Introduction

The GIS procedure creates and maintains the spatial databases 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 points and lines that make up a map). 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 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 is available for display in a map.
- one or more layer entries, SAS catalog entries of type GISLAYER that identify features with 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 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 using the GIS Spatial Data Importing window.

### PROC GIS Syntax

```sas
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 composite1 ENCLOSES composite2;
    COVERAGE operation <libref.catalog.>coverage-entry </options>;
    LAYER operation <libref.catalog.>layer-entry </options>;
    MAP operation <libref.catalog.>map-entry </options>;
    COPY <libref.catalog.>entryENTRY TYPE=typeDESTCAT=libref.catalog
            DESTLIB=libref</options>;
    MOVE <libref.catalog.>entryENTRY TYPE=typeDESTCAT=libref.catalog
            DESTLIB=libref</options>;
    SYNC <libref.catalog.>entryENTRYTYPE=type</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 following 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 and identifies the current spatial entry:

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

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

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

```sas
composite create state /
    class=state
```
PROC GIS Statement

The PROC GIS statement invokes the GIS procedure and, optionally, specifies the default SAS catalog in which GISSPA, GISCOVER, GISLAYER, and GISMAP 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

Each of the statement forms specifies the default SAS catalog in which GISSPA, GISCOVER, GISLAYER, and GISMAP entries referred to in subsequent statements in the PROC GIS step are stored.

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 assumed.

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 named in the CATALOG statement rather than the one specified in the CATALOG= argument.
- 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 specified with the CATALOG= option. Refer to the description of these statements later in this chapter for more information.

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.

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.
CATALOG Statement

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

The CATALOG statement identifies the default SAS catalog in which GISSPA, GISCOVER, GISLAYER, and GISMAP entries are stored when you specify one-level catalog entry names in subsequent statements in the PROC step.

Note: The CATALOG statement permanently overrides the CATALOG= argument specified in the PROC GIS statement. If you use the CATALOG= argument in the PROC GIS statement and then submit a CATALOG statement, subsequent statements in the GIS procedure refer to the catalog named in the CATALOG statement.

CONTENTS Option

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

Libref and Catalog—name Arguments

<libref.>catalog-name specifies the SAS catalog in which GISSPA, GISCOVER, GISLAYER, and GISMAP entries referred to in subsequent statements in the PROC GIS step are stored.

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 assumed.

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 created or updated during an invocation of the GIS procedure are stored in the current spatial entry. Any subsequent COVERAGE statements 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 with 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 GISSPA entry the current spatial entry for subsequent operations.

The following list contains descriptions of the SPATIAL statement operations:

**CONTENTS**

CONTENTS prints information about the contents of the specified GISSPA 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**

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

DELETE deletes the specified GISSPA entry. By default, any polygonal index data sets referred to in the GISSPA entry are also deleted. The chains, nodes, or detail data sets 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 action. 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 GISSPA 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 using the _ALL_ keyword.

REPLACE

REPLACE re-creates the specified GISSPA entry and removes any previously defined composites, polygonal index names, and lattice hierarchies. If an entry with the specified name does not exist, REPLACE creates a new GISSPA entry. The new 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

UPDATE updates the specified GISSPA entry by applying new values for specified arguments. The updated spatial entry becomes the current spatial entry for the GIS procedure.

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 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

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

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 GISSPA entry and the data set. To
prevent breaking the association, use the GIS MOVE statement with the CHECKPARENTS 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 following 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'

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

CARTESIAN | LATLON

CARTESIAN | LATLON specifies the coordinate system used in the stored spatial data.

CARTESIAN

data are in an arbitrary rectangular coordinate system

LATLON

data are in a geographic (spherical) coordinate system.

The default is LATLON.

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

CHAINS

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.
If you use a one-level name, the data set will be created (or replaced) in the library specified in the CATALOG= option on the PROC GIS statement. Use a two-level name to override the default library.

DEGREES | RADIANS | SECONDS

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

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 containing detail definitions might describe the curvy outline of a coastal road.

If you use a one-level name, the data set will be created (or replaced) in the library specified in the CATALOG= option on the PROC GIS statement. Use a two-level name to override the default library.

EAST | WEST

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.

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

KEEP

KEEP specifies that polygonal index data sets are retained when the spatial entry is deleted. This option is valid only with the DELETE operation.

MERGE

MERGE=(<libref.catalog.spatial-entry-1< ..., <libref.catalog.spatial-entry-n>><EDGEMATCH | OVERLAP>) builds a new spatial entry which references two or more existing GISSPA entries. The dependent data sets for the spatial entries are not physically merged.

An error occurs if any of the specified spatial entries do not exist.

If you specify a one-level name for any of the entries to be merged, the spatial entry is assumed to be in the catalog specified in the CATALOG= argument with the PROC GIS statement or in the most recently issued CATALOG statement. An error occurs if no catalog has previously been specified.

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 using values from the adjoining spatial data sets. The affected chains data sets are
rewritten unless the LINKONLY option is specified, and there is no way to reverse the operation.

OVERLAP

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

MULT

MULT=multiplier-value specifies a constant value by which spatial data coordinates are multiplied. The default is MULT=1.

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

NODES

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.

If you use a one-level name, the data set will be created (or replaced) in the library specified in the CATALOG= option on the PROC GIS statement. Use a two-level name to override the default library.

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

NORTH | SOUTH

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

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

DESCRIPTION

DESCRIPTION='string' specifies a descriptive string, up to 256 characters long, that is stored in the description field of the GISSPA 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:

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

**Update an existing spatial entry**

The following code fragment replaces the existing detail data set with MAPS.USAD for an existing spatial entry:

```
spatial update maps.usa.usa / details=maps.usad;
```
Merge three existing spatial databases

The following code fragment creates a new spatial entry named TRIANGLE.GISSPA in the current catalog by merging three existing spatial entries, ORANGE, DURHAM, and WAKE. In this example, each of the GISSPA entries to be merged is stored in a different library.

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

COMPOSITE Statement

```plaintext
COMPOSITE operation composite-name </arguments>;
```

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, 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: You can 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

CREATE defines associations between variables in the chains and nodes data sets and stores these associations 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. However, even those that are represented by a single SAS data set variable must be declared with a COMPOSITE CREATE statement if you want to use them in a GIS statement (such as a LAYER statement to define a LAYER entry).

DELETE

**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 using the `_ALL_` keyword.

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 COMPOSITE DELETE 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 COMPOSITE DELETE statement.
- the special value `_ALL_` to delete all the composites in the current spatial entry.

REPLACE

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

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 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).

Optional Arguments

When you specify CREATE, REPLACE, or UPDATE for operation in a COMPOSITE statement, you can specify one or more of these options following 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

BILATERAL indicates that the composite is a left/right type. 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:

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

Note: BILATERAL composites are used to define polygonal layers in a LAYER statement.

CLASS

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.

VAR

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 keywords, except in the following circumstances:

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

CLASS=ADDRESS

ADDRESS indicates that the composite defines addresses in the chains data set.

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)
```

CLASS=AREA

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 address matching.

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)
```

CLASS=CITY | PLACE

CITY | PLACE indicates that the composite defines features related to geographic location, such as cities. Composites of this class are used in address matching.
By default, CITY is not considered an AREA-type composite. If your spatial data contains closed city boundaries, you must explicitly define the composite as an AREA class as well:

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

CLASS=CLASSIFICATION

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 adding points to the map interactively in the GIS Map window, at least one CLASSIFICATION-type composite must be defined in the spatial entry.

CLASS=DIRECTION_PREFIX

DIRECTION_PREFIX indicates that the composite defines the directional prefix component of an aggregate feature name, such as the North in North Main Ave.

CLASS=DIRECTION_SUFFIX

DIRECTION_SUFFIX indicates that the composite defines the direction suffix component of an aggregate feature name, such as the South in 2nd St South.

CLASS=NAME

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.

CLASS=PLUS4

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 contains 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);

CLASS=TYPE

TYPE indicates that the composite defines the feature type component of an aggregate feature name, such as the Ave in N Harrison Ave.

CLASS=X

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

CLASS=Y

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

CLASS=ZIPCODE

ZIPCODE indicates that the composite defines postal delivery codes in the chains data. Composites of this class are used in address matching.
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:

```sas
composite create zip /
  var=(zipl zipr)
  class=(zipcode area);
VAR=ssociation-declaration
```

The association-declaration argument for the VAR= option can be one of the following, depending on the class-type values 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 which 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 consisting 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:

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

#### Define a composite for a bilateral feature

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

```sas
composite create state /
  var=(left=statel,right=stater)
```
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:

```sql
composite create custadd /
    var=(fromleft=elmbegl,fromright=elmbegr,
        toleft=elmendl,toright=elmendr)
    class=address;
run;
```

### POLYGONAL INDEX Statement

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

### Description

The POLYGONAL INDEX statement creates, modifies, or deletes polygonal index data sets 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. The spatial database must include a polygonal index data set for each feature 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 (GISSPA) entry to determine area boundaries. The composites used for polygonal indexes must have the CLASS attribute AREA (or one of the political subdivision area classes such as COUNTRY, STATE, or COUNTY which imply AREA).

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

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

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 using the _ALL_ keyword.

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 argument 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.

REPLACE

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 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

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 an existing data set owned by a different spatial entry specified in the OUT= argument already exists, you must use the FORCE argument to cause it to be overwritten.

Polygonal-index

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).

**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.

- COMPOSITE
- ERRORS
- FORCE
- KEEP
- OUT

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:

**COMPOSITE**

COMPOSITE=(composite-1<, ..., composite-n>) specifies the composite or list of composites that define the boundaries of the enclosed polygonal areas used to create the index. If the composite list consists of a single composite name, 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 AREA.

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

**ERRORS**

ERRORS=<number> specifies whether messages about any topological errors that are detected while the index is being constructed are written to the SAS log. 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**

FORCE indicates that an existing polygonal index data set specified in the OUT= argument can be overwritten 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**

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

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

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

If you use a one-level name, the data set will be created (or replaced) in the library specified in the CATALOG= option on the PROC GIS statement. Use a two-level name to override the default library.

CAUTION:
Do not use host commands to move or rename polygonal index data sets. Because the polygonal index data set names are referred to 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 GIS MOVE statement with the CHECKPARENTS option instead of a host command.

POLYGONAL INDEX Statement Example

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

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

LATTICE Statement

LATTICE composite-names;

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 re-evaluated 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 are not automatically 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 (GISSPA) 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.
**LATTICE Statement Arguments**

The following arguments can be used with the LATTICE statement:

```
composite-1
ENCLOSES | ->
composite-2
```

An error occurs if there is no current spatial entry for the GIS procedure. Use the SPATIAL statement 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:

**Composite-1**

Composite-1 is an area composite that geographically contains other enclosed AREA type composites. Composite-1 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 composite-2 is a single area composite which is not contained within other enclosed areas and which does not itself enclose any other areas.

**ENCLOSES**

`ENCLOSES | ->` is the separator between LATTICE composites.

**Composite-2**

Composite-2 is an area composite that is geographically within composite-1's polygonal areas, or a single area not contained by another when `_UNIVERSE_` is composite-1. Composite-2 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 associations, 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    ----> county
                 county    ----> tract
```
tract ---> 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. You 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 used to create a map and a subsetting WHERE clause to define the subset of spatial data, or coverage, of the map you want to display.

For example, you could create a GISCOVER entry, MYCAP, that contains geographic information for your state capital. MYCAP subsets the spatial database defined in the spatial entry MYSTATE, which contains geographic information 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 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

CONTENTS prints information about the contents of the specified GISCOVER 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

CREATE creates a new GISCOVER entry to subset a spatial entry.

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

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

DELETE

DELETE removes the specified GISCOVER 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 action, you can also specify the special value _ALL_ for the coverage entry name argument to delete all GISCOVER 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 using the _ALL_ keyword.

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

REPLACE

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

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

UPDATE

UPDATE modifies the specified GISCOVER 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 GISCOVER entry you want to create, delete, replace, or update. The coverage-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).

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

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

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

SPATIAL

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

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

WHERE

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 less.

To specify a WHERE expression greater than 200 characters, you must break the expression up 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. 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 named GMAPS.USA.ALL.GISCOVER that defines a coverage of the entire spatial database defined in GMAPS.USA.USA.GISSPA:

```sas
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 named SEAST of type GISCOVER that defines a coverage of only the selected states from the current spatial entry:

```sas
coverage create seast /
   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 a previously created 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, RIVERS, to represent the water features in your spatial data.

You can define layers as either static or thematic. A static layer has a fixed set of graphical attributes (fill colors, outline colors, and so forth). A thematic layer 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 are 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**

CONTENTS displays the contents of the specified GISLAYER 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**

CREATE creates a new GISLAYER entry to define a layer 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**

DELETE removes the specified GISLAYER 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 GISLAYER entries in the current catalog.

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

**REPLACE**

REPLACE overwrites the specified GISLAYER entry or creates a new GISLAYER 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

UPDATE modifies the specified GISLAYER 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 GISLAYER entry you want to create, delete, replace, or update. The general form of the argument is

<libref.catalog.layer-entry>

If you specify a one-level name, the layer entry is assumed to be in the catalog 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 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).

---

**Optional Arguments**

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

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

- **COMPOSITE=composite-name**
- **DESCRIPTION='string'**
- **DETAILON=scale-value**
- **DETAILS | NODETAILS**
- **MAP=<libref.catalog.map-entry>**
- **LABELON=label-on-scale**
- **OFFSCALE=scale-value**
- **ONSCALE=scale-value**
- **STATIC | THEMATIC**
- **TYPE=POINT | LINE | AREA**
The following list contains descriptions of additional LAYER statement arguments:

COMPOSITE

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 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=STATERT), then the implied WHERE clause for the layer is WHERE STATEL NE STATERT.

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

DESCRIPTION='string' or DES='string' specifies a descriptive string, up to 256 characters long, that is stored in the description field of the GISLAYER entry. The default description is blank.

DETAILON

DETAILON=scale-value specifies the scale at or below which detail coordinates are displayed, provided that detail coordinates 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 only effective when detail coordinates are read for the layer. The DETAILS argument controls whether detail coordinates are read.

DETAILS | NODETAILS

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

MAP=libref.catalog.map-entry identifies a GISMAP entry that provides theme information for layers created in version 6.11 of SAS/GIS. This option is ignored for later version layers. 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 Version 6.11 layers.

Note: The MAP= argument is valid only in conjunction with the CONTENTS keyword and is the only option permitted with the CONTENTS operation.

LABELON

LABELON=scale-value specifies the 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

OFFSCALE=scale-value specifies the 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

ONSCALE=scale-value specifies the 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

STATIC toggles the current theme in the layer off so 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.

THEMATIC

THEMATIC toggles the current theme in the layer on so 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.

TYPE

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 are 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 filled polygons.

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

UNITS

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

**ENGLISH**

selects miles per inch (MI/IN) as the scale units.

**METRIC**

selects kilometers per centimeter (KM/CM) as the scale units.

**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=('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 less.

To specify a WHERE expression greater than 200 characters, you must break the expression up 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. 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 keyword. For area layers, you must use the COMPOSITE= argument. If you use the WHERE= argument, the default layer type is LINE.

**FORCE**

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

**THEME**

THEME=(options) allows you to manipulate thematic layers. When you specify CONTENTS, CREATE, REPLACE, or UPDATE for the operation keyword in a
LAYER statement and specify THEME as an option, you can specify the following additional arguments.

- **operation**
  - LINK=link-name
  - THEMEVAR=var-name
  - RANGE=DEFAULT | DISCRETE | LEVELS
  - NLEVELS=int
  - POSITION=int
  - MAKE_CURRENT | NOT_CURRENT

The following list contains descriptions of the THEME arguments:

**THEME Operations**

In the LAYER statement THEME argument, the operation argument can be one of the following:

- **CREATE**
  - CREATE creates a new theme for the specified GISLAYER 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 FORCE. The THEME CREATE statement does not overwrite existing theme entries. Use THEME REPLACE to replace an existing entry.

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

- **REPLACE**
  - REPLACE overwrites the specified theme for the GISLAYER entry. The THEME REPLACE statement has the effect of canceling the previously issued THEME CREATE statement for the specified layer entry.

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

- **UPDATE**
  - UPDATE updates the specified theme for the GISLAYER entry by applying new values for specified arguments.

  An error occurs if the specified layer does not have at least one existing theme. For a THEME UPDATE statement, you must specify a value for at least one of the arguments LINK=, VAR=, RANGE=, NLEVELS=, or MAKE_CURRENT | 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**
  - DELETE removes the specified theme for the specified GISLAYER entry.

  For a THEME DELETE statement, you must specify a value for THEMEVAR=varname or POSITION=int. An error occurs if you specify THEMEVAR=varname and 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 deleting the layer theme.

**THEME LINK**

`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**

`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.

**THEME RANGE**

`RANGE=DEFAULT | DISCRETE | LEVELS` specifies the thematic range type.

- **DEFAULT**
  
  increments are calculated automatically using a SAS/GRAPH statistical algorithm.

- **DISCRETE**
  
  the range is treated as a series of discrete values instead of a continuous variable. If the variable specified in the `VAR=` option 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=int` instead.

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

**THEME NLEVELS**

`NLEVELS=int` specifies the number of range breaks 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.

**THEME POSITION**

`POSITION=int` specifies the position number of the target theme. 1 is the first theme, 2 is the second theme, and so on. Negative values of `int` 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.
THEME MAKE_CURRENT

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.

THEME NOT_CURRENT

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

---

**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:

```plaintext
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. Δ

**Define a layer using a category variable**

Assume that the spatial database contains a variable named CFCC with 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:

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

*Note:* The colon (:) modifier on 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." Δ
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.

```plaintext
proc gis;
  layer update sasuser.mall.store /
    map = sasuser.mall.stores
    theme = (create
      themevart = sqft
      link = mallstor
      range = discrete
      pos = -1
      not_current);
run;
```

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 operation 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

CONTENTS prints information about the contents of the specified GISMAP 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 map's display and projection option settings
- a list of associated data sets and link names
- a list of the operations 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

CREATE creates a new GISMAP entry defining 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

DELETE removes the specified GISMAP 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 GISMAP 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 using the _ALL_ keyword.

REPLACE

REPLACE overwrites the specified GISMAP 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

UPDATE modifies the specified GISMAP entry by applying new values for specified arguments.

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

**Entry Name**

In the MAP statement, the map entry-name identifies the GISMAP 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 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).

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)
- MULT=multiplier-value
- LABEL=libref.dsname | NONE
- ACTION=(operation-arguments)
- ATTRIBUTE=(attribute-arguments)
- RENAME_LAYER old-name = new-name
- FORCE
- NOWARN

The following list contains descriptions of additional MAP statement options:

CARTESIAN | LATLON

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

- CARTESIAN
  - data are in an arbitrary rectangular 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
If the spatial entry specifies the LATLON coordinate system, then you must also specify the LATLON argument for the MAP statement.

**COVERAGE**

```
COVERAGE=libref.catalog.>coverage-entry (or
COVER=libref.catalog.>coverage-entry) specifies the GISCOVER entry to which
the map refers. The coverage determines the geographic extent of the map.
```

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

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

**DEGREES | RADIANS | SECONDS**

```
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 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**

```
DESCRIPTION='string' or DES='string' specifies a descriptive string, up to 256
characters long, that is stored in the description field of the GISMAP entry. The
default description is blank.
```

**DETAILS | NODETAILS**

```
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**

```
LAYERS=(libref.catalog.>layer-entry-1, ..., libref.catalog.>layer-entry-n)
specifies a list of GISLAYER entry names. The layer list determines which
components of the spatial data within the specified coverage are available in the
map and how they are displayed.
```

If you specify one-level names for any of the layer entries, the entries are
assumed to be in the catalog specified in the PROC GIS statement or in the most
recently issued CATALOG statement. An error occurs if no catalog has previously
been specified.

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

**MULT**

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

**LABEL**

```
LABEL=libref.dsname | NONE assigns or removes a label data set reference.
Specifying libref.dsname assigns the data set and removes any existing data set
```
ACTION

ACTION lets you copy, delete, or update GIS actions associated with a map entry. The arguments used with ACTION are

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

The following list contains descriptions of the ACTION arguments:

ACTION COPY

ACTION COPY copies existing operations from one map entry to another. Specify the map entry containing the operations to be copied with the FROM=map-entry argument. The operations are copied to the map specified in the MAP statement.

Specify the operations to be copied with the NAME=link-name argument. Specifying NAME=_ALL_ copies all operations in the specified map. Existing operations in the map to be updated are not overwritten unless you specify the FORCE argument.

ACTION DELETE

ACTION DELETE removes an existing operation from the map entry. Specify the operation to be deleted with the NAME=operation-name argument. Specifying NAME=_ALL_ deletes all operations. Use the NOWARN argument to suppress messages when an operation is not found.

ACTION UPDATE

ACTION UPDATE modifies existing operations in the map being updated. Specify the operation to be updated with the NAME=operation-name argument. Specifying NAME=_ALL_ updates all operations. NAME=_ALL_ is required for UPDATE.

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

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

ACTION NAME

NAME=operation-name | _ALL_ specifies the operation to be copied, deleted, or updated. operation-name identifies a single operation, while _ALL_ specifies all operations.

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

WHEN = OFF | IMMEDIATE | DEFERRED is used with ACTION UPDATE to change the execution setting of the specified operation.

- **OFF**
  - the operation will not be executed when a layer element is selected.

- **IMMEDIATE**
  - the operation will be executed as soon as a layer element is selected.

- **DEFERRED**
  - the operation’s execution will be deferred when a layer element is selected.

ACTION FROM

FROM = map-entry is used with ACTION COPY and specifies the source map entry containing operations to be copied. Specify the operations to be copied from the map with the NAME = operation-name argument.

ACTION RENAME

RENAME = new-operation-name renames the link specified in the NAME = operation-name for ACTION 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

- **COPY**
- **DELETE**
- **UPDATE**
- **NAME = link-name | _ALL_**
- **FROM = map-entry**
- **RENAME = new-link-name**

The following list contains descriptions of the ATTRIBUTE arguments:

ATTRIBUTE COPY

ATTRIBUTE COPY copies existing attribute data links from one map entry to another. Specify the map entry containing the links to be copied with the FROM = map-entry argument. The links are copied to the map specified in the MAP statement.

Specify the link to be copied with the NAME = link-name argument. Specifying NAME = _ALL_ copies all links in the specified map. Existing links in the map to be updated are not overwritten unless you specify the FORCE argument.

ATTRIBUTE DELETE

ATTRIBUTE DELETE removes an existing attribute data link from the map entry. Specify the link to be deleted with the NAME = link-name argument. Specifying NAME = _ALL_ deletes all data links. Use the NOWARN argument to suppress messages when a link is not found. This does not delete the attribute data set, only the link.
ATTRIBUTE UPDATE

ATTRIBUTE UPDATE modifies existing data links in the map being updated. Specify the link to be updated with the NAME=link-name argument. Specifying NAME=_ALL_ updates 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_.

ATTRIBUTE NAME

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 ATTRIBUTE UPDATE with the RENAME argument.

ATTRIBUTE FROM

FROM=map-entry is used with ATTRIBUTE COPY and specifies the map entry containing data links to be copied. Specify the links to be copied from the map with the NAME=link-name argument.

RENAME

RENAME=new-link-name renames the link specified in the NAME=link-name for the ATTRIBUTE UPDATE operation.

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

RENAME_LAYER

RENAME_LAYER old-name=new-name changes the name of an existing layer in the map 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.

FORCE

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

NOWARN

NOWARN specifies that messages are not to be issued about operations or attribute links not being found during deletion. Use this argument when specifying ACTION DELETE or ATTRIBUTE DELETE.

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.

```plaintext
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:

```plaintext
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:

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

---

**COPY Statement**

COPY libref.catalog.entry ENTRYTYPE=type DESTCAT=libref.catalog
  DESTLIB=libref <options>

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

---

**ENTRYTYPE Argument**

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

---

**DESTCAT Argument**

DESTCAT=libref.catalog specifies the destination for the copied catalog entries.

---

**DESTLIB Argument**

DESTLIB=libref specifies the destination for the copied data sets.
Optional Arguments

- BLANK
- REPLACE
- ALIAS=(old-lib-1=new-lib-1 < ... ,old-lib-n=new-lib-n>)
- SELECT=_ALL_ | ENTRY | DATASETS | SPATIAL | LABEL | OTHER

The following list contains descriptions of additional COPY statement arguments:

BLANK

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

REPLACE

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

ALIAS

ALIAS=(old-lib-1=new-lib-1 < ... ,old-lib-n=new-lib-n>) specifies libref translations. In each pair, new-lib is the libref to which to change and old-lib is the libref to change from.

SELECT

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

_ALL_

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

ENTRY

copy all dependent catalog entries

DATA

copy all dependent data sets. Equivalent to specifying SPATIAL, LABEL, and OTHER.

SPATIAL

copy dependent spatial data sets

LABEL

copy dependent label data sets

OTHER

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

MOVE Statement

MOVE libref.catalog.entry ENTRYTYPE=type DESTCAT=libref.catalog DESTLIB=libref <options>

MOVE moves a GIS catalog entry or data set. You can move a single GIS object or include the dependent entries and data sets referenced by the source.
ENTRYTYPE Argument
ENTRYTYPE=type specifies the type of entry to move. Values for type are
- GISSPA or SPATIAL
- GISMAP or MAP
- GISLAYER or LAYER
- GISCOVER or COVERAGE

DESTCAT Argument
DESTCAT=libref.catalog specifies the destination for the moved catalog entries.

DESTLIB Argument
DESTLIB=libref specifies the destination for the moved data sets.

Optional Arguments
- BLANK
- REPLACE
- CHECKPARENT
- ALIAS=(old-lib-1=new-lib-1 <, ... ,old-lib-n=new-lib-n>)
- SELECT=_ALL_ | ENTRY | DATASETS | SPATIAL | LABEL | OTHER

The following list contains descriptions of additional MOVE statement arguments:

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

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

CHECKPARENT
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.

ALIAS
ALIAS=(old-lib-1=new-lib-1 <, ... ,old-lib-n=new-lib-n>) specifies libref translations. In each pair, new-lib is the libref to which to change and old-lib is the libref to change from.

SELECT
SELECT specifies which data sets or catalog entries referenced by the source entry should be moved. Values for this argument are
_ALL_
move all dependent catalog entries and data sets. Equivalent to specifying both ENTRY and DATA.

ENTRY
move all dependent catalog entries

DATA
move all dependent data sets. Equivalent to specifying SPATIAL, LABEL, and OTHER.

SPATIAL
move dependent spatial data sets

LABEL
move dependent label data sets

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

SYNC Statement

SYNC libref.catalog.entry ENTRYTYPE=type <options>
SYNC updates a GIS catalog entry libref and the internal pathname.

ENTRYTYPE Argument
ENTRYTYPE=type specifies the type of entry to update. Values for type are

- GISPA or SPATIAL
- GISPAM or MAP
- GISLAYER or LAYER
- GICOVER or COVERAGE

Optional Arguments

- 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:

BLANK

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

ALIAS

ALIAS=(new-lib-1=old-lib-1,< ... ,new-lib-2=old-lib-2>) specifies libref translations. In each pair, new-lib is the libref to which to change and old-lib is the libref to change from.

SELECT

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

_ALL_

update all dependent catalog entries. Equivalent to specifying ENTRY.

ENTRY

update all dependent catalog entries