entry that has the specified name.

Note: When creating or replacing spatial entries, you can either define entirely new spatial entries or merge two or more existing spatial entries. △

SPATIAL Statement Operations

In a SPATIAL statement, the *operation* argument can be one of the following:

- CONTENTS
- CREATE
- DELETE
- REPLACE
- UPDATE

Note: If you omit the *operation* keyword, the SPATIAL statement makes the specified spatial entry the current spatial entry for subsequent operations. No SPATIAL statement options can be used in a spatial assignment statement. △

The following list contains descriptions of the SPATIAL statement operations:

CONTENTS
Prints information about the specified spatial entry to the Output window, including:

- A list of the dependent data objects (data sets or other spatial entries) that store the spatial data
- A list of the SAS data sets (chains, nodes, details, and polygonal indexes) that store the spatial data
- A list of the composites for the spatial data
- The lattice hierarchy for the spatial data
- The storage projection characteristics of the spatial data.

No additional arguments (other than the spatial entry name) are used with this operation. An error occurs if the specified spatial entry does not exist.

Note: The specified spatial entry does not become the current spatial entry for subsequent operations unless no spatial entry is currently selected. △

CREATE
Generates a new spatial entry in which subsequent composites, polygonal index names, and lattice hierarchies that are specified in the GIS procedure are stored. The new spatial entry becomes the current spatial entry for subsequent operations.

An error occurs if a spatial entry with the specified name already exists. The SPATIAL CREATE statement does not overwrite existing spatial entries. Use SPATIAL REPLACE to replace an existing entry.

For a SPATIAL CREATE statement, you must also specify both the CHAINS= and NODES= arguments or the MERGE= argument.

DELETE
Deletes the specified spatial entry. By default, any polygonal index data sets that are referred to in the spatial entry are also deleted. The chains, nodes, or details data sets that are referred to in the spatial entry are not deleted. To retain

existing polygonal index data sets when the spatial entry is deleted, use the KEEP argument in the SPATIAL DELETE statement.

KEEP is the only additional argument (other than the spatial entry name) that can be used with this operation. An error occurs if the specified spatial entry does not exist.

Note: For the DELETE operation, you can also specify the special value **_ALL_** for the spatial entry name argument to delete all spatial entries in the current catalog. △

CAUTION:

Use the DELETE operation with care. The GIS procedure does not prompt you to verify the request before deleting the spatial entry. Be especially careful when you use the **_ALL_** keyword. △

REPLACE

Overwrites the specified spatial entry or creates a new entry if an entry with the specified name does not exist. The specified spatial entry becomes the current spatial entry for subsequent operations. The SPATIAL REPLACE statement has the effect of canceling all previously issued SPATIAL CREATE, COMPOSITE, POLYGONAL INDEX, and LATTICE statements for the specified spatial entry.

For the SPATIAL REPLACE statement, you must specify both the CHAINS= and NODES= arguments or the MERGE= argument.

UPDATE

Modifies the specified spatial entry by applying new values for specified arguments. The updated spatial entry becomes the current spatial entry for the subsequent operations.

An error occurs if there is no existing spatial entry with the specified name.

Spatial-Entry Name

In a SPATIAL statement, the spatial entry name argument identifies the GISSPA-type entry that you want to create, replace, update, delete, or make the current spatial entry. The general form of the argument is

<*libref.catalog.*>*spatial-entry*

CAUTION:

Do not use host commands to move or rename SAS data sets that are referenced in GISSPA entries. Moving or renaming a data set that is referred to in a spatial entry breaks the association between the spatial entry and the data set. To prevent breaking the association, use the PROC GIS MOVE statement with the CHECKPARENT option instead of a host command. △

SPATIAL Statement Optional Arguments

When you specify CREATE, REPLACE, or UPDATE for the *operation* keyword, you can specify one or more of the following optional arguments after the spatial entry name.

Note: Separate the list of arguments from the spatial entry name with a slash (/). △

- CARTESIAN | LATLON
- CHAINS=*data-set*
- DEGREES | RADIANS | SECONDS
- DETAILS=*data-set*
- EAST | WEST

- KEEP
- MERGE=(*spatial-list*) <EDGEMATCH <LINKONLY> | OVERLAP>
- MULT=*multiplier-value*
- NODES=*data-set*
- NORTH | SOUTH
- DESCRIPTION='*string*'

When you specify DELETE for the operation keyword, only the following option is allowed:

- KEEP

The following list contains descriptions of the optional SPATIAL statement arguments:

CARTESIAN | LATLON
Specifies the coordinate system that is used in the stored spatial data.

CARTESIAN
Data are in an arbitrary rectangular (plane) coordinate system

LATLON
Data are in a geographic (spherical) coordinate system.
The default is LATLON.

Note: The CARTESIAN and LATLON arguments are ignored when the MERGE= argument is used. △

CHAINS=*data-set*
Names the SAS data set that contains chain definitions for the spatial database. A chain is one or more line segments that connect one node (or point on the map) to another. For example, a series of chains can represent a railroad or a river.

Note: The CHAINS= argument is required when you use the CREATE or REPLACE keyword and do not specify the MERGE= argument. △

DEGREES | RADIANS | SECONDS
Specifies the coordinate units for the stored spatial data when the coordinate system is geographic (LATLON). The default is RADIANS.

Note: This argument is ignored when the CARTESIAN or MERGE= arguments are used. △

DETAILS=*data-set*
Names the SAS data set that contains detail definitions for the spatial database. Details are the points at angle breaks in chains. They provide a finer granularity for the chain's line segments. A data set that contains detail definitions might describe the curvy outline of a coastal road.

EAST | WEST
Specifies the hemisphere in which the spatial data points lie. The default is EAST. EAST refers to points east of the Prime Meridian (0 degrees) at Greenwich, England, while WEST refers to points west of the Prime Meridian.

If the data are in the Western Hemisphere, longitude values (the X coordinates) are negative, that is -35° 45' 08". If your data are in the Western Hemisphere but have positive longitudes, your map is displayed flipped or with the east and west directions reversed. See Chapter 2, "Preparing Spatial Data," on page 13 for an example of this behavior. Applying the WEST argument to the spatial data causes the longitudes to be negated when the data are read in, and the map is displayed correctly.

Note: This argument is ignored when the CARTESIAN or MERGE= arguments are used. △

KEEP
Specifies that polygonal index data sets are not deleted when the spatial entry is deleted. This option is valid only with the DELETE operation.

MERGE=(<*libref.catalog.*>*spatial-entry-1*<, ..., <*libref.catalog.*>*spatial-entry-n*>)
Lets you build a new spatial entry by referencing two or more existing spatial entries. The dependent data sets for the spatial entries are not actually combined when you use the MERGE argument; the new spatial entry includes them by reference. An error occurs if any of the specified spatial entries do not exist.

You can specify either of the following additional arguments in conjunction with the MERGE= argument:

EDGEMATCH <LINKONLY>
Matches common boundaries between the merged spatial entries. Missing values along common boundary chains are filled in where possible by using values from the adjoining spatial data sets. The affected chains data sets are rewritten unless the LINKONLY option is specified, and you cannot reverse the operation.

OVERLAP
Merges spatial entries without attempting to match boundaries. The chains data sets for the merged entries are not rewritten. This is the default behavior.

MULT=*multiplier-value*
Specifies a constant value by which the stored spatial data coordinates were multiplied. The default is MULT=1.

Note: This argument is ignored when the MERGE= argument is used. △

NODES=*data-set*
Names the SAS data set that contains node definitions for the spatial database. A node is a point on the map, usually representing the intersection of latitude and longitude positions. For example, a node can represent the intersection of two streets.

Note: The NODES= argument is required when you use the CREATE or REPLACE keyword and do not specify the MERGE= argument. △

NORTH | SOUTH
Indicates the hemisphere in which the spatial data points lie. The default is NORTH.

If the data are in the southern hemisphere, latitude values (the Y coordinates) are negative, for example, -45° 12' 33". If your data are in the southern hemisphere, but the latitude values are positive, your map is displayed inverted with the east and west directions reversed. Applying the SOUTH argument to the spatial data causes the latitude values to be negated when the data is read in, and the map is displayed with the correct side up.

Note: This argument is ignored when the CARTESIAN or MERGE= arguments are used. △

DESCRIPTION='*string*'
Specifies a descriptive phrase, up to 256 characters long, that is stored in the description field of the spatial entry. The default description is blank.

SPATIAL Statement Examples

Define the current spatial entry

The following code fragment makes MAPS.NC.NC.GISSPA the current spatial entry that is used for subsequent operations:

```
proc gis cat=maps.nc;
   spatial nc;
```

Update an existing spatial entry

The following code fragment replaces the existing details data set with MAPS.USAD for the existing MAPS.USA.USA.GISSPA spatial entry:

```
spatial update maps.usa.usa / details=maps.usad;
```

Merge three existing spatial databases

The following code fragment creates a new spatial entry that is named TRIANGLE.GISSPA in the current catalog by merging three existing spatial entries, ORANGE, DURHAM, and WAKE. In this example, each of the spatial entries to be merged is stored in a different library. See Chapter 5, “Working with Spatial Data,” on page 55 for more information on merging.

```
spatial create triangle / merge=(gmap1.orange.orange,
                                 gmap2.durham.durham,
                                 gmap3.wake.wake);
```

COMPOSITE Statement

COMPOSITE *operation composite-name* </ *options*>;

Description

The COMPOSITE statement defines, modifies, or deletes associations between variables in the chains and nodes data sets. Once defined, composites can be referenced by other GIS procedure statements.

For example, if a spatial database contains the variables COUNTYL and COUNTYR that identify the chains' left and right values for a county ID variable, you could use the COMPOSITE statement to create a composite called COUNTY by associating the two spatial database variables. The COUNTY composite could then be used to define the county boundaries for the map.

Composites are stored in the currently specified spatial (GISSPA) entry. An error occurs if you submit a COMPOSITE statement when no spatial entry is currently selected.

Note: Use the SPATIAL CONTENTS statement to view the composites for a spatial entry. △

COMPOSITE Statement Operations

In a COMPOSITE statement, the *operation* can be one of the following:

- CREATE
- DELETE
- REPLACE
- UPDATE

The following list contains descriptions of the COMPOSITE statement operations:

CREATE
Defines associations between variables in the chains and nodes data sets and stores these composites in the current spatial entry.

A warning is issued and processing of the current RUN group is halted if a composite with the specified name already exists. The COMPOSITE CREATE statement does not overwrite existing composites. Use COMPOSITE REPLACE to overwrite an existing composite.

Note: Not all spatial database variables are composites of multiple SAS data set variables. Some composites represent a single SAS data set variable. △

DELETE
Deletes the specified composite from the current spatial entry.

No additional arguments (other than the composite name) are used with this operation. A warning is issued and processing of the current RUN group is halted if the specified composite does not exist.

Note: The DELETE operation of the COMPOSITE statement removes a composite from the spatial entry but does not delete the SAS variables from their respective SAS data sets. △

For the DELETE operation, you can also specify the following alternative forms for the *composite-name* argument:

- A list of composite names, separated by spaces, to delete more than one composite in a single DELETE operation.
- The special value **_ALL_** to delete all the composites in the current spatial entry.

CAUTION:
Use DELETE with care. The GIS procedure does not prompt you to verify the request before deleting an existing composite. Be especially careful when you use **_ALL_**. △

REPLACE
Overwrites the previous definition of a composite in the current spatial entry, or creates a new composite if the specified *composite-name* did not previously exist.

UPDATE
Applies new values for the specified arguments to an existing composite.

A warning is issued and processing of the current RUN group is halted if there is no existing composite with the specified name.

composite-name
In a COMPOSITE statement, the *composite-name* argument names the composite that you want to create, replace, delete, or update.

The *composite-name* value must conform to the rules for SAS names:

- The name can be no more than 32 characters long.
- The first character must be a letter or underscore (_). Subsequent characters can be letters, numeric digits, or underscores. Blanks are not permitted.
- Mixed-case names are honored for presentation purposes. However, because any comparison of names is not case-sensitive, you cannot have two names

that differ only in case (for example, State and STATE are read as the same name).

COMPOSITE Statement Optional Arguments

When you specify CREATE, REPLACE, or UPDATE for *operation* in a COMPOSITE statement, you can specify one or more of these options to follow the *composite-name*.

- BILATERAL
- CLASS=
- VAR=

Note: Separate the list of options from the *composite-name* with a slash (/). △

The following list contains descriptions of the additional COMPOSITE statement options:

BILATERAL
Indicates that the composite is a left/right type. BILATERAL composites are used to define polygonal layers in a LAYER statement. This argument provides an implicit VAR= argument, where the LEFT= and RIGHT= variable names are constructed by appending **L** and **R** to the specified composite name. For example, the following two statements are equivalent:

```
composite create state / class=area bilateral;
composite create state / class=area
                         var=(left=statel,right=stater);
```

CLASS=*class-type*
Defines the role of the composite in the spatial database. The CLASS= option links specific functionality to particular composites. The default is CLASS=CLASSIFICATION. See “CLASS=Class-Type” on page 92 for more information on the CLASS type values.

VAR=*association-declaration*
Defines a variable or an association between related variables in the current spatial chains or nodes data set. Variables for all composites are assumed to be in the chains data set except for CLASS=X and CLASS=Y variables, which must be in the nodes data set.

The VAR= argument is required when you use the CREATE or REPLACE operations, except in the following circumstances:

- If you omit the VAR= argument and specify CLASS=CLASSIFICATION (or omit the CLASS= argument), the *composite-name* that you specify is also used as the variable name. For example, the following statements are equivalent:

```
composite create cfcc;
composite create cfcc / var=cfcc class=classification;
```

- If you omit the VAR= argument and specify one of the bilateral *class-type* values such as AREA or STATE, the suffixes **L** and **R** are added to the *composite-name* to form the variable name pair for the association. For example, the following statements are equivalent:

```
composite create state / class=state;
composite create state / class=state
```

```
var=(statel stater);
```

For other *class-type* values, the VAR= argument is required when you use the CREATE or REPLACE keywords. See "VAR=*association-declaration*" on page 94 for more information on the VAR= values.

CLASS=Class-Type

The *class-type* for the CLASS= option can be one of the following:

- ADDRESS
- AREA
- CITY | PLACE
- CLASSIFICATION
- DIRECTION_PREFIX
- DIRECTION_SUFFIX
- NAME
- PLUS4
- TYPE
- X
- Y
- ZIPCODE

The following list contains descriptions of the CLASS=Class-type arguments:

ADDRESS
Indicates that the composite defines addresses in the chains data set that is used for geocoding.

Data set address values are the numeric portion of a street address, for example, the **100** in the street address, **100 North Main Street**. A chain has four values to define the address range for each side:

FROMLEFT	Beginning address on the left side
TOLEFT	Ending address on the left side
FROMRIGHT	Beginning address on the right side
TORIGHT	Ending address on the right side.

When you use specify ADDRESS for the *class-type* value, you must use the following form of the VAR= argument:

VAR=(<FROMLEFT=>*variable*, <FROMRIGHT=>*variable*,
<TOLEFT=>*variable*, <TORIGHT=>*variable*)

AREA
Indicates that the composite defines polygonal areas.

For polygonal areas that represent political subdivisions, you can specify the following alternative *class-type* values to indicate which features the areas represent:

COUNTRY
Indicates that the composite defines countries in the chains data.

COUNTY
Indicates that the composite defines counties in the chains data.

STATE
Indicates that the composite defines states in the chains data. Composites of this class are used in geocoding.

When you use AREA (or COUNTRY, STATE, or COUNTY) for the *class-type* value, you must specify the bilateral form of the VAR= argument to specify the variables that identify the features on the left and right sides of each chain in the area:

VAR=(<LEFT=>*variable*, <RIGHT=>*variable*)

CITY | PLACE
Indicates that the composite defines features that are related to geographic location, such as cities. Composites of this class are used in geocoding.

By default, CITY is not considered an AREA-type composite. If your spatial data contain closed city boundaries, you must explicitly define the composite as an AREA class as well:

```
composite create towns / var=(cityl cityr) class=(city area);
```

CLASSIFICATION
Indicates that the composite defines a general descriptive value that can be used to classify features in the map.

Note: In order to create new point layers when you add points to the map interactively in the GIS Map window, you must define at least one CLASSIFICATION-type composite in the spatial entry. △

DIRECTION_PREFIX
Indicates that the composite defines the directional prefix component of an aggregate feature name, such as the **North** in **North Main Ave**. Composites of this class are used in geocoding.

DIRECTION_SUFFIX
Indicates that the composite defines the direction suffix component of an aggregate feature name, such as the **South** in **2nd St South**. Composites of this class are used in geocoding.

NAME
Indicates that the composite defines the names of features in the chains data, such as **Central Park**, or the name component of an aggregate feature name, such as the **Main** in **E Main St**. Composites of this class are used in geocoding.

PLUS4
Indicates that the composite defines extended postal delivery codes (U.S. ZIP+4) in the chains data. Composites of this class are used in address matching.

By default, PLUS4 is not considered an AREA-type composite. If your chains data contain closed ZIP+4 boundaries, you must explicitly define the composite as an AREA class as well:

```
composite create zip4 / var=(zip4l zip4r) class=(area plus4);
```

TYPE
Indicates that the composite defines the feature type component of an aggregate feature name, such as the **Ave** in **N Harrison Ave**. Composites of this class are used in geocoding.

X
Indicates that the composite defines the X coordinates for the nodes in the nodes data set.

Y
Indicates that the composite defines the Y coordinates for the nodes in the nodes data set.

ZIPCODE
Indicates that the composite defines postal delivery codes in the chains data. Composites of this class are used in geocoding.

By default, ZIPCODE is not considered an AREA-type composite. If your chains data set contains closed ZIP code area boundaries, you must explicitly define the composite as an AREA class as well:

```
composite create zip / var=(zipl zipr)
                       class=(zipcode area);
```

VAR=*association-declaration*

The *association-declaration* argument for the VAR= option can be one of the following, depending on the *class-type* values that are specified in the CLASS= option:

variable
Declares a composite consisting of a single SAS variable. Use this form for single-variable association classes such as CLASSIFICATION, DIRECTION_PREFIX, DIRECTION_SUFFIX, NAME, TYPE, X, and Y.

(<LEFT=>*variable-1*, <RIGHT=>*variable-2*)
Declares a composite consisting of two variables that represent the left and right sides of a feature. Association declarations of this form can be used to define the boundaries between elements in the spatial data. Use this form for bilateral association classes such as AREA, CITY, COUNTRY, COUNTY, PLACE, STATE, ZIPCODE, and PLUS4.

(<FROMLEFT=>*variable-1*, <FROMRIGHT=>*variable-2*, <TOLEFT=>*variable-3*, <TORIGHT=>*variable-4*)
Declares a composite that consists of four variables that separately represent the left and right sides of a feature. Association declarations of this form can be used to define the locations of specific addresses in the spatial data. Use this form for the ADDRESS class.

Variable is the name of a SAS data set variable in the chains data set. An error occurs if any of the specified variables do not exist in the chains data set.

COMPOSITE Statement Examples

Define a single-variable composite

The following code fragment associates the class Y with the variable named LAT in the nodes data set to indicate that the variable contains north-south coordinate information:

```
composite create latitude / var=LAT class=y;
run;
```

Define a composite for a bilateral feature

Either of the following code fragments associates a pair of variables in the chains data set that contain values for the left and right sides of area boundaries:

```
composite create state / var=(left=statel,right=stater)
                          class=area;
run;
composite create state/ bilateral
                        class=area;
run;
```

Define a composite for an address feature

The following code fragment associates two pairs of variables in the chains data set that contain values for the corners of address boundaries:

```
composite create custadd /
   var=(fromleft=FRADDL,fromright=FRADDR,
        toleft=TOADDL,toright=TOADDR)
   class=address;
run;
```

POLYGONAL INDEX Statement

POLYGONAL INDEX *operation polygonal-index </ options>*;

Description

The POLYGONAL INDEX statement creates, replaces, modifies, or deletes polygonal index data sets by using a libref and polygonal index references from a spatial entry. Polygonal indexes delineate enclosed areas in the spatial data by noting the chains that form polygons. This statement is also used to compute the enclosed areas, the centroid coordinates, and the perimeter lengths of the individual polygons.

The spatial database must include a polygonal index data set for each feature type that you intend to represent as an area layer in the map. For example, to represent states and counties as enclosed areas, you must have separate polygonal indexes for each.

The POLYGONAL INDEX statement uses composite values from the current spatial entry to determine area boundaries. The composites that are used for polygonal indexes must have the CLASS attribute AREA (or one of the political subdivision area classes such as COUNTRY, STATE, or COUNTY that imply AREA by default).

Polygonal index definitions are stored in the currently specified spatial entry. An error occurs if you submit a POLYGONAL INDEX statement when no spatial entry is currently selected.

Note: You can use the SPATIAL CONTENTS statement to view the polygonal index definitions for a spatial entry. △

POLYGONAL INDEX Statement Operations

In a POLYGONAL INDEX statement, the *operation* is one of the following:

- CREATE
- DELETE
- REPLACE
- UPDATE

The following list contains descriptions of the POLYGONAL INDEX statement operations:

CREATE
Creates a polygonal index data set and stores the polygonal index definition in the current spatial entry.

A warning is issued and processing of the current RUN group is halted if either a polygonal index definition or a SAS data set with the specified names already exist. The POLYGONAL INDEX CREATE statement does not overwrite existing index definitions or data sets. Use POLYGONAL INDEX REPLACE to replace an existing index definition or data set.

For a POLYGONAL INDEX CREATE statement, you must specify both the COMPOSITE= and OUT= arguments.

DELETE
Removes the specified polygonal index definition from the spatial entry. By default, the POLYGONAL INDEX DELETE statement also deletes the associated index data set. You can use the KEEP option to prevent the index data set from being deleted.

KEEP is the only additional argument (other than the polygonal index name) that can be used with this operation. A warning is issued and processing of the current RUN group is halted if the specified polygonal index does not exist.

For DELETE, you can also specify the special value **_ALL_** for the *polygonal-index* argument to delete all the polygonal index definitions in the current spatial entry.

CAUTION:
Use DELETE with care. The GIS procedure does not prompt you to verify the request before deleting an existing polygonal index. Be especially careful when you use the **_ALL_**. △

REPLACE
Overwrites the polygonal index definition in the current spatial entry or creates a new polygonal index definition if the specified index does not exist.

For a POLYGONAL INDEX REPLACE statement, you must specify both the COMPOSITE= and OUT= arguments.

Note: If the data set that is specified in the OUT= argument already exists and belongs to a different spatial entry, you must specify the FORCE argument to cause it to be overwritten. △

UPDATE
Modifies only the specified characteristics for an existing polygonal index.

A warning is issued and processing of the current RUN group is halted if there is no existing polygonal index with the specified name. If data set that is owned by a different spatial entry that is specified in the OUT= argument already exists, you must use the FORCE argument to cause it to be overwritten.

Polygonal-Index Name

In a POLYGONAL INDEX statement, the *polygonal-index* argument names the polygonal index you want to create, delete, replace, or update.

The *polygonal-index* value must conform to the rules for SAS names:

- The name can be no more than 32 characters long.
- The first character must be a letter or underscore (_). Subsequent characters can be letters, numeric digits, or underscores. Blanks are not permitted.

- Mixed-case names are honored for presentation purposes. However, because any comparison of names is not case-sensitive, you cannot have two names that differ only in case (for example, State and STATE are read as the same name).

POLYGONAL Index Statement Optional Arguments

When you specify CREATE, REPLACE, or UPDATE for the *operation* in a POLYGONAL INDEX statement, you can specify the following additional arguments following the polygonal index name.

- AREA
- CENTROID < = GEOMETRIC | VISUAL >
- COMPOSITE
- ERRORS
- FORCE
- KEEP
- OUT

When you specify DELETE for the operation keyword, only the following option is allowed:

- KEEP

Note: Separate the list of arguments from the polygonal index name with a slash (/). △

The following list contains descriptions of optional POLYGONAL INDEX statement arguments:

AREA
Calculates the enclosed areas and perimeter lengths for the lowest-level area composite that is specify on the COMPOSITE= argument. The calculated area is added to the polygonal index data set in a variable named AREA. A label on the AREA variable contains the storage area units. The calculated perimeters are added to the polygonal index data set in a variable named PERIMETER. A label in the PERIMETER variable contains the units.

CENTROID<=GEOMETRIC | VISUAL>

CENTROID=GEOMETRIC
Returns the actual calculated centroids, which may or may not fall within the boundaries of their corresponding polygons. The coordinates are added to the polygonal index data set in variables that are named CTRX and CTRY. A label in the CTRX and CTRY variables contains the storage projection units indicaties that this is a GEOMETRIC centroid. Specifying the CENTROID argument by itself returns the same results as specifying CENTROID=GEOMETRIC.

CENTROID=VISUAL
Returns adjusted centroids that are guaranteed to fall within the boundaries of their corresponding polygons. The coordinates are added to the polygonal index data set in variables that are named CTRX and CTRY. A label on the CTRX and CTRY variables contains the storage projection units and indicates that this is a VISUAL centroid.

COMPOSITE=(*composite-name-1*<, ..., *composite-name-n*>)
Specifies the composite or list of composites that define the boundaries of the enclosed polygonal areas that are used to create the index. If the *composite-name*

list consists of a single composite, you can omit the parentheses. An error occurs if any of the specified composites are not defined in the current spatial entry or if any do not have the CLASS attribute of AREA.

Note: The COMPOSITE= argument is required when you use the CREATE or REPLACE operation. △

ERRORS<=*number*>
Specifies whether messages about any topological errors that are detected while the index is being constructed are written to the SAS log. A polygon boundary consists of a single chain with the same starting and ending point, or multiple chains that form a closed boundary. A closed polygon boundary must start and end at the same point. A topology error occurs when the polygon is not closed. You can specify the ERRORS argument with no added parameter to print all topological error messages, or you can add the =*number* parameter to specify the maximum number of topological error messages that will be written to the log.

FORCE
Indicates that an existing polygonal index data set that is specified in the OUT= argument can be overwritten, even if it belongs to a different spatial entry. If you omit this option, the data set is not replaced and a warning is issued.

KEEP
Specifies that polygonal index data sets are to be retained when the index definition is removed from the spatial entry. This option is valid only with the DELETE operation.

OUT=*data-set*
Names the index data set that you want to create, replace, or update.

Note: The OUT= argument is required when you use the CREATE or REPLACE operation. △

CAUTION:
Do not use host commands to move or rename polygonal index data sets. Because the polygonal index data set names are stored in GISSPA entries, moving or renaming a polygonal index data set breaks the association between the GISSPA entry and the data set. To prevent breaking the association, use the PROC GIS MOVE statement with the CHECKPARENT option instead of a host command. △

POLYGONAL INDEX Statement Example

The following code fragment builds a polygonal index data set that is named GMAPS.STATEX. The data set identifies the boundaries of the polygons for the area feature that is identified by the STATE composite in the current spatial entry:

```
polygonal index create state / composite=state
                               out=gmaps.statex;
run;
```

LATTICE Statement

LATTICE *outer-composite-name-1* ENCLOSES *inner-composite-name-1*
<...*outer-composite-name-n* ENCLOSES *inner-composite-name-n*>
<_UNIVERSE_ ENCLOSES *inner-composite-name*>;

Description

The LATTICE statement defines the relationships between areas in a spatial database, that is, it defines which areas enclose other smaller areas (such as states enclose counties).

When a lattice hierarchy is defined, the area composite values for new points are assigned automatically as the points are added to the map. The composite values are also reevaluated automatically when an existing point is moved to a new location. A lattice definition also makes it possible to simultaneously assign attribute values to all points in a point layer by setting area attributes in the GIS Layer window. Area attributes cannot be assigned to new points, moved points, geocoded points, or imported points unless a lattice has been defined.

The lattice definition is written to the current spatial entry. If the current spatial entry already has a lattice definition, it is replaced. An error occurs if you submit a LATTICE statement when no spatial entry is currently selected.

Note: Because the LATTICE statement uses composites, you must include a RUN statement following a COMPOSITE statement. This ensures that the composite is created before the LATTICE statement executes and attempts to use the composite. △

Note: You cannot specify an operation argument (for example, UPDATE) in the LATTICE statement. REPLACE is the resulting operation. The only way to delete a lattice from a spatial entry is to execute a SPATIAL REPLACE operation. △

LATTICE Statement Arguments

The following arguments can be used with the LATTICE statement:

outer-composite-name ENCLOSES *inner-composite-name*

UNIVERSE ENCLOSES *inner-composite-name*

An error occurs if there is no current spatial entry for the GIS procedure. Use the SPATIAL statement, omitting the operation keyword, to specify the current spatial entry.

The LATTICE statement checks lattice definitions for circular references. For example, a lattice definition of the following form would cause an error:

```
LATTICE A ENCLOSES B
        B ENCLOSES C
        C ENCLOSES B;
```

The following list contains descriptions of the LATTICE statement arguments:

Outer-composite-name
is an area composite that geographically contains other enclosed AREA type composites. *Outer-composite-name* must have the CLASS attribute AREA (or one of the political subdivision area classes such as COUNTRY, STATE, or COUNTY).

You can also use the special value **_UNIVERSE_** to signify that *inner-composite-name* is a single area composite that is not contained within other enclosed areas and that does not itself enclose any other areas.

ENCLOSES
Is the separator between LATTICE composites. The following symbol can be used in place of ENCLOSES: → .

Inner-composite-name
Is an area composite that is geographically placed within the *outer-composite-name* polygonal areas, or a single area. That is not contained by another when

UNIVERSE is *outer-composite-name*. *Inner-composite-name* must have the CLASS attribute AREA (or one of the political subdivision area classes such as COUNTRY, STATE, or COUNTY).

LATTICE Statement Examples

Single Hierarchy

For a lattice hierarchy that comprises several composites, the general form of the LATTICE statement is

LATTICE A ENCLOSES *B*

B ENCLOSES *C*

C ENCLOSES *D*;

Assume that the spatial database contains states that are subdivided into counties, that the counties are further subdivided into tracts, that the tracts are further subdivided into blocks, and that corresponding composites are defined for each. The following code fragment defines the lattice for the spatial database:

```
lattice state  encloses county
        county encloses tract
        tract  encloses block;
```

Multiple Hierarchies

You can define more than one lattice hierarchy for a spatial database, for example, when the map has overlapping AREA-type composites that are not related. A single LATTICE statement is used, but the GIS procedure recognizes the break between the two hierarchies, as follows:

```
lattice state  encloses county /* first lattice */
        county encloses tract  /* first lattice */
        tract  encloses block  /* first lattice */
        mall   encloses store; /* second unrelated lattice */
```

Single-Element Lattice

If the map has only one AREA-type composite, it is called a universe-enclosed association. Use the **_UNIVERSE_** keyword to define a lattice for a universe-enclosed association, as follows:

```
lattice _universe_ encloses tract;
```

COVERAGE Statement

COVERAGE *operation* <*libref.catalog.*>*coverage-entry* </ *options*>;

Description

The COVERAGE statement

□ Displays information about the contents of a coverage entry

- Creates a new coverage entry, replaces an existing coverage entry, or modifies the characteristics of a previously created coverage entry
- Deletes a coverage entry.

A coverage entry is a SAS catalog entry of type GISCOVER that contains information about the spatial data that are used to create a map. The entry also contains a subsetting WHERE clause to define the subset of spatial data, or coverage, of the map that you want to display.

For example, you could create a coverage entry, MYCAP, that contains geographic information for your state capital. MYCAP subsets the spatial database that is defined in the spatial entry MYSTATE, which contains geographic information that is used to create a map of your entire state.

Note: Even if you want to display the entire geographic scope of your spatial data and not a subset, you must still create a coverage entry by using WHERE='1'. △

COVERAGE Statement Operations

In a COVERAGE statement, the *operation* is one of the following:

- CONTENTS
- CREATE
- DELETE
- REPLACE
- UPDATE

The following list contains descriptions of the COVERAGE statement operations:

CONTENTS
Prints information about the specified coverage entry to the Output window, including the WHERE clause that defines the spatial database subset and details of the spatial database as provided by the SPATIAL CONTENTS statement.

No additional arguments (other than the coverage entry name) are used with this operation. An error occurs if the specified coverage entry does not exist.

CREATE
Creates a new coverage entry.

An error occurs if a coverage entry with the specified name already exists. The CREATE operation does not overwrite existing coverage entries. Use the REPLACE operation to replace an existing entry.

For a COVERAGE CREATE statement, you must also specify the WHERE= argument.

DELETE
Removes the specified coverage entry.

No additional arguments (other than the coverage entry name) are used with this operation. An error occurs if the specified coverage entry does not exist.

For the DELETE operation, you can also specify the special value **`_ALL_`** for the coverage entry name argument to delete all coverage entries in the current catalog.

CAUTION:
Use DELETE with care. The GIS procedure does not prompt you to verify the request before deleting the coverage entry. Be especially careful when you use **`_ALL_`**. △

Note: You must specify new coverages for any map entries that refer to the deleted coverage entry. △

REPLACE
Overwrites the specified coverage entry or creates a new entry if an entry with the specified name does not exist. The REPLACE operation has the effect of canceling the previously issued CREATE operation for the specified coverage entry.

For a REPLACE operation, you must also specify the WHERE= argument.

UPDATE
Modifies the specified coverage entry by applying new values for specified arguments.

An error occurs if there is no existing coverage entry with the specified name.

Coverage-Entry Name

<libref.catalog.>coverage-entry

In a COVERAGE statement, the *coverage-entry* name argument identifies the coverage entry that you want to create, delete, replace, or update. The *coverage-entry* name must conform to the rules for SAS names:

- The name can be no more than 32 characters long.
- The first character must be a letter or underscore (_). Subsequent characters can be letters, numeric digits, or underscores. Blanks are not permitted.
- Mixed-case names are honored for presentation purposes. However, because any comparison of names is not case-sensitive, you cannot have two names that differ only in case (for example, State and STATE are read as the same name).

COVERAGE Statement Optional Arguments

When you specify CREATE, REPLACE, or UPDATE for the *operation* in a COVERAGE statement, you can specify one or more of these options following the coverage entry name.

- DESCRIPTION='*string*'
- SPATIAL=*spatial-entry*
- WHERE=('*where-string-1* <... '*where-string-n*')

Note: Separate the list of arguments from the *coverage-entry* name with a slash (/). △

The following list contains descriptions of additional COVERAGE statement options:

DESCRIPTION='*string*'
Specifies a descriptive phrase, up to 256 characters long, that is stored in the description field of the GISCOVER entry. The default description is blank.

SPATIAL=*spatial-entry*
Specifies the GISSPA-type entry to which the coverage definition refers. The default is the current spatial entry.

An error occurs if there is no existing spatial entry that has the specified name, or if you omit this argument when no spatial entry is currently selected.

WHERE=<(>'*where-string-1*' <... '*where-string-n*'><)>
Specifies a WHERE clause that subsets the chains data set to define a geographic coverage of a spatial database. *Where-string* can contain a complete valid WHERE expression of 200 characters or fewer.

To specify a WHERE expression greater than 200 characters, you must break the expression into separate quoted strings. When WHERE= is processed, the

strings are concatenated, with a space between each string, and the entire expression is evaluated.

If you are using multiple strings, each string does not have to contain a complete WHERE expression, but the concatenated expression must be valid.

You can use any of the variables in the chains data set in the WHERE expression, not just the coordinate variables. However, the WHERE clause must delineate a bounded geographic region. You can use only the variables in the WHERE expression, not composites. Specify WHERE='1' to define a coverage that includes the entire spatial database.

Note: The WHERE= argument is required when you use the CREATE or REPLACE keyword. △

COVERAGE Statement Examples

Define a universal coverage

The following code fragment creates a coverage entry that is named GMAPS.USA.ALL.GISCOVER. The code defines a coverage of the entire spatial database that is defined in GMAPS.USA.USA.GISSPA:

```
proc gis cat=gmaps.usa;
   spatial usa;
   coverage create all / where='1';
run;
```

Define a coverage subset

Assume that the chains data set for the current spatial entry has the variables STATEL and STATER that contain FIPS state codes for the states on the left and right side of each chain. The following code fragment creates a coverage entry that is named SOUTHEAST of type GISCOVER. The code defines a coverage of only the selected states from the current spatial entry:

```
coverage create southeast /
   where=("statel in (1,12,13,28,37,45,47) |
           stater in (1,12,13,28,37,45,47)");
run;
```

LAYER Statement

LAYER *operation* <*libref.catalog.*>*layer-entry* </ *options*>;

Description

The LAYER statement

- Displays information about the contents of a layer entry
- Creates a new layer entry, replaces an existing layer entry, or modifies the characteristics of an existing layer entry
- Deletes a layer entry.

A layer entry is a SAS catalog entry of type GISLAYER that stores information about a layer in a map. Each layer represents a different set of features on the map and defines

how the features are displayed. For example, you could create a layer entry named RIVERS to represent the water features in your spatial data.

Layers can be displayed as either static or thematic. When a layer is displayed as static, it has a fixed set of graphical attributes (fill colors, outline colors, and so forth) for all of the features in that layer. When a layer is displayed as thematic, it uses values of a response variable in an associated attribute data set to determine the graphical attributes for the layer. Information about the theme value ranges and the attribute data is stored in the layer entry.

LAYER Statement Operations

In a LAYER statement, *operation* can be one of the following:

- CONTENTS
- CREATE
- DELETE
- REPLACE
- UPDATE.

The following list contains descriptions of the LAYER statement operations:

CONTENTS
: Displays the characteristics of the specified layer entry in the OUTPUT window, including the WHERE clause that defines the layer and lists of the layer's parameters and graphical attributes.

  An error occurs if the specified layer entry does not exist.

CREATE
: Creates a new layer entry to define a particular set of features in the spatial database.

  An error occurs if a layer entry with the specified name already exists. The LAYER CREATE statement does not overwrite existing layer entries. Use LAYER REPLACE to replace an existing entry.

  For a LAYER CREATE statement, you must also specify either the COMPOSITE= argument or the WHERE= argument. (For area layers, you must use the COMPOSITE= argument.)

DELETE
: Removes the specified layer entry.

  No additional arguments (other than the layer entry name) are used with this operation. An error occurs if the specified layer entry does not exist.

  For the DELETE operation, you can also specify the special value **_ALL_** for the *layer-entry* name to delete all layer entries in the current catalog.

  Note: You must specify a new layer list for any map entries that refer to the deleted layer entry. △

  CAUTION:

  Use DELETE with care. The GIS procedure does not prompt you to verify the request before deleting the layer entry. Be especially careful when you use **_ALL_**. △

REPLACE
: Overwrites the specified layer entry or creates a new layer entry if an entry with the specified name does not exist. The LAYER REPLACE statement has the effect

of canceling the previously issued LAYER CREATE statement for the specified layer entry.

For a LAYER REPLACE statement, you must also specify either the COMPOSITE= argument or the WHERE= argument. (For area layers, you must use the COMPOSITE= argument.)

UPDATE
Modifies the specified layer entry by applying new values for specified arguments.

An error occurs if there is no existing layer entry with the specified name.

Layer-Entry Name

In a LAYER statement, the *layer-entry* name identifies the layer entry that you want to create, delete, replace, or update. The general form of the argument is

<libref.catalog.>layer-entry

The *layer-name* must conform to the rules for SAS names:

- The name can be no more than 32 characters long.
- The first character must be a letter or underscore (_). Subsequent characters can be letters, numeric digits, or underscores. Blanks are not permitted.
- Mixed-case names are honored for presentation purposes. However, because any comparison of names is not case-sensitive, you cannot have two names that differ only in case (for example, State and STATE are read as the same name).

LAYER Statement Optional Arguments

When you specify CONTENTS, CREATE, REPLACE, or UPDATE for *operation* in a LAYER statement, you can specify one or more of the after additional arguments after the layer entry name.

- COMPOSITE=*composite-name*
- DESCRIPTION='*string*'
- DETAILON=*scale-value*
- DETAILS | NODETAILS
- MAP=*<libref.catalog.>map-entry*
- LABELON=*scale-value*
- OFFSCALE=*scale-value*
- ONSCALE=*scale-value*
- STATIC | THEMATIC
- TYPE=POINT | LINE | AREA
- UNITS=*unit-specification*
- WHERE=('*where-string-1* <...'*where-string-n*'>)
- FORCE
- DEFAULT=(*static-arguments*)
- THEME=*(theme-arguments)*

Note: Separate the list of arguments from the *layer-entry* name with a slash (/). △

The following list contains descriptions of additional LAYER statement arguments:

COMPOSITE=*composite-name*
Specifies a composite that defines the common characteristic of the features in the layer. The COMPOSITE= argument is an alternative to specifying a WHERE clause by using the WHERE= argument. For example, if you use specify COMPOSITE=STATE in the LAYER statement and the composite named STATE was created with the variable association VAR=(LEFT=STATEL,RIGHT=STATER), then the implied WHERE clause for the layer is WHERE STATEL NE STATER.

Note: Either the COMPOSITE= argument or the WHERE= argument is required when you use the CREATE or REPLACE keyword. For area layers, you must use the COMPOSITE= argument. △

DESCRIPTION='*string*'
Specifies a descriptive phrase, up to 256 characters long, that is stored in the description field of the layer entry. The default description is blank.

DETAILON=*scale-value*
Specifies the scale at or below which detail coordinates are displayed, provided that detail points are available. This argument helps keep the detail level of a layer to a minimum when the map is zoomed to a large scale. By default, detail is displayed at all scales when detail is turned on.

Note: The DETAILON= argument is effective only when detail coordinates are read for the layer. The DETAILS argument controls whether detail coordinates are read. △

DETAILS | NODETAILS
Specifies whether the detail coordinates are read for this layer. The default is NODETAILS.

If you specify DETAILS to read the detail coordinates from the database, you can use the DETAILON= argument to control the scale at which the detail coordinates are actually displayed.

MAP=<*libref.catalog.*>*map-entry*
Identifies a GISMAP-type entry that provides theme information for layers that are created in SAS/GIS in Release 6.11 of the SAS System. This option is ignored for layers that are generated by later releases of SAS/GIS. For thematic layers, the link to the associated data set and the name of the response variable for the theme are stored in the map entry rather than in the layer entry. If you omit this argument, the LAYER CONTENTS statement is unable to provide thematic display information for layers that were created in SAS/GIS in Release 6.11 of the SAS System.

Note: The MAP= argument is valid only in conjunction with the CONTENTS and UPDATE operations and is the only option that is permitted with the CONTENTS operation. △

LABELON=*scale-value*
Specifies the numeric scale at or below which map labels are displayed. This argument helps keep the number of items in the map window to a minimum when the map is zoomed to a large scale. By default, labels are displayed at all scales.

OFFSCALE=*scale-value*
Specifies the numeric scale at or below which the layer display is turned off. When you zoom in below the specified scale, the layer is hidden. When you zoom out beyond the specified scale, the layer is displayed. By default, the layer is displayed at all zoom scales.

Note: If you specify both the OFFSCALE= and ONSCALE= arguments, the *scale-value* for OFFSCALE= must be less than the *scale-value* for ONSCALE=. △

ONSCALE=*scale-value*
Specifies the numeric scale at or below which the layer display is turned on. When you zoom in below the specified scale, the layer is displayed. When you zoom out beyond the specified scale, the layer is hidden. This argument helps prevent clutter by enabling you to suppress selected layers when the map is zoomed to a large scale. By default, the layer is displayed at all zoom scales.

Note: If you specify both the ONSCALE= and OFFSCALE= arguments, the *scale-value* for ONSCALE= must be greater than the *scale-value* for OFFSCALE=. △

STATIC
Toggles the current theme off so that it is not displayed when the map is opened. It does not remove the theme from the layer entry. If the layer has no theme, STATIC is ignored. The default appearance of a newly created layer is STATIC. Static graphical attributes can be modified with the DEFAULT=(...) on page 108 option as described later in this section.

THEMATIC
Toggles the current theme in the layer on so that it is displayed when the map is opened. If the layer has no theme, this option has no effect. Use the THEME=(...) option to create a theme in a layer. See THEME= later in this section for more information.

TYPE=POINT | LINE | AREA
Specifies the type of layer. The TYPE argument affects how the layer is displayed in a map. The default is TYPE=LINE.

POINT
The layer's features are discrete points and have no length or area associated with them. If a POINT feature has left and right attributes, the values of the attributes must be identical.

LINE
The layer's features have length, and they can have different values for their left and right attributes. However, a LINE feature can enclose an area, even though it is displayed as a line.

AREA
The layer's features have length and area associations and the layer is displayed as enclosed polygons.

Note: Each area layer must have a polygonal index for the composite that defines the area boundaries. △

UNITS=*unit-specification*
Specifies the scale units for subsequent ONSCALE=, OFFSCALE=, and DETAILON= argument values. The default is UNITS=METRIC (for example, kilometers per centimeter).
Unit-specification can be one of the following:

ENGLISH
Selects nonmetric as the scale units, for example, miles per inch or feet per inch.

METRIC
Selects metric as the scale units, for example, kilometers per centimeter or meters per centimeter.

real-units/*map-units*
Selects a user-defined combination of units. Valid values for *real-units* and *map-units* are as follows:

- KM | KILOMETER | KILOMETERS
- M | METER | METERS
- CM | CENTIMETER | CENTIMETERS
- MI | MILE | MILES
- FT | FOOT | FEET
- IN | INCH | INCHES

The value of *real-units* is typically KM, M, MI, or FT, and the value of *map-units* is usually either CM or IN.

WHERE=('*where-string-1*' <... '*where-string-n*'>)
Specifies a WHERE clause that subsets the chains data set to define a geographic layer of a spatial database. *Where-string* can contain a complete valid WHERE expression of 200 characters or fewer.

To specify a WHERE expression greater than 200 characters, you must break the expression into separate quoted strings. When WHERE= is processed, the strings are concatenated, with a space between each string, and the entire expression is evaluated.

If you are using multiple strings, each string does not have to contain a complete WHERE expression, but the concatenated expression must be valid.

You can use any of the variables in the chains data set in the WHERE expressions, not just the coordinate variables. However, the layer definition must not delineate a bounded geographic region, but rather a particular subset of the spatial data that is independent of the coverage. For example, a STREETS layer may apply to all the spatial data, even if streets do not exist in many areas. You can use only the variables in the WHERE expression, not composites. Specify WHERE='1' to define a layer that contains all the features in the map.

Note: Either the WHERE= argument or the COMPOSITE= argument is required when you use the CREATE or REPLACE operation. For area layers, you must use the COMPOSITE= argument. If you use the WHERE= argument, the default layer type is LINE. △

FORCE
Allows you to create more than one theme by using the same variable from the same attribute data set.

DEFAULT=
Is used to define the static appearance of a layer. Within the DEFAULT= argument, you indicate the appropriate appearance options for the layer.

DEFAULT= *options*:

POINT=
Defines the static appearance of the symbols in a point layer. This option allows you to specify the color, size, font, and a specific character to be used for the symbols. It is valid only when TYPE=POINT is in the layer definition.

POINT= syntax:

```
DEFAULT = (
   POINT = (COLOR = color
            SIZE = [1..21]
            FONT=font-name
            CHARACTER='character' ));
```

POINT= arguments:

COLOR= Specifies the color of the point symbol. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX*rrggbb*, an HLS color code in the form H*hhhllss*, or a gray-scale color code in the form GRAY*ll*. For more information about color-naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. The default color is BLACK.

SIZE= Specifies the size of the point symbol. SIZE= must specify an integer that is greater than or equal to 1, and less than or equal to 21. The default size is 8.

FONT= Specifies the font to use for the point symbol. FONT= must specify a valid fontname. The default font is MARKER. Font verification can be overridden by using the FORCE option on the LAYER statement.

CHARACTER= Specifies the character to use for the point symbol. CHARACTER= must specify a single character in quotes. The default character is 'W', which in the MARKER font represents a "dot."

LINE=

Defines the static appearance of the lines in a line layer. Allows you to specify the color, width, and style to be used for the lines. Valid only when TYPE=LINE is in the layer definition.

LINE= syntax:

```
DEFAULT = (LINE = (COLOR = color
                   WIDTH = [1..20]
                   STYLE = SOLID | DASHED | DOTTED));
```

LINE= arguments:

COLOR= Specifies the color of the line. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX*rrggbb*, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color-naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. The default color is BLACK.

WIDTH= Specifies the width of the line. WIDTH= must specify an integer that is greater than or equal to 1, and less than or equal to 20. The default width is 1.

STYLE= Specifies the style of the line. STYLE= must specify either SOLID, DASHED, or DOTTED. The default style is SOLID.

CENTERLINE=

Defines the static appearance of the optional centerline in a line layer. The option allows you to specify whether a centerline is displayed as well as the color, width, and style to be used for the centerlines. It is valid only when TYPE=LINE in the layer definition.

CENTERLINE= syntax:

```
DEFAULT = (CENTERLINE = (ON | OFF
                         COLOR = color
```

```
                                        WIDTH = [1..20]
                                        STYLE = SOLID | DASHED | DOTTED));
```

CENTERLINE= arguments:

ON | OFF — Specifies whether the optional centerline is displayed. The default is OFF.

COLOR= — Specifies the color of the centerline. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX**rrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. The default color is BLACK.

WIDTH= — Specifies the width of the centerline. WIDTH= must specify an integer that is greater than or equal to 1 and less than or equal to 20. The default width is 1.

STYLE= — Specifies the style of the centerline. STYLE= must specify either SOLID, DASHED, or DOTTED. The default style is SOLID.

AREA=

Defines the static appearance of the area fills in an area layer. Area allows you to specify the color and fill style as well as angle and spacing parameters for hatched and crosshatched fill styles. It is valid only when TYPE=AREA in the layer definition.

AREA= syntax:

```
DEFAULT = (AREA = (COLOR = color
                   STYLE = EMPTY | FILLED | HATCH | CROSSHATCH
                   ANGLE = angle-value
                   SPACING = [2..10] ));
```

AREA= arguments:

COLOR= — Specifies the fill color of the area. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX**rrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color-naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. The default color is GRAY.

STYLE= — Specifies the fill style of the area. STYLE= must specify either EMPTY, FILLED, HATCH, or CROSSHATCH. The default style is FILLED, which means the area contains one, solid color.

ANGLE= — Specifies an angle for hatched and crosshatched lines. ANGLE= must specify a real number that is greater than or equal to zero and less than 90 (for crosshatch), or greater than or equal to zero and less than 180 (for hatch). The default angle for both the hatch and crosshatch is 0.

SPACING= — Specifies the spacing between hatched lines or crosshatched lines. SPACING= must specify an integer that is greater than or equal to 2 and less than or equal to

10. The lower the number, the less space between lines (the higher the number, the more space between lines). The default spacing is 7.

OUTLINE=

Defines the appearance of the area outlines in an area layer. The option allows you to specify the color, width, and style to be used for the outlines. It is valid only when TYPE=AREA is in the layer definition.

OUTLINE= syntax:

```
DEFAULT = (OUTLINE = (ON | OFF
                      COLOR = color
                      WIDTH = [1..20]
                      STYLE = SOLID | DASHED | DOTTED));
```

OUTLINE= arguments:

ON | OFF — Specifies whether the area outline is displayed. The default is ON.

COLOR= — Specifies the color of the outline. COLOR= must specify a predefined SAS color name, an RGB color code in the form C**xrrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color-naming schemes, see "SAS/GRAPH Colors in *SAS/GRAPH Software: Reference*. The default color is BLACK.

WIDTH= — Specifies the width of the outline. WIDTH= must specify an integer that is greater than or equal to 1 and less than or equal to 20. The default width is 1.

STYLE= — Specifies the style of the outline. STYLE= must specify either SOLID, DASHED, or DOTTED. The default style is SOLID.

Note: The following THEME= list entry is an optional argument of the LAYER statement. It is not an option of the DEFAULT= optional argument of the LAYER statement. △

THEME =

THEME=(*operations options*) allows you to modify or delete existing themes or to create new themes. In the LAYER statement THEME argument, the *operation* argument can be one of the following:

CREATE

REPLACE

UPDATE

DELETE

The following list contains descriptions of the THEME= operations:

CREATE

Creates a new theme for the specified layer entry.

An error occurs if a theme already exists for the layer that uses the same variable in the same attribute data set, unless you also specify the FORCE option in the LAYER statement. The CREATE statement does not overwrite existing themes. Use REPLACE to replace an existing theme.

For a CREATE statement, you must also specify the LINK= and VAR= arguments.

REPLACE
Overwrites the specified theme for the layer entry. The REPLACE statement has the effect of canceling the previously issued CREATE statement for the specified layer entry.

For a REPLACE statement, you must also specify both the LINK= argument and the VAR= arguments.

UPDATE
Modifies the specified theme for the layer entry by applying new values for specified arguments.

An error occurs if the specified layer does not have at least one existing theme. For an UPDATE statement, you must specify a value for at least one of the arguments LINK=, VAR=, RANGE=, NLEVELS=, MAKE_CURRENT, or NOT_CURRENT.

If you do not specify LINK=, the current data set link is used. If you do not specify THEMEVAR=, the current thematic variable is used.

DELETE
Removes the specified theme from the specified layer entry.

For a DELETE statement, you must specify a value for THEMEVAR=*varname* or POSITION=*integer*. An error occurs if you specify THEMEVAR=*varname* and if a theme based on *varname* does not exist.

CAUTION:
Use DELETE with care. The GIS procedure does not prompt you to verify the request before it deletes the layer theme. △

Additional LAYER Statement Optional Arguments

When you specify CONTENTS, CREATE, REPLACE, or UPDATE for the *operation* keyword in a LAYER statement and specify THEME=, you can specify the following additional options.

LINK=*link-name*

THEMEVAR=*var-name*

RANGE= DEFAULT | DISCRETE | LEVELS

NLEVELS=*integer*

POSITION=*integer*

MAKE_CURRENT | NOT_CURRENT

COMPOSITE=(*comp-name*)

DATASET=<*libref.*>*data-set*

DATAVAR=(*variable-name*

POINT=(*argument*)

LINE=(*argument*)

CENTERLINE=(*argument*)

AREA=(*argument*)

OUTLINE=(*argument*)

The following list contains descriptions of the THEME= options:

LINK=*link-name*
Specifies the attribute data set containing the theme variable to be used. If you do not specify *link-name* and you are performing an update, the current data set link is used.

THEMEVAR=*var-name*
Specifies the theme variable in the linked attribute data set (specified in LINK=*link-name*). If you do not specify *var-name* and you are performing an update, the current theme variable is used.

THEMEVAR=*var-name* also specifies the theme to delete or to make current.

RANGE=DEFAULT | DISCRETE | LEVELS
Specifies the thematic range type.

DEFAULT
Increments are calculated automatically using an algorithm that is based on the 1985 paper by G.R. Terrell and D. W. Scott, "Oversmoothed Nonparametric Density Estimates" in the *Journal of the American Statistical Association*, Volume 80, pages 209-214.

DISCRETE
The range is treated as a series of discrete values instead of a continuous variable. If the variable that is specified in the VAR= argument is a character variable, only RANGE=DISCRETE is allowed.

LEVELS
The range is divided into evenly spaced increments. You do not have to specify RANGE=LEVELS if you specify NLEVELS=*integer* instead.

If you do not specify RANGE=, DEFAULT is used for numeric variables and DISCRETE is used for character variables.

NLEVELS=*integer*
Specifies the number of range levels in the theme. The value for NLEVELS must be an integer greater than one. You cannot specify both NLEVELS and RANGE=DEFAULT or RANGE=DISCRETE. If you specify NLEVELS, RANGE=*LEVELS* is assumed and can be omitted.

POSITION=*integer*
Specifies the position number of the target theme. POSITION=1 is the first theme, POSITION=2 is the second theme, and so on. Negative values of *integer* are the position number counting backward from the last theme. POSITION=-1 specifies the last theme; POSITION=-2 specifies the second to the last theme, and so on. POSITION=0 specifies the current theme, whatever position that theme is in the series.

If you do not specify a value for POSITION, new themes are created at the end of the theme list, UPDATE operations are performed on the last theme, and DELETE fails unless you specify the target theme with THEMEVAR=*var-name*.

If you use POSITION to specify a new theme position, *layer-name* must not already contain a theme at that position.

MAKE_CURRENT
Specifies that the specified theme is to be the current theme when the map opens. MAKE_CURRENT is the default when a theme is created or updated.

NOT_CURRENT
Specifies that the specified theme should be created or modified but is not to be made the current theme.

COMPOSITE=(*comp-name-1* | <,..., *comp-name-n*>)
Lists one or more spatial composite names when you create a new key or link for a theme. If only one composite is listed, you can omit the parentheses. The composites are paired with the attribute data set variables that are named in the DATAVAR= argument. If the composite names and the data set variable names are the same, you can justify them once with either the COMPOSITE= or DATAVAR= lists, and those names will be used for both.

Note: This is not the same argument as the COMPOSITE = argument that is used to set up a WHERE clause when you create an AREA type layer. △

DATASET= <*libref.*>*data-set*
Specifies the attribute data set when you create a new key link for a theme. If you specify a one-level data set name, the default library is WORK.

DATAVAR= (*variable-1* <,..., *variable-n*>)
Lists attribute data set variables when you create a new key link for a theme. If only one variable is listed, you can omit the parentheses. These variables are paired with the spatial composites that are named in the COMPOSITE= argument. If the data set variable names and the composite names are the same, you can justify them once with either the COMPOSITE= or DATAVAR= lists, and those names will be used for both.

POINT=
Defines the appearance of the symbol for each level of the specified theme in a point layer. The option allows you to specify the color, size, font and specific character to be used for the symbols. It is valid only when TYPE=POINT is specified in the layer definition.

POINT= syntax:

```
THEME = (POINT = ((LEVEL = integer | FIRST | LAST
                   COLOR = color | CURRENT
                   SIZE = [1..21] | CURRENT
                   FONT = font1
                   CHARACTER='char')
                  BLENDCOLOR
                  BLENDSIZE));
```

POINT= arguments:

LEVEL= Specifies which theme range is being operated upon. For example, LEVEL=1 refers to the first range level in this theme. LEVEL=FIRST and LEVEL=LAST can also be used to denote the initial and final range levels. If LEVEL=1 is omitted, the entered theme parameters are assigned to the range levels in sequence.

COLOR= Specifies the color of the point symbol. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX**rrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information on color naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. CURRENT is specified when you want to specify BLENDCOLOR and you simply want this range level to maintain its current color.

SIZE= Specifies the size of the point symbol. SIZE= must specify an integer that is greater than or equal to 1 and less than or equal to 21. Defaults to the size of the static point symbol for this layer. CURRENT is specified when you want to specify BLENDSIZE and use this existing range level size as one of those points to interpolate between.

FONT= Specifies the font to use for the point symbol. FONT= must specify a valid fontname. The default is the font of the static point symbol for this layer. Font verification can be overridden by using the FORCE option in the LAYER statement.

CHARACTER= Specifies the character to use for the point symbol. CHARACTER= must specify a single character in quotes. The defaults is the character of the static point symbol for this layer.

BLENDCOLOR Interpolates the color values for any theme range levels between those that you specified with LEVEL=. If you want to blend between existing colors, indicate the colors with COLOR=CURRENT.

BLENDSIZE Interpolates the point size for any theme range levels between those that you specified with LEVEL=. To blend between existing sizes, indicate the sizes as SIZE=CURRENT.

LINE=

Defines the appearance of the line for each level of the specified theme in a line layer. The option allows you to specify the color, width and style to be used for the lines. It is valid only when TYPE=LINE in the layer definition.

LINE= syntax:

```
THEME = ( LINE= ((LEVEL = integer | FIRST | LAST
                  COLOR = color | CURRENT
                  WIDTH = [1..20]
                  STYLE = SOLID | DASHED | DOTTED)
                 BLENDCOLOR
                 BLENDWIDTH));
```

LINE= arguments:

LEVEL= Specifies which level of the theme is being operated upon. For example, LEVEL=1 refers to the first range level in this theme. LEVEL=FIRST and LEVEL=LAST can also be used to denote the initial and final range levels. If the LEVEL= arguments are omitted, the entered theme parameters are assigned to the range levels in sequence.

COLOR= Specifies the color of the line. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX**rrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color-naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. CURRENT is used when you want to BLENDCOLORS and use this range level color as one of the colors to interpolate with.

WIDTH= Specifies the width of the line. WIDTH= must specify an integer that is greater than or equal to 1 and less than or equal to 20. The default is the width of the static line for this layer. CURRENT is used when you want to specify BLENDWIDTH and use this existing range level width as one of those to interpolate with.

STYLE= Specifies the style of the line. STYLE= must specify either SOLID, DASHED, or DOTTED. The default is the style of the static line for this layer.

BLENDCOLOR Interpolates the color values for any theme range levels between those specified with LEVEL=. If you want to blend between existing colors, indicate the colors with COLOR=CURRENT.

BLENDWIDTH Interpolates the line width for any theme range levels between those specified with LEVEL=. To blend between existing widths, indicate the widths as WIDTH=CURRENT.

CENTERLINE=
Defines the appearance of the optional centerline for the specified theme in a line layer. The option allows you to specify whether a centerline is displayed as well as the color, width, and style to be used for the centerlines. It is valid only when TYPE=LINE is in the layer definition.

Note: A centerline does not vary in a single theme. Its appearance is the same for all range levels. △

CENTERLINE= syntax:

```
THEME = (CENTERLINE = (ON | OFF
                       COLOR = color | CURRENT
                       WIDTH = [1..20]
                       STYLE = SOLID | DASHED | DOTTED));
```

CENTERLINE= arguments:

ON | OFF Specifies whether the optional centerline is displayed. The default is the same on/off status as the static centerline for this layer.

COLOR= Specifies the color of the centerline. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX**rrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color-naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. The default is the color of the static centerline for this layer.

WIDTH= Specifies the width of the centerline. WIDTH= must specify an integer that is greater than or equal to 1 and less than or equal to 20. The default is the width of the static centerline for this layer.

STYLE= Specifies the style of the centerline. STYLE= must specify either SOLID, DASHED, or DOTTED. The defaults is the style of the static centerline for this layer.

AREA=
Defines the appearance of the area fill for each level of the specified theme in an area layer. AREA= allows you to specify the color and fill style as well as angle and spacing parameters for hatched and crosshatched fill styles. It is valid only when TYPE=AREA in the layer definition.

AREA= syntax:

```
THEME = (AREA = ((LEVEL = integer | FIRST | LAST
                  COLOR = color | CURRENT
                  STYLE = EMPTY | FILLED | HATCH | CROSSHATCH
                  ANGLE = angle-value
                  SPACING = [2..10]) | CURRENT
                 BLENDCOLOR
                 BLENDSPACING));
```

AREA= arguments:

LEVEL= Specifies which level of the theme is being modified. For example, LEVEL=1 refers to the first range level in this theme.

LEVEL=FIRST and LEVEL=LAST can also be used to denote the initial and final range levels. If the LEVEL= arguments are omitted, the entered theme parameters are assigned to the range levels in sequence.

COLOR= Specifies the fill color of the area. COLOR= must specify a predefined SAS color name, an RGB color code in the form CX**rrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color-naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. The defaults is GRAY. CURRENT is specified when you want to specify BLENDCOLORS and use this range level color as one of the colors to interpolate with.

STYLE= Specifies the fill style of the area. STYLE= must specify either EMPTY, FILLED, HATCH, or CROSSHATCH. The defaults is the style of the static area for this layer.

ANGLE= Specifies an angle for hatched and crosshatched lines. ANGLE= must specify a real number that is greater than or equal to zero and less than 90 (for crosshatch), or greater than or equal to 0 and less than 180 (for hatch). The default is the angle of the static area for this layer.

SPACING= Specifies the spacing between hatched lines or crosshatched lines. SPACING= must specify an integer that is greater than or equal to 2 and less than or equal to 10. The lower the number, the less space that there is between lines (the higher the number, the more space that there is between lines). The defaults is the spacing of the static area for this layer.

BLENDCOLOR Interpolates the color values for any theme range levels between those specified with LEVEL=. If you want to blend between existing colors, indicate the colors with COLOR=CURRENT.

BLENDSPACING Interpolates the hatched or crosshatched style for any theme range levels between those specified with LEVEL=. To blend between existing spacing values, indicate them as SPACING=CURRENT. If any intermediate range levels are not hatched or crosshatched, BLENDSPACING ignores them.

OUTLINE=

Defines the appearance of the polygon outlines for each level of the specified theme in an area layer. OUTLINE= allows you to specify the color, width, and style to be used for the outlines. It is valid only when TYPE=AREA in the layer definition.

OUTLINE= syntax:

```
THEME = (OUTLINE = (ON | OFF
                    COLOR = color
                    WIDTH = [1..20]
                    STYLE = SOLID | DASHED | DOTTED));
```

OUTLINE= arguments:

ON | OFF Specifies whether the area outline is displayed. The defaults is the same on/off status as the static outline for this layer.

COLOR= Specifies the color of the outline. COLOR= must specify a predefined SAS color name, an RGB color code in the form

	CX**rrggbb**, an HLS color code in the form H**hhhllss**, or a gray-scale color code in the form GRAY**ll**. For more information about color-naming schemes, see "SAS/GRAPH Colors" in *SAS/GRAPH Software: Reference*. The defaults is the color of the static outline for this layer.
WIDTH=	Specifies the width of the outline. WIDTH= must specify an integer that is greater than or equal to 1 and less than or equal to 20. The defaults is the width of the static outline for this layer.
STYLE=	Specifies the style of the outline. STYLE= must specify either SOLID, DASHED, or DOTTED. The defaults is the style of the static outline for this layer.

LAYER Statement Examples

Define a layer using a composite

If the chains data set contains pairs of variables that indicate values for the areas on the left and right sides of the chains, then you can use these variable pairs to define area layers. The following code fragment defines a composite that identifies county boundaries (chains for which the area values on the left and right sides are unequal) and uses that composite to define an area layer:

```
composite create county / var=(left=countyl,right=countyr)
                          class=area;
run;
polygonal index create county / composite=county
                                out=gmaps.cntyx;
run;
layer create county / composite=county
                      type=area;
run;
```

Note: The polygonal index must be defined for the composite in order to display this area layer in a map. △

Define a layer using a category variable

Assume that the spatial database contains a variable named CFCC that has values that identify what each chain represents. Assume also that the values of the CFCC variable for all roads begin with the letter A (A0, A1, and so forth, depending on the category of road). The following code fragment defines a line layer that consists of all features that are roads:

```
layer create roads / where='cfcc =: "A"'
                     type=line;
run;
```

Note: The colon (:) modifier to the equals operator restricts the comparison to only the first *n* characters of the variable value, where *n* is the number of characters in the comparison string. In the preceding example, the WHERE clause tests for "where the value of CFCC begins with A." △

Create a theme

This example creates a new theme for the SASUSER.MALL.STORES map, supplied with the SAS/GIS tutorial. The theme uses the *sqft* variable in the *mallstor* attribute data set to define the theme.

```
proc gis;
   spatial sasuser.mall.mall;
   layer update sasuser.mall.store / theme = (create
                                               themevar = sqft
                                               dataset = sasuser.mallstor
                                               datavar = store
                                               composite = store
                                               link = mallstor
                                               range = discrete
                                               pos = -1
                                               not_current );
   run;
quit;
```

Update an Existing Theme

Using the *sqft* theme that was created in the previous example and modify it as follows:

- Change the theme variable to *rent* from the same attribute data set.
- Break the *rent* values into nine theme range levels.
- Make the first level blue.
- Make the last level cxff0000 (red).
- Blend the colors for the intermediate range levels.

```
proc gis c=sasuser.mall;
   spatial mall;
   layer update store / theme=(update
                               pos = 1
                               themevar = rent
                               range = levels
                               nlevels = 9
                               area = ((level = first color = blue    )
                                       (level = last  color = cxff0000)
                                       blendcolor));
   run;
quit;
```

MAP Statement

MAP *operation* <*libref.catalog.*>*map-entry* </ *options*>;

Description

The MAP statement

- Displays information about the contents of a map entry

- Creates a new map entry, replaces an existing map entry, or modifies the characteristics of a previously created map entry
- Deletes a map entry.

A map entry is a SAS catalog entry of type GISMAP that defines the displayed features of a map. The definition specifies which layers the map contains and which coverage of the spatial database is used. The map entry also stores legend definitions and action definitions for the map, information about the projection system used to display the map, and the names of any associated SAS data sets and of the data set that contains labels for map features.

MAP Statement Operations

In the MAP statement, *operation* can be one of the following:

- CONTENTS
- CREATE
- DELETE
- REPLACE
- UPDATE

The following list contains descriptions of the MAP statement operations:

CONTENTS
Prints information about the specified map entry to the OUTPUT window, including

- A list of the data objects (coverage and layer entries and label data set) that compose the map entry
- Details of the spatial database as provided by the COVERAGE CONTENTS and SPATIAL CONTENTS statements
- Details of the layer definitions as provided by the LAYER CONTENTS statement
- Lists of the projection method that is used to display the map
- A list of associated data sets and link names
- A list of the GIS actions that have been defined for the map
- A list of legend definitions for the map.

No additional arguments (other than the *map-entry* name) are used with this operation. An error occurs if the specified map entry does not exist.

CREATE
Creates a new map entry that defines a map that can be displayed in the GIS Map window.

An error occurs if a map entry with the specified name already exists. The MAP CREATE statement does not overwrite existing map entries. Use MAP REPLACE to overwrite an existing entry.

For a MAP CREATE statement, you must also specify the COVERAGE= and LAYERS= arguments.

DELETE
Removes the specified map entry.

No additional arguments (other than the map entry name) are used with this operation. An error occurs if the specified map entry does not exist.

For the DELETE operation, you can also specify the special value **_ALL_** for the map entry name argument to delete all map entries in the current catalog.

CAUTION:
Use DELETE with care. The GIS procedure does not prompt you to verify the request before deleting the map entry. Be especially careful when you use **_ALL_**. △

REPLACE
Overwrites the specified map entry. The MAP REPLACE statement has the effect of canceling the previously issued MAP CREATE statement for the specified map entry.

For a MAP REPLACE statement, you must also specify the COVERAGE= and LAYERS= options.

UPDATE
Modifies the specified map entry by applying new values for specified arguments.

An error occurs if there is no existing map entry with the specified name.

Map-Entry Name

In the MAP statement, the map entry identifies the map entry you want to create, delete, replace, or update. The general form of the argument is

<libref.catalog.>map-entry

If you specify a one-level name, the map entry is assumed to be in the catalog that is specified in the PROC GIS statement or in the most recently issued CATALOG statement. An error occurs if no catalog has previously been specified.

The *map-entry* name must conform to the rules for SAS names:

- The name can be no more than 32 characters long.
- The first character must be a letter or underscore (_). Subsequent characters can be letters, numeric digits, or underscores. Blanks are not permitted.
- Mixed-case names are honored for presentation purposes. However, because any comparison of names is not case-sensitive, you cannot have two names that differ only in case (for example, State and STATE are read as the same name).

MAP Statement Optional Arguments

When you specify CREATE, REPLACE, or UPDATE for the MAP operation, you can specify one or more of the following options following the map-entry name.

Note: Separate the list of options from the map *entry-name* with a slash (/). △

- CARTESIAN | LATLON
- COVERAGE=*<libref.catalog.>coverage-entry*
- DEGREES | RADIANS | SECONDS
- DESCRIPTION='*string*'
- DETAILS | NODETAILS
- LAYERS=(*layer-entries*)
- LAYERS+=(*layer-entries*)
- LAYERS-=(*layer-entries*)
- LAYERSON=(*layer-entries*) | _ALL_
- LAYERSON+=(*layer-entries*)
- LAYERSON-=(*layer-entries*)

- LAYERSOFF=(*layer-entries*) | _ALL_
- LAYERSOFF+=(*layer-entries*)
- LAYERSOFF-=(*layer-entries*)
- MULT=*multiplier-value*
- LABEL= *libref.data-set* | NONE
- ACTION=(*operation-arguments*)
- ATTRIBUTE=(*attribute-arguments*)
- RENAME_LAYER *old-name = new-name*
- LEGEND=HIDEALL | UNHIDEALL | REMOVALL
- FORCE
- NOWARN

The following list contains descriptions of additional MAP statement options:

CARTESIAN | LATLON
Specifies the coordinate system used for the displayed spatial data. The default is LATLON.

CARTESIAN
Data are in an arbitrary rectangular (plane) coordinate system

LATLON
Data are in a geographic (spherical) coordinate system.

Note: The map entry must use the same coordinate system as the spatial entry from which the map is derived. If the spatial entry specifies the CARTESIAN coordinate system, then you must also specify the CARTESIAN argument for the MAP statement. If the spatial entry specifies the LATLON coordinate system, then you must also specify the LATLON argument for the MAP statement. △

COVERAGE=<*libref.catalog.*>*coverage-entry*
Specifies the coverage entry to which the map refers. The coverage determines the geographic extent of the map.

Note: The COVERAGE= argument is required when you use the CREATE or REPLACE keyword. △

DEGREES | RADIANS | SECONDS
Specifies the coordinate units for the displayed spatial data when the coordinate system is geographic (LATLON). The default is RADIANS.

The unit system that you select defines the allowable range for coordinate values. For example, if you specify DEGREES, then all X coordinate values must be in the range -180 to 180, and all Y coordinate values must be in the range -90 to 90.

DESCRIPTION='*string*'
Specifies a descriptive phrase up to 256 characters long that is stored in the description field of the GISMAP entry. The default description is blank.

DETAILS | NODETAILS
Specifies whether detail coordinates are read for the entire map. The default is NODETAILS.

Note: You can use the LAYER statement's DETAILS and DETAILON= options to control the display of detail coordinates for a particular layer. The MAP statement's DETAILS option overrides the LAYER statement's DETAILS option. △

LAYERS=(<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Specifies a list of GISLAYER-type entry names. The specified layers form the complete list of layers in the map entry. If the map entry already contains a list of layers, they are replaced by these layers.

Note: The LAYERS= argument is required when you use the CREATE or REPLACE keyword. △

LAYERS+= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Specifies a list of GISLAYER-type entry names. These layers are added to the map's current layer list.

LAYERS-= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Specifies a list of GISLAYER-type entry names. These layers are removed from the map's current layer list. However, the layer entries are not deleted. They remain in their respective catalogs.

LAYERSON= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Specifies a list of GISLAYER-type catalog entries that will be turned on for this map. All other layers will be turned off. Any on-scale/off-scale settings are deactivated. Specifying LAYERSON=(_ALL_) turns all layers on.

LAYERSON+= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Adds the specified layer(s) to the LAYERSON list and deactivates any on-scale/off-scale settings for the specified layer(s).

LAYERSON-= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Removes the specified layer(s) from the LAYERS ON list and deactivates any on-scale/off-scale settings for the specified layer(s).

LAYERSOFF= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Specifies a layer (or list of layers) to be turned off for this map. All other layers are turned on. Any on-scale/off-scale settings are deactivated. Specifying LAYERSOFF=(_ALL_) turns all layers off.

LAYERSOFF+= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Adds the specified layer(s) to the "LAYERS OFF" list and deactivate any on-scale/off-scale settings for the specified layer(s).

LAYERSOFF-= (<*libref.catalog.*>*layer-entry-1* <, ..., <*libref.catalog.*>*layer-entry-n*>)
Removes the specified layer(s) from the "LAYERS OFF" list and deactivates any on-scale/off-scale settings for the specified layer(s).

Note: The following information applies to the LAYERSON and LAYERSOFF options:

- If a layer in any of the lists does not exist in the map, a warning is issued and that layer is ignored. (A missing layer does not end the current RUN-group processing.) Each layer is evaluated individually, so if other layers are valid they are toggled appropriately.
- If a layer is in both the LAYERSON list and the LAYERSOFF list, this condition generates a warning and ends that RUN-group.
- If one of the LAYERS options is specified in addition to LAYERSON or LAYERSOFF, the LAYERS parameter are processed first. Therefore, if a layer is removed from the map by using the LAYERS parameter, it cannot be referenced in a LAYERSON/LAYERSOFF parameter in that same statement. This action generates a warning, but the RUN-group processing does not stop.
- If both LAYERSON and LAYERSOFF are used in the same statement, both parameters must specify -= and/or +=. Specifying both LAYERSON=(...) and

LAYERSOFF=(...) in the same statement causes a conflict, and therefore is not allowed.

- The _ALL_ option cannot be mixed with layer names, that is, _ALL_ must appear by itself.
- _ALL_ cannot be used with either the += or the -= operators.

△

MULT= *multiplier-value*
Specifies a constant integer value by which spatial data coordinates are multiplied when the data are displayed. The default is MULT=1E7. If the unit multiplier is too large, it is recomputed when the map is opened, and a note is printed to the SAS log showing the new value. If your map opens and appears to be empty, your MULT value may be too small.

LABEL= <*libref.data-set*> | NONE assigns or removes a label data set reference.
LABEL= <*libref.data-set*>
Assigns the specified label data set to the map. If the map already has a label data set, the original is deassigned. However, it is not overwritten. LABEL=NONE deassigns the current label data set from the map entry, but the data set is not deleted.

ACTION= *(arguments)*
Lets you copy, delete, or update GIS actions that are associated with a map entry. The arguments that are used with ACTION are

- COPY
- DELETE
- UPDATE
- NAME=*action-name* | _ALL_
- WHEN= OFF | IMMEDIATE | DEFER
- FROM=*map-entry*
- RENAME=*new-action-name*

The following list contains descriptions of the ACTION options:

COPY
Copies existing actions from one map entry to another. Specify the map entry that contains the actions to be copied with the FROM=*map-entry* argument. The actions are copied to the map that is specified in the MAP statement.

Specify the actions to be copied with the NAME=*link-name* argument. If you specify NAME=_ALL_ you copy all actions in the specified map. Existing actions in the map to be updated are not overwritten unless you specify the FORCE argument.

DELETE
Removes an existing action from the map entry. Specify the action to be deleted with the NAME=*action-name* argument. If you specify NAME=_ALL_, you delete all actions. Use the NOWARN argument in the MAP statement to suppress messages when an action is not found.

UPDATE
Modifies existing actions in the map that is being updated. Specify the action to be updated with the NAME=*action-name* argument. If you specify NAME=_ALL_ , you update all actions. NAME= is required for UPDATE.

If you specify a single action, you can use the RENAME=*new-action-name* argument to change the link name. You cannot use RENAME if you specify NAME=_ALL_.

You can also change the action's execution settings with the WHEN argument.

NAME=*action-name* | _ALL_
Specifies the action to be copied, deleted, or updated. *action-name* identifies a single action, while _ALL_ specifies all actions.

Note: You cannot specify NAME=_ALL_ if you are using ACTION UPDATE with the RENAME argument. △

WHEN= OFF | IMMEDIATE | DEFERRED
Used with UPDATE to change the execution setting of the specified action.

OFF
The action is not executed when a layer feature is selected.

IMMEDIATE
The action is executed as soon as a layer feature is selected.

DEFERRED
The action's execution must be performed explicitly after a layer feature has been selected.

FROM=*map-entry*
Used with the ACTION argument COPY operation, FROM= specifies the source map entry that contains actions to be copied. Specify the actions to be copied from the map with the NAME=*action-name* argument.

RENAME=*new-action-name*
Renames the link that is specified in the NAME=*action-name* for UPDATE.

Note: You cannot specify RENAME if you have also specified NAME=_ALL_. △

ATTRIBUTE
ATTRIBUTE lets you copy, delete, or update data links. The arguments used with ATTRIBUTE are

- CREATE
- COPY
- DELETE
- UPDATE
- NAME=*link-name* | _ALL_
- FROM=*map-entry*
- RENAME=*new-link-name*
- DATASET=*libref.data-set*
- COMPOSITE=(*composite-name-1* <,...,*composite-name-n*>)
- DATAVAR=(*var-name-1* <,...,*var-name-n*>)

The following list contains descriptions of the ATTRIBUTE arguments:

CREATE
Adds a new attribute data link to the map.

COPY
Copies existing attribute data links from one map entry to another. Specify the map entry that contains the links to be copied by using the FROM=*map-entry* argument. The links are copied to the map that is specified in the MAP statement.

Specify the link to be copied with the NAME=*link-name* option. If you specify NAME=_ALL_, you copy all links in the specified map. Existing links in the map to be updated are not overwritten unless you specify the FORCE option in the MAP statement.

DELETE
Removes an existing attribute data link from the map entry. Specify the link to be deleted with the NAME=*link-name* argument. If you specify NAME=_ALL_, you delete all data links. Use the NOWARN option in the MAP statement to suppress messages when a link is not found. This does not delete the attribute data set, only the link.

UPDATE
Modifies existing data links in the map that is being updated. Specify the link to be updated with the NAME=*link-name* argument. If you specify NAME=_ALL_, you update all data links. NAME= is required for the UPDATE operation.

If you specify a single link, you can use the RENAME=*new-link-name* argument to change the link name. You cannot use RENAME if you specify NAME=_ALL_.

NAME=*link-name* | _ALL_
Specifies the attribute data link to be copied, deleted, or updated. *link-name* identifies a single data link, while _ALL_ specifies all data links.

Note: You cannot specify NAME=_ALL_ if you are using UPDATE with the RENAME argument. △

FROM=*map-entry*
Used with the ATTRIBUTE COPY operation, FROM= specifies the map entry that contains data links to be copied. Specify the links to be copied from the map with the NAME=*link-name* argument.

RENAME=*new-link-name*
Renames the link that is specified in the NAME=*link-name* for the UPDATE operation.

Note: You cannot specify RENAME if you have also specified NAME=_ALL_. △

DATASET=*libref.data-set*
Specifies the attribute data set when you create a new key link.

COMPOSITE=(*composite-name-1 <,...composite-name-n>*)
Lists spatial composite names when you create a new key link. These composites are paired with the attribute data set variables that are named in the DATAVAR= option. If the composite names and the data set variable names are the same, you can just specify them once with either the COMPOSITE= or DATAVAR= lists, and those names will be used for both.

DATAVAR=(*varname-name-1 <,...var-name-n>*)
Lists attribute data set variables when you create a new key link. These variables are paired with the spatial composites that are named in the COMPOSITE= option. If the data set variable names and the composite names are the same, you can just specify them once with either the COMPOSITE= or DATAVAR= lists, and those names will be used for both.

RENAME_LAYER *old-name=new-name*
Changes the name of an existing layer in the map that is being updated. This argument also changes the name of the layer entry in the catalog.

If other maps use the renamed layer, you must issue a MAP UPDATE statement for those maps as well.

LEGEND=
Use the LEGEND option with the MAP UPDATE statement to hide, display, or remove map legends. The following list contains details on the options that you can use with the LEGEND option:

HIDEALL
Causes all existing legends to be hidden (not displayed) when the map is opened.

UNHIDEALL
Causes all existing legends to be displayed when the map is opened.

REMOVEALL
Removes all of the existing legends from the map. This behavior is immediate and permanent. You cannot restore the legends and will have to recreate them.

You can specify only one of the legend options at a time.

FORCE
Specifies that existing actions or attribute links may be overwritten during copy operations. Use this argument with the COPY argument in the ACTION or ATTRIBUTE argument.

NOWARN
Specifies that messages are not to be issued about actions or attribute links that are not found during deletion. Use this argument when you specify the DELETE argument in the ACTION or ATTRIBUTE argument.

MAP Statement Examples

Define a new map

The following code fragment creates an entry named STORES of type GISMAP in the current catalog. The map is based on the coverage defined in the GISCOVER entry named MALL in the current catalog and uses the GISLAYER entries STORE, FIRE, INFO, PHONE, and RESTROOM in the current catalog.

```
map create stores / coverage=mall
                    layers=(store, fire, info, phone, restroom);
run;
```

Update an existing map definition

The following code fragment updates the MAPS.USA.USA.GISMAP entry to use detail data when the map is displayed:

```
map update maps.usa.usa / details;
run;
```

Copy attribute data set links

The following code fragment copies the SIMPLUSR attribute link from GISSIO.SIMPLUS.SIMPLE to WORK.SIMPLE.SIMPLE:

```
proc gis;
   map update work.simple.simple /
       attribute = (name=simplusr
                    copy from=gissio.simplus.simple);
   run;
```

COPY Statement

COPY <*libref.catalog.*>*entry*<*.type*> </ *options*> ;

Description

COPY copies a SAS/GIS catalog entry or data set. You can copy a single GIS entry or include the dependent entries and data sets that are referenced by the source.

Note: If you use PROC COPY or another utility to copy a SAS/GIS catalog entry or data set, you may receive warnings in your SAS log that the paths are not the same. If you receive a message that the paths are not the same, you can use the SYNC statement to reset the paths. See "SYNC Statement" on page 131 for more information. △

COPY Statement Optional Arguments

- ENTRYTYPE=*type*
- DESTCAT=*libref.catalog*
- DESTLIB=*libref*
- BLANK
- REPLACE
- ALIAS=(*old-lib-1=new-lib-1* <, ... ,*old-lib-n=new-lib-n*>)
- SELECT= _ALL_ | ENTRY | DATASETS | SPATIAL | LABEL | OTHER | NOSOURCE

The following list contains descriptions of additional COPY statement arguments:

ENTRYTYPE=*type*
Specifies the type of GIS catalog entry to copy. Values for *type* are

- GISSPA or SPATIAL
- GISMAP or MAP
- GISLAYER or LAYER
- GISCOVER or COVERAGE.

This argument can be omitted if a complete, four-level entry name is specified. The following are identical:

- COPY MAPS.USA.STATE ET=GISMAP...
- COPY MAPS.USA.STATE.GISMAP...

Note: When you specify four-level entry names, *type* must be the actual SAS/GIS catalog entry extension, for example, GISMAP, not MAP. △

DESTCAT=*libref.catalog*
Specifies the destination for the copied catalog entries.
If *libref* is omitted, WORK is used as the default. Entries are copied to WORK.*catalog*. If DESTCAT= is omitted, *libref* defaults to WORK and the *catalog* to the catalog name of the source being copied. For example, if you are copying MAPS.USA.STATE, and you omit DESTCAT=, the copy of the data set is written to WORK.USA.STATE.

DESTLIB=*libref*
Specifies the destination library for the copied data sets.
If DESTLIB= is omitted, the default *libref* is WORK.

BLANK
Specifies that internal pathnames should be cleared in the copied entries.

REPLACE
Specifies that both existing catalog entries and data sets that have the same name as copied entries and data sets should be overwritten.

ALIAS=(*old-lib-1=new-lib-1 <, ... ,old-lib-n=new-lib-n>*)
Specifies libref translations. The *old-lib* value is the libref that is stored in the existing catalog entry. The *new-lib* value is the libref that you want to substitute in the new copy of the entry.

SELECT=
SELECT= specifies which data sets or catalog entries that are referenced by the source entry should be copied. Values for this option are

ALL
Copies all dependent catalog entries and data sets. It is equivalent to specifying both ENTRY and DATA.

ENTRY
Copies all dependent catalog entries

DATA SETS
Copies all dependent data sets. It is equivalent to specifying SPATIAL, LABEL, and OTHER.

SPATIAL
Copies dependent spatial data sets

LABEL
Copies dependent label data sets

OTHER
Copies other dependent data sets (besides spatial and label data sets), such as linked attribute data sets

NOSOURCE
Copies entry dependents as specified, but does not copy the specified source entry.

MOVE Statement

MOVE <*libref.catalog.*>*entry*<*.type*> </ *options*>;

Description

MOVE moves a SAS/GIS catalog entry or data set. You can move a single GIS entry or include the dependent entries and data sets that are referenced by the source.

Before you use the MOVE statement on a catalog entry or data set, make sure that you have WRITE permission to the source location. The MOVE statement deletes the original entry or data set and creates a new copy in the target directory. If you do not have WRITE permission to the source location, MOVE leaves the original entry or data set in its directory and creates a copy in the target directory.

MOVE Statement Optional Arguments

□ ENTRYTYPE=*type*

- DESTCAT=*libref.catalog*
- DESTLIB=*libref*
- BLANK
- REPLACE
- CHECKPARENT
- ALIAS=(*old-lib-1=new-lib-1 <, ... ,old-lib-n=new-lib-n>*)
- SELECT= _ALL_ | ENTRY | DATASETS | SPATIAL | LABEL | OTHER | NOSOURCE

The following list contains descriptions of additional MOVE statement arguments:

ENTRYTYPE=*type*
Specifies the type of GIS catalog entry to move. Values for *type* are

- GISSPA or SPATIAL
- GISMAP or MAP
- GISLAYER or LAYER
- GISCOVER or COVERAGE.

This argument can be omitted if a complete, four-level entry name is specified. The following are identical:

- MOVE MAPS.USA.STATE ET=GISMAP...
- MOVE MAPS.USA.STATE.GISMAP...

Note: When you specify four-level entry names, *type* must be the actual SAS/GIS catalog entry extension, for example, GISMAP, not MAP. △

DESTCAT
specifies the destination for the moved catalog entries.

If *libref* is omitted, WORK is used as the default. Entries are moved to the WORK.*catalog*. If DESTCAT= is omitted, *libref* defaults to WORK and the *catalog* to the catalog name of the source being moved. For example, if you are moving MAPS.USA.STATE, and you omit DESTCAT=, the data set that you are moving is written to WORK.USA.STATE.

DESTLIB
specifies the destination for the moved data sets.

If DESTLIB= is omitted, the default *libref* is WORK.

BLANK
specifies that internal pathnames should be cleared in the moved entries.

REPLACE
specifies that both existing catalog entries and data sets that have the same name as moved entries and data sets should be overwritten.

CHECKPARENT
specifies that data sets and catalog entries are checked before they are moved to see what other GIS entries references them. If any references are found, the catalogs and data sets are copied instead of being moved.

If CHECKPARENT is not specified (the default), data sets and catalog entries are moved without checking for references, which may cause problems with other GIS entries.

CAUTION:
Do not use host commands to move or rename SAS data sets that are referenced in GISSPA entries. Moving or renaming a data set that is referred to in a spatial

entry breaks the association between the spatial entry and the data set. To prevent breaking the association, use the PROC GIS MOVE statement with the CHECKPARENT option instead of a host command. △

ALIAS=
specifies libref translations. The *old-lib* value is the libref that is stored in the existing catalog entry. The *new-lib* value is the libref that you want to substitute in the moved entry.

SELECT=
specifies which data sets or catalog entries that are referenced by the source entry should be moved. Values for this argument are

ALL
moves all dependent catalog entries and data sets. Equivalent to specifying both ENTRY and DATA.

ENTRY
moves all dependent catalog entries

DATA
moves all dependent data sets. It is equivalent to specifying SPATIAL, LABEL, and OTHER.

SPATIAL
moves dependent spatial data sets

LABEL
move dependent label data sets

OTHER
moves other dependent data sets (besides spatial and label data sets), such as linked attribute data sets

NOSOURCE
moves entry dependents as specified, but does not move the specified source entry.

SYNC Statement

SYNC <*libref.catalog.*>*entry*<*.type*> </ *options*>;

Description

SYNC updates a SAS/GIS catalog entry libref and the internal pathname. ENTRYTYPE=*type* specifies the type of entry to update. Values for *type* are

- GISSPA or SPATIAL
- GISMAP or MAP
- GISLAYER or LAYER
- GISCOVER or COVERAGE

This argument can be omitted if you specify a complete, four-level entry name.

SYNC Statement Optional Arguments

- ENTRYTYPE=*type*

- BLANK
- ALIAS=(*old-lib-1=new-lib-1 <, ... ,old-lib-n=new-lib-n>*)
- SELECT= _ALL_ | ENTRY

The following list contains descriptions of additional SYNC statement arguments:

ENTRYTYPE=*type*
Specifies the type of GIS catalog entry to move. Values for *type* are

- GISSPA or SPATIAL
- GISMAP or MAP
- GISLAYER or LAYER
- GISCOVER or COVERAGE.

This argument can be omitted if a complete, four-level entry name is specified. The following are identical:

- SYNC SASUSER.MALL.STORES ET=GISMAP...
- SYNC SASUSER.MALL.STORES.GISMAP ...

Note: When specifying four-level entry names, *type* must be the actual SAS/GIS catalog entry extension, for example, GISMAP, not MAP. △

BLANK
BLANK specifies that internal pathnames should be cleared in the updated entries.

ALIAS=(*new-lib-1=old-lib-1 <, ... ,new-lib-2=old-lib-2>*)
Specifies libref translations. The *old-lib* value is that the libref that is stored in the existing catalog entry. The *new-lib* value is the libref you that want to substitute in the synchronized version of the entry.

SELECT=
Specifies which catalog entries that are referenced by the source entry should be updated. Values for this argument are

ALL
Updates all dependent catalog entries. This is equivalent to specifying ENTRY.

ENTRY
Updates all dependent catalog entries.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *Working with Spatial Data Using SAS/GIS® Software, Version 8*, Cary, NC: SAS Institute Inc., 1999.

Working with Spatial Data Using SAS/GIS® Software, Version 8

ISBN 1–58025–519–1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999