SAS® Publishing

Base SAS® Guide to Information Maps

THE POWER TO KNOW®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2006. *Base SAS® Guide to Information Maps*. Cary, NC: SAS Institute Inc.

Base SAS® Guide to Information Maps

ISBN 13: 978-1-59047-866-0
ISBN 10: 1-59047-866-5

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

1st printing, February 2006
2nd printing, October 2006

Contents

CHAPTER 1 Overview of SAS Information Maps

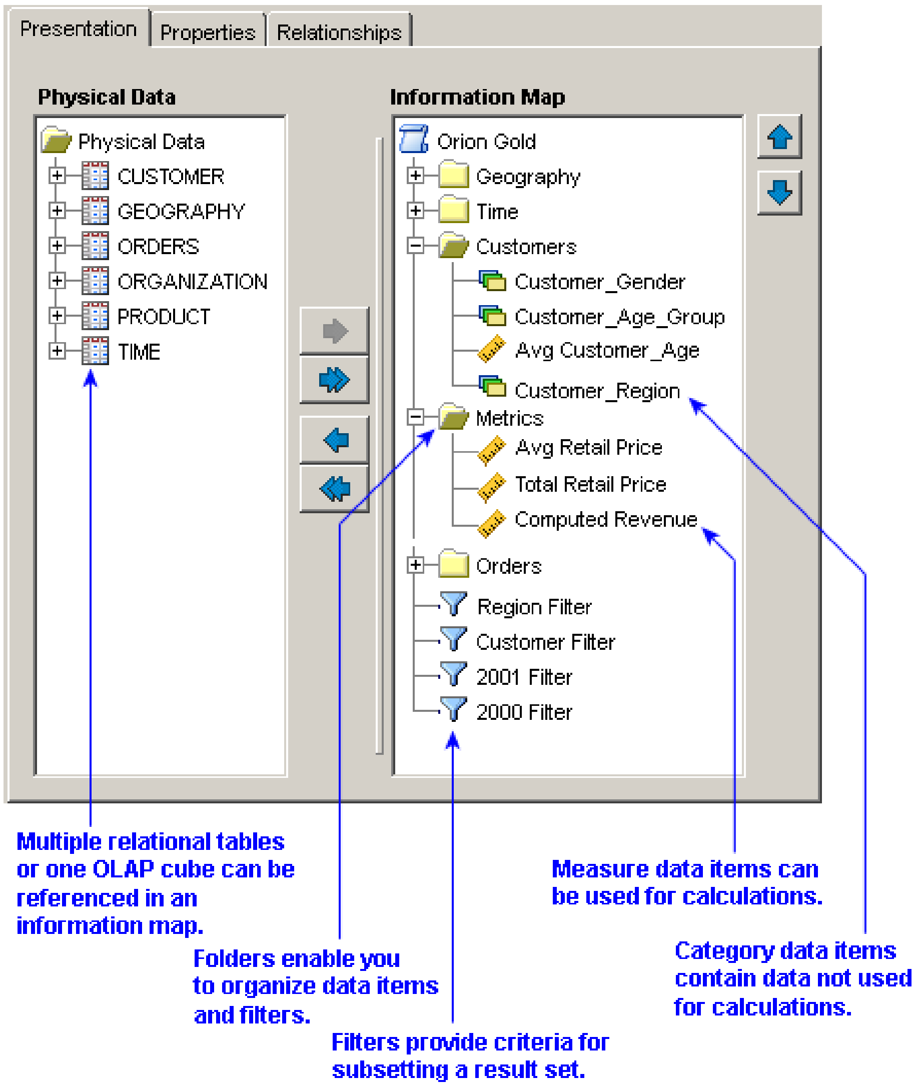
What Is a SAS Information Map?

A *SAS Information Map* is business metadata about your data. Information maps are user-friendly metadata definitions of data sources and enable your business users to query data in order to meet specific business needs. *Metadata* is information about the structure and content of data and the applications that process and manipulate that data. Note that an information map does not contain any physical data.

An information map contains data items and filters. A *data item* can refer to a physical data field or a calculation. A *filter* contains criteria for subsetting data. Data items and filters are used for building queries.

Depending on the data source, an information map might have many data items and filters. *Folders* can be used to organize the data items and filters in order to make it easy for business users to find what they want within the information map.

The following figure shows you what an information map looks like from within SAS Information Map Studio, an application that provides a graphical user interface for creating and viewing information maps.



Why Are SAS Information Maps Important?

Information maps provide a business metadata layer that enables business users to ask questions and get answers themselves. This frees IT resources from one-time reporting requests and reduces the need to provide training in programming and database structures.

Information maps enable business users to easily access enterprise-wide data by providing the following benefits:

- Users are shielded from the complexities of the physical data.
- Data storage is transparent to users, regardless of whether the data is relational or multidimensional or whether the data is in a SAS data set or in a third-party database management system.
- Business formulas and calculations are predefined, which makes them usable on a consistent basis.
- Users can easily query data for answers to business questions without having to know query languages.

CHAPTER 2 The INFOMAPS Procedure

Overview: INFOMAPS Procedure

The INFOMAPS procedure enables you to create information maps programmatically. You can also use the procedure to modify an existing information map by adding new data sources or new data items. For this release, however, you cannot change the definitions of any existing data item, filter, data source, folder, or relationship within an information map.

Information maps provide a layer of metadata that describes physical data in terms that business users can understand. Information maps enable you to give descriptive labels and detailed descriptions to assist business users. For example, you can create data items with names such as "Age Group" or "Sales Revenue from Internet Orders".

Information maps can contain data items and filters. Data items refer to physical data. A data item can refer to a single column from a physical table or can contain an expression that utilizes zero or more columns. A data item is classified as either a category item or a measure item. You can organize data items into folders and subfolders to help users find the information they need.

In addition to using the INFOMAPS procedure to create information maps, you can also use the interactive client application, SAS Information Map Studio, to create, update, and manage information maps. When you have created or modified an information map, you can access it using the Information Maps engine and retrieve the data that the information map describes. For information, see Chapter 3, "Using the SAS Information Maps LIBNAME Engine," on page 31.

For information about defining metadata, installing and setting up a standard SAS Metadata Server, or changing the standard configuration options for the SAS Metadata Server, see the *SAS Intelligence Platform: System Administration Guide*.

Operating Systems Supported by the INFOMAPS Procedure

The following operating systems are supported:

- Windows (32–bit)
- Solaris (64–bit)
- UNIX (HP-UX, AIX 64–bit)

Syntax: INFOMAPS Procedure

PROC INFOMAPS
METAPASS="*password*"
METAPORT=*port-number*
METAREPOSITORY="*repository-name*"
METASERVER="*address*"
METAUSER="*user-ID*"
<MAPPATH="*repository-path*">;

DELETE INFOMAP *information-map-name*
<MAPPATH="*repository-path*">;

EXPORT <INFOMAP *information-map-name*>
<FILE=*fileref* |"*physical-path*">
<MAPPATH="*repository-path*">;

IMPORT FILE=*fileref* |"*physical-path*";

INSERT DATAITEM
COLUMN=*datasource-ID.column-name* | EXPRESSION="*expression-text*"
<AGGREGATION=*aggregate-function*>
<AGGREGATIONS_DROP_LIST=(*aggregate-function-list*)>
<CLASSIFICATION=CATEGORY | MEASURE>
<DESCRIPTION="*descriptive-text*">
<FOLDER=*folder-name* | "*folder-path*">
<FORMAT="*format-name*">
<ID="*dataitem-ID*">
<NAME="*name*">
<TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP>;

INSERT DATASOURCE
SASSERVER | SERVER=*server-name*
TABLE=*library.table* <COLUMNS=(*column–1 column–2 ... column-n*) | _ALL_ >
<ID=*datasource-ID*>;

INSERT DATASOURCE
SASSERVER | SERVER=*server-name*
CUBE=<*schema.*>*cube* <_ALL_>
<ID=*datasource-ID*>;

INSERT FILTER *filter-name*
CONDITION="*conditional-expression*"
<DESCRIPTION="*descriptive-text*">

<FOLDER=*folder-name* | "*folder-path*">;

INSERT FOLDER *folder-name*
<PARENT=*parent-folder-name*>
<DESCRIPTION="*descriptive-text*">;

INSERT RELATIONSHIP *left-table* **INNER | LEFT | RIGHT | FULL JOIN** *right-table* **ON** "*conditional-expression*";

LIST <DATAITEMS | DATASOURCES | FILTERS | _ALL_> ;

OPEN INFOMAP *information-map-name*
<CREATE_TARGET_FOLDER=YES | NO>
<INIT_CAP=YES | NO>
<MAPPATH="*repository-path*">
<REPLACE_UNDERSCORES=YES | NO>
<USE_LABELS=YES | NO>;

SAVE <INFOMAP *information-map-name*>
<MAPPATH="*repository-path*">;

PROC INFOMAPS Statement

Connects to the specified metadata server.

PROC INFOMAPS
METAPASS="*password*"
METAPORT=*port-number*
METAREPOSITORY="*repository-name*"
METASERVER="*address*"
METAUSER="*user-ID*"
<MAPPATH="*repository-path*">;

Required Arguments

METAPASS="*password*"
specifies the password that corresponds to the user ID that connects to the metadata server. For example, `metapass="My Password"` or `metapass="MyPassword"`. If the password is not all uppercase or does not contain a blank space (or spaces), enclosing the identifier in quotation marks is optional.

If this option is not specified, the value is obtained from the METAPASS= system option. See the METAPASS= system option in the *SAS Language Reference: Dictionary*.

Alias: PASSWORD= | PW=

METAPORT=*port-number*
specifies the TCP port that the metadata server is listening to for connections. For example, `metaport=8561`.

If this option is not specified, the value is obtained from the METAPORT= system option. See the METAPORT= system option in the *SAS Language Reference: Dictionary*.

Alias: PORT=

METAREPOSITORY="*repository-name*"

specifies a name that is assigned to a particular metadata repository to use on the metadata server. For example, `metarepository="myrepos"`. Because the repository name is not case-sensitive, you can omit the quotation marks if the name does not contain a blank space (or spaces).

If this option is not specified, the value is obtained from the METAREPOSITORY= system option. See the METAREPOSITORY= system option in the *SAS Language Reference: Dictionary*.

Alias: REPOSITORY= | REPOS= | REPNAME=

METASERVER="*address*"

specifies the network IP (Internet Protocol) address of the computer that hosts the metadata server. For example, `metaserver="myip.us.mycompany.com"`.

If this option is not specified, the value is obtained from the METASERVER= system option. See the METASERVER= system option in the *SAS Language Reference: Dictionary*.

Alias: SERVER= | HOST=

METAUSER="*user-ID*"

specifies the user ID to connect to the metadata server. For example, `metauser="myUserID"`. The user ID is not case-sensitive. If the user ID does not contain a blank space (or spaces) or special characters, enclosing the identifier in quotation marks is optional.

If this option is not specified, the value is obtained from the METAUSER= system option. See the METAUSER= system option in the *SAS Language Reference: Dictionary*.

Alias: USER= | USERID= | ID=

Restriction: In the metadata server, you must have at least one login definition that contains a user ID that corresponds to the user ID that you specify here. For information about login definitions, see the User Manager Help for logins in the SAS Management Console.

Restriction: If your metadata server runs in a Windows environment, then you must fully qualify the user ID by including the domain or machine name that you specified when your login object was created in the SAS Metadata Repository. For example: `metauser="Windows-domain-name\user-ID"`.

Option

MAPPATH="*repository-path*"

specifies the path to the location within the metadata repository of the information map to create, open, or delete. After the connection is made, the path is stored so that you do not need to specify it again on subsequent statements such as OPEN, SAVE, DELETE, or EXPORT. However, if do you specify a path on a subsequent statement, then that path overrides the stored path.

Alias: PATH=

Restriction: By default, SAS Web Report Studio looks for information maps in the directory `/BIP Tree/ReportStudio/Maps`. If you are creating an information map for use with SAS Web Report Studio, you must store it in that directory unless a system administrator has changed the directory to something other than the default.

Examples

```
proc infomaps metauser="myUserID"
              metapass="myPassword"
              metaserver="myip.us.mycompany.com"
              metaport=8561
              metarepository="Foundation";
```

DELETE INFOMAP Statement

Deletes an information map from the metadata repository.

DELETE INFOMAP *information-map-name*
<MAPPATH="*repository-path*">;

Required Arguments

information-map-name
specifies the name of the information map to delete.

Option

MAPPATH="*repository-path*"
specifies the path to the location within the metadata repository of the information map to delete. The path is required unless a path has been specified in the PROC INFOMAPS statement. The repository path from the DELETE statement overrides the path from the PROC INFOMAPS statement.

Examples

```
delete infomap "my testmap"
       mappath="/BIP Tree/ReportStudio/Maps";

delete infomap "myMap";
```

EXPORT Statement

Exports an information map in its XML representation.

EXPORT <INFOMAP *information-map-name*>
<FILE=*fileref* | "*physical-path*">
<MAPPATH="*repository-path*">;

Options

INFOMAP *information-map-name*
specifies the name of the information map to export. If the name does not contain a blank space (or spaces), or special characters, or is case sensitive, enclosing the identifier in quotation marks is optional. If you do not specify a name, the current active information map is exported.

FILE=*fileref* | "*physical-path*"
specifies an external file to which to export an XML representation of the information map. If the external file already exists, it is replaced.
If the FILE= option is not specified, the information is sent to the SAS log.

Requirement: If you use an external text editor to modify the XML file after it has been exported, then the editor must encode the file using the Unicode UTF-8 format in order for SAS Information Map Studio to import it correctly.

MAPPATH="*repository-path*"
specifies the path to the location within the metadata repository of the information map to export. The MAPPATH option is ignored if no information map name is specified.
If you do not specify a path, the default is the repository path specified in the OPEN INFOMAP statement, if a path is specified there. If not, the default is the repository path specified in the PROC INFOMAPS statement. Exporting fails if the repository path is not specified in the EXPORT statement or in an OPEN INFOMAP or PROC INFOMAPS statement.

Examples

```
/* Export an information map to a physical path. */
/* Note that the sample paths are operating system-specific. */
export infomap "my testmap" file="c:\test\test.xml"
                mappath="/BIP Tree/ReportStudio/Maps";

/* Export an information map to a fileref. */
filename xmlfile "c:\test\test.xml";
export infomap "my testmap" file=xmlfile
                mappath="/BIP Tree/ReportStudio/Maps";
```

IMPORT Statement

Imports an information map from an external XML file.

IMPORT FILE=*fileref* |"*physical-path*";

Required Arguments

FILE=*fileref* |"*physical-path*"
specifies the fileref or physical path of an XML file from which an information map is imported.

Requirement: If you use an external text editor to modify the XML file before importing it, then the editor must encode the file using the Unicode UTF-8 format for SAS Information Map Studio to import it correctly.

Details

After importing an information map, you must issue a SAVE statement to save it. If you specify a name on the SAVE statement, then that name overrides the name specified in the XML file. If you save it with the same name and in the same location as an existing information map, then the imported information map replaces the existing information map in the metadata repository.

The location where the imported information map is saved is determined according to the following order of precedence:

1 The MAPPATH specified on the SAVE statement
2 The MAPPATH specified on the OPEN INFOMAP statement
3 The MAPPATH specified on the PROC INFOMAPS statement

The IMPORT statement always opens a new information map. Any changes made to an open information map are lost if those changes are not saved before importing.

Examples

```
/* Create a new information map from an external file. */
import file="c:\test\test.xml";
save;
```

INSERT DATAITEM Statement

Inserts a data item for the specified column or expression into the information map that is currently open.

INSERT DATAITEM
COLUMN=*datasource-ID.column-name* | EXPRESSION="*expression-text*"
<AGGREGATION=*aggregate-function*>
<AGGREGATIONS_DROP_LIST=(*aggregate-function-list*)>
<CLASSIFICATION=CATEGORY | MEASURE>
<DESCRIPTION="*descriptive-text*">
<FOLDER=*folder-name* | "*folder-path*">
<FORMAT="*format-name*">
<ID="*dataitem-ID*">
<NAME="*name*">
<TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP>;

Required Arguments

COLUMN=*datasource-ID.column-name*

specifies a physical column. The *datasource-ID* is the ID of a data source in the current information map. It must match the ID of the table that contains the physical column, as shown in the following example:

```
insert datasource sasserver="SASMain"
                  table="Common".WORLDPOP2002
                  id="PopulationData";

insert dataitem column="PopulationData".Projected_Population_millions_;
```

The *column-name* is the SAS name of a column defined in the relational table associated with data source ID. The INFOMAPS procedure inserts a data item for this column into the information map.

Restriction: This option applies only to a relational data source.

Interaction: If you specify the COLUMN= option, then you cannot specify the EXPRESSION= option.

EXPRESSION="*expression-text*"

specifies the combination of data elements, literals, functions, and mathematical operators that are used to derive the value of a data item when the information map is used in a query.

Note: If you are using the Information Maps engine to access an information map containing character type data items created with the EXPRESSION= option, you should be aware of the EXPCOLUMNLEN= option of the LIBNAME statement. By default, the Information Maps engine sets the data length for columns of these data items to 32 characters. You can use the EXPCOLUMNLEN= option to change the default length. For more information on the EXPCOLUMNLEN= option, see "Other LIBNAME Statement Options for the Information Maps Engine" on page 41 and "EXPCOLUMNLEN= Data Set Option" on page 43. △

Requirement: If you specify the EXPRESSION= option, then you must also specify the TYPE= option.

Requirement: *Relational data:* Any reference to physical or business data in a relational table must be enclosed in double angle brackets <<...>>. Everything between double angle brackets is maintained just as it is; that is, case and blank spaces are maintained.

If you are referring to a physical column, then you must qualify the column with the source ID. For example, **`<<Transaction.Sales_Tax>>`**. If you are referring, in an expression, to a data item in the current information map, then you do not need to qualify the data item ID—though you can use the qualifier **`root`**, for example, **`<<root.MODEL_ID>>`**.

Requirement: *OLAP data:* Expressions for OLAP data items must resolve to a valid, one-dimensional MDX set. Use double angle brackets <<...>> to enclose references to an OLAP measure, OLAP dimension, OLAP hierarchy, or an OLAP level. Use single sets of square brackets [...] to enclose a reference to an OLAP member. For example:

```
<<Measures.new_business_value_sum>>,
<<campaigns>>,
<<campaigns.campaigns>>,
[campaigns].[All campaigns].[ADVT]
```

Interaction: If you specify the EXPRESSION= option, then you cannot specify the COLUMN= option.

Tip: If you are using INSERT DATAITEM to insert a non-calculated data item from a physical column, it is preferable for performance reasons to use the COLUMN= option instead of the EXPRESSION= option.

Options

AGGREGATION=*aggregate-function*
specifies the default aggregate function that the information map user sees.

Restriction: AGGREGATION applies only to a relational data item that is a measure.

Restriction: If you do not specify an aggregate function, it defaults to the SUM function. If SUM is not available, then it defaults to COUNT. If neither the COUNT nor SUM function is available, it defaults to the first of the available supported aggregation functions.

Restriction: If you create a measure data item and include an existing measure data item or a known aggregate in the expression of the new measure data item, then the only aggregate functions available to the new measure data item are **`InternalAggregation`** or **`InternalAggregationAdditive`**.

AGGREGATIONS_DROP_LIST=(*aggregate-function-list*)
removes one or more functions from the list of aggregate functions available to a data item.

Default: All available aggregate functions for the data item are selected.

Restriction: This option applies only to a relational data item that is a measure.

Requirement: Separate multiple aggregate functions with a blank space. For example:

```
AGGREGATIONS_DROP_LIST=(Freq FreqDistinct CSSDistinct)
```

Available aggregate functions are shown in the following table. For more information about these aggregate functions (except for **`InternalAggregation`** and **`InternalAggregationAdditive`**), see "Summarizing Data: Using Aggregate Functions" in the *SAS SQL Query Window User's Guide*, which you can find at **`support.sas.com/documentation/onlinedoc/91pdf/`**.

Table 2.1 Aggregate Functions

Function	Definition
AVG, MEAN	average or mean of values
AvgDistinct, MeanDistinct	average or mean of distinct values
COUNT, FREQ, N	number of non-missing values
CountDistinct, FreqDistinct, NDistinct	number of distinct non-missing values
CountPlusNMISS	number of distinct non-missing values plus the number of distinct missing values
CSS	corrected sum of squares
CSSDistinct	corrected sum of squares of distinct values

Function	Definition
CV	coefficient of variation (percent)
CVDistinct	coefficient of variation (percent) of distinct values
InternalAggregation	defined in an expression
InternalAggregationAdditive	defined in an expression (additive)
MAX	largest value
MIN	smallest value
NMISS	number of missing values
NMISSDistinct	number of distinct missing values
PRT	probability of a greater absolute value of Student's *t*
PRTDistinct	probability of a greater absolute value of Student's *t* of distinct values
RANGE	range of values
RANGEDistinct	range of distinct values
STD	standard deviation
STDDistinct	standard deviation of distinct values
STDERR	standard error of the mean
STDERRDistinct	standard error of the mean of distinct values
SUM	sum of values
SumDistinct	sum of distinct values
T	Student's *t* value for testing the hypothesis that the population mean is zero
TDistinct	Student's *t* value for testing the hypothesis that the population mean of distinct values is zero
USS	uncorrected sum of squares
USSDistinct	uncorrected sum of squares for distinct values
VAR	variance
VarDistinct	variance of distinct values

CLASSIFICATION=CATEGORY | MEASURE
specifies the usage type of the data item to be created. If you do not specify the CLASSIFICATION= option, the INFOMAPS procedure assigns a default classification based on:

- the contents of the expression if the EXPRESSION= option is used
- the data type of the physical column if the COLUMN= option is used

In general, if a data item is created from a physical column, then CATEGORY is the default classification unless the physical column is of type NUMERIC and is not a key. Data items inserted with the EXPRESSION= option also default to CATEGORY. However, if the expression contains an aggregation, the default classification is MEASURE instead.

DESCRIPTION="*descriptive-text*"
specifies the description for the data item, which can be viewed by the information map consumer.

Alias: DESC=

Restriction: The description must be enclosed in quotation marks. Also, note that when used in SAS programs, descriptions are limited to 256 characters.

FOLDER=*folder-name* | "*folder-path*"
specifies the folder in the information map into which to insert the data item. If the folder is in the root directory of the information map, then you can specify the folder by name. For example, `FOLDER=CUSTOMERS`. If the folder name does not contain a blank space (or spaces) or non-alphanumeric characters, enclosing the identifier in quotation marks is optional.

Restriction: If you do not enclose the folder name in quotation marks, the name that you specify is changed to uppercase and will match only a folder whose name is uppercase.

Restriction: If the folder is not in the root directory, then you must specify the path to the folder beginning with the initial slash and enclose the path in quotation marks. For example, `FOLDER="/CUSTOMERS/Europe"`.

Restriction: The following characters are not valid in a folder name:

- / \
- null characters
- non-blank nonprintable characters

Restriction: A name can contain blank spaces, but cannot consist only of blank spaces.

FORMAT="*format-name*"
specifies the SAS format of the data item.

If you do not specify a SAS format, the INFOMAPS procedure may set a default format for the data item based on the following factors:

- the classification of the data item
- whether there is a format defined in the physical resource referenced in the data item expression
- the actual data type of the physical resource referenced

Restriction: The format name must be enclosed in quotation marks.

Restriction: This argument applies only to relational data items and OLAP measures.

ID="*dataitem-ID*"
specifies the ID assigned to the data item being inserted. The ID is a value that you can use in an expression to uniquely identify the associated data item in the current information map. If you do not specify the ID= option, the INFOMAPS procedure generates an ID. The value that is generated for a data item depends on how the data item is inserted:

- If the NAME= option is specified, the data item name is used as the seed for generating the ID.
- If the NAME= option is not specified, how the ID is generated depends on whether the data item is inserted from a physical column or from the EXPRESSION= option.
  - If the data item is inserted from a physical column in one of the following ways:

    INSERT DATAITEM with the COLUMN= option specified

    INSERT DATASOURCE with either the _ALL_ or the COLUMNS= option specified then the ID is generated from either the SAS name or label of the physical column. The settings of the USE_LABELS, REPLACE_UNDERSCORES, and INIT_CAP options determine the exact value and casing of the ID.

- If the data item is inserted with the EXPRESSION= option, then the INFOMAPS procedure assigns a unique ID of the form

```
"DataItem"<number>
```

(where <number> is an internally maintained counter for ID generation. This counter is also used for generating IDs for other business data, including filters and data sources).

The INSERT DATAITEM statement prints a note displaying the ID of the data item if the ID has a different value from the data item name. You can use the LIST statement to view the IDs of all the data items in the current information map.

Restriction: Nulls and non-blank nonprintable characters are not valid in an ID. The following characters

```
. < > [ ] { } \ / ^ @ ~
```

will be replaced with an underscore (_).

Restriction: The first 32 characters of an ID must be unique. An ID that differs only by case from another ID in the current information map is not considered unique. If the ID specified is not unique, the INFOMAPS procedure terminates the data-item insertion and prints an error message in the SAS log.

Tip: An ID must be unique across an information map. You will receive an error message if you specify an ID that is the same as an existing ID (data item, data source, filter, or other).

NAME="*name*"

specifies the name assigned to the data item in the information map. A name is optional, descriptive text that makes it easier for business users to understand what the data is about. A data item's name is for display purposes only—you use a data item's ID to refer to it in code rather than its name. If you do not specify a name, the name defaults to the column name or column label (based on the setting of the USE_LABELS option from the OPEN INFOMAP statement) if the COLUMN= option is used; otherwise, the INFOMAPS procedure provides a default name.

If you specify a name but do not specify an ID, the INFOMAPS procedure uses the name to generate a unique ID.

Note: In the XML format of an information map, the data item name is encoded with a <label> tag. △

Restriction: There is no limit on the length of the name of a relational data item. OLAP data item names can be at most 30 characters.

Restriction: A name can contain blank spaces, but cannot consist only of blank spaces. Nulls and non-blank nonprintable characters are not valid characters in a name. A name can contain the following special characters:

```
. < > [ ] { } \ / ^ @ ~
```

but they will be replaced with an underscore (_) in the ID that is generated from the name. Square brackets [...] are not valid in an OLAP data item name.

TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP

specifies the data type of the data item's expression.

Restriction: For OLAP data, the only valid types are NUMERIC and CHARACTER.

Interaction: If you specify the EXPRESSION= option, then you must specify the TYPE= option. If you specify the COLUMN= option, then you do not have to specify the TYPE= option. In this case, the INFOMAPS procedure derives the type from the type of the column.

Examples

```
/* Use the COLUMN= option to insert a data item for a physical column. */
insert dataitem column="TRANSACTION".Sales_Amount
  name="Average Sale"
  desc="Average amount of sale per transaction"
  id="Average_Sale"
  folder="Measures" classification=measure format="Comma8.2"
  aggregation=AvgDistinct;

/* Use the EXPRESSION= option to insert a calculated data item. */
insert dataitem
  expression="<<TRANSACTION.Sales_Amount>>+<<TRANSACTION.Sales_Tax>>"
  name="Total Sales Amount Plus Tax"
  id="Total_with_Tax"
  type=numeric
  desc="Total with Tax" classification=measure format="Comma8.2";

/* Insert a data item for an OLAP member. */
insert dataitem
  expression="<<customer_age.customer_age.age_group_1>>.MEMBERS"
  name="Customer_Age_Group_1"
  type=char
  desc="First age group" classification=category;

/* Insert a data item for an OLAP hierarchy. */
insert dataitem
  expression="<<customer_age.customer_age>>.Hierarchy"
  name="Customer Age"
  id="Customer_Age"
  type=char;
```

INSERT DATASOURCE Statement

Makes available to the current information map the data from either a table or cube. The information map must have been opened with an OPEN INFOMAP statement.

INSERT DATASOURCE
SASSERVER | SERVER=*server-name*
TABLE=*library.table* <COLUMNS=(*column–1 column–2 ... column-n*) | _ALL_ >
<ID=*datasource-ID*>;

or:

INSERT DATASOURCE
SASSERVER | SERVER=*server-name*
CUBE=<*schema.*>*cube* <_ALL_>
<ID=*datasource-ID*>;

Required Arguments

SASSERVER | SERVER=*server-name*
identifies the SAS server. The server can be either a SAS application server for relational data (SAS libraries) or a SAS OLAP Server for cube data. The type of server being accessed is identified by the TABLE= option or the CUBE= option.

TABLE=*library.table*
identifies a relational table as a data source for the current information map.

A table must be:

- □ registered in the currently connected metadata repository or a parent repository of the current metadata repository.
- □ associated with a SAS library that is registered in the SAS application server specified by the SASSERVER= option.

In order for an information map to use a table, the table must have a unique name in its SAS library (for a SAS table) or database schema (for a table from a different DBMS) in the metadata repository. If multiple tables in a SAS library or database schema have the same name, then you must perform one of the following tasks before you can use any of the tables with an information map:

- □ From either SAS Data Integration Studio or the Data Library Manager in SAS Management Console, you can rename a table by changing the value of the `Name` field on the `General` tab in the properties window for the table.
- □ From SAS Data Integration Studio, delete the duplicate tables.

Restriction: Although you can access either relational data or cube data, you cannot access both within the same information map.

Restriction: You can use multiple INSERT DATASOURCE statements to add multiple relational tables to the same information map. However, when accessing multiple tables, all tables must be accessed from the same SAS Workspace Server.

CUBE=<*schema.*>*cube*
identifies an OLAP cube as a data source for the current information map.

A cube must be:

- □ registered in the currently connected metadata repository or a parent repository of the current metadata repository.
- □ associated with a schema that is registered in the SAS OLAP Server specified by the SASSERVER= option.

Restriction: Although you can access either relational data or cube data, you cannot access both within the same information map.

Restriction: You can insert only one OLAP cube into an information map.

Options

ALL
specifies to insert a data item for each physical column or hierarchy as defined in the specified table or cube.

Interaction: If you specify the _ALL_ option, then you cannot specify the COLUMNS= option.

COLUMNS=(*column-1 column-2 ... column-n*)

specifies one or more physical column names as defined in the specified table. The INFOMAPS procedure inserts a data item into the information map for each of these named columns.

The column list can be a single SAS column name or a list of SAS column names separated by at least one blank space and enclosed in parentheses.

Restriction: This option applies only to a relational data source.

Requirement: If you specify the COLUMNS= option, then you must specify it immediately after the TABLE= option.

Interaction: If you specify the COLUMNS= option, then you cannot specify the _ALL_ option.

ID="*datasource-ID*"

specifies the ID assigned to the data source. The ID is a value that you can use in an expression to uniquely identify the associated data source in the current information map.

If you do not specify the ID= option, the INFOMAPS procedure generates an ID for the data source based on the specified table or cube name. If the generated ID is different from the table or cube name, then the INFOMAPS procedure prints a note in the SAS log with the generated ID. You can use the LIST statement to display data source IDs.

Restriction: Nulls and non-blank nonprintable characters are not valid in an ID. The following characters

```
. < > [ ] { } \ / ^ @ ~
```

will be replaced with an underscore (_).

Tip: An ID must be unique across an information map. You will receive an error message if you specify an ID that is the same as an existing ID (data item, data source, filter, or other).

Details

The inserted data sources are logical representations of the data that you can use to query the physical data. If you are familiar with SAS Information Map Studio, using an INSERT DATASOURCE statement in a program is equivalent to choosing **`Insert Table`** or **`Insert Cube`** from the menu bar in SAS Information Map Studio.

You can insert multiple tables as data sources into an information map. To refer to a table data source in an expression, you must use its ID. By default, the ID of a table data source is the same as the table name. If a table data source already exists in the current information map that has the same name as the table that you are attempting to insert, then the INFOMAPS procedure renames the new table to ensure a unique data source name and ID. To view a list of all the data sources in the current information map, use the LIST DATASOURCES statement. Note that even though the table data source name and its ID have the same value by default, you can use the ID= option to specify a different ID, or you can use SAS Information Map Studio to change the table name after the information map is saved.

Examples

```
/* Insert all the columns from a relational data source. */
insert datasource sasserver="SASMain"
                  table="Basic Data".CUSTOMER _ALL_ ;
```

```
/* Insert only 3 columns from a relational data source. */
insert datasource sasserver="SASMain"
                  table="OrionTables".CUSTOMER_DIM
                  columns=("Customer_id" "Customer_name" "Customer_age") ;

/* Insert an OLAP data source. */
insert datasource sasserver="SASMain"
                  cube="SAS Main - OLAP Schema".class
                  id="Sample_Data";
```

INSERT FILTER Statement

Inserts a filter into the current information map. A filter provides criteria for subsetting a result set. For relational databases, a filter is a WHERE clause.

INSERT FILTER *filter-name*
CONDITION="*conditional-expression*"
<DESCRIPTION="*descriptive-text*">
<FOLDER=*folder-name* | "*folder-path*">;

Required Arguments

filter-name
specifies the name of a filter to insert into the current information map.

Restriction: Nulls and non-blank nonprintable characters are not valid characters for a filter.

CONDITION="*conditional-expression*"
specifies a conditional expression that is used to filter the data.

Requirement: The conditional expression must be enclosed in quotation marks and must immediately follow the filter name.

Requirement: *Relational data:* Any reference to physical or business data in a relational table must be enclosed in double angle brackets <<...>>. Everything between double angle brackets is maintained just as it is; that is, case and blanks are maintained.

If you are referring to a physical column, then you must qualify the column with the source ID. For example, **`<<Transaction.Sales_Tax>>`**. If you are referring, in an expression, to a data item in the current information map, then you do not need to qualify the data item ID—though you can use the qualifier **`root`**, for example, **`<<root.MODEL_ID>>`**.

Requirement: *OLAP data:* Expressions for OLAP data items must resolve to a valid, one-dimensional MDX set. Use double angle brackets <<...>> to enclose references to an OLAP measure, OLAP dimension, OLAP hierarchy, or an OLAP level. Use single sets of square brackets [...] to enclose a reference to an OLAP member.

Options

DESCRIPTION="*descriptive-text*"
specifies the description of the filter to be inserted.

Alias: DESC=

FOLDER=*folder-name* | "*folder-path*"
specifies the folder in the information map into which to insert the filter. If the folder is in the root directory of the information map, then you can specify the folder by name. For example, **`FOLDER=CUSTOMERS`**. If the folder name does not contain a blank space (or spaces) or non-alphanumeric characters, enclosing the identifier in quotation marks is optional.

Restriction: If you do not enclose the folder name in quotation marks, the name that you specify is changed to uppercase and will match only a folder whose name is uppercase.

Restriction: If the folder is not in the root directory, then you must specify the path to the folder beginning with the initial slash and enclose the path in quotation marks. For example, **`FOLDER="/CUSTOMERS/Europe"`**.

Restriction: The following characters are not valid in a folder name:

- / \
- null characters
- non-blank nonprintable characters

Restriction: A name can contain blank spaces, but cannot consist only of blank spaces.

Examples

```
/* Insert a relational table filter. */
insert filter MyFilter condition='<<Geography.Country>> = "CANADA" ';

/* Insert an MDX filter. */
insert filter dates1
condition="<<Dates_FirstChild>> <>
         [cust_dates].[All cust_dates].[1996].[1996/06].[24JUN96]";

/* Insert an MDX filter. */
insert filter dates2
condition="<<Dates_Dates>>=[cust_dates].[All cust_dates].[1998].[1998/02],
         [cust_dates].[All cust_dates].[1998].[1998/02].[03FEB98]";
```

INSERT FOLDER Statement

Inserts a map folder into the current information map.

INSERT FOLDER *folder-name*
 <PARENT=*parent-folder-name*>
 <DESCRIPTION="*descriptive-text*">;

Required Arguments

folder-name
specifies the name of the map folder to insert into the current information map.

Restriction: If you do not enclose the folder name in quotation marks, the name that you specify is changed to uppercase in the information map.

Tip: When referring to the folder, remember that case is important. A good practice is to always enclose folder names in quotation marks so that the folder name in your code matches the folder name in the information map.

Options

PARENT=*parent-folder-name*
specifies the parent folder of the folder that you are inserting into the information map. By specifying the parent folder, you specify where in the information map to insert the folder. If the parent folder is in the root directory of the information map, then you can specify the parent folder by name. For example, **`PARENT=CUSTOMERS`**. If the parent folder name does not contain a blank space (or spaces) or non-alphanumeric characters, enclosing the identifier in quotation marks is optional.

Restriction: If you do not enclose the parent folder name in quotation marks, the name that you specify is changed to uppercase and will match only a folder whose name is uppercase.

Restriction: If the parent folder is not in the root directory, then you must specify the path to the folder beginning with the initial slash and enclose the path in quotation marks. For example, **`PARENT="/CUSTOMERS/Europe"`**.

Restriction: The following characters are not valid in a parent folder name:

- / \
- null characters
- non-blank nonprintable characters

Restriction: A parent folder name can contain blank spaces, but cannot consist only of blank spaces.

DESCRIPTION="*desc*"
specifies the description of the folder to be created.

Alias: DESC=

Examples

```
insert folder "measures";
insert folder "subMeasures" parent="measures";
insert folder "subsubMeasures" parent="/measures/subMeasures";
```

INSERT RELATIONSHIP Statement

Inserts a join into the current information map. RELATION is an alias for RELATIONSHIP.

INSERT RELATIONSHIP *left-table* **INNER | LEFT | RIGHT | FULL**
  JOIN *right-table* **ON** "*conditional-expression*";

Required Arguments

left-table
specifies the data source ID of the first table in the relationship.

right-table
specifies the data source ID of the second table in the relationship.

"*conditional-expression*"
specifies the columns to be joined to create a single relationship between two tables.

Requirement: The conditional expression must be enclosed in quotation marks.

Requirement: The columns referenced in the conditional expression must be qualified with the associated data source ID and must be enclosed in double angle brackets <<...>>.

Details

INSERT RELATIONSHIP applies only to relational tables. If there already exists a join between the specified tables, the new join replaces the old one.

When specifying a table, you must specify the data source ID associated with the table. IDs are case sensitive. You can use the LIST DATASOURCES statement to see the IDs of data sources in your information map.

Examples

```
insert relationship "CUSTOMER" inner join "TRANSACTION"
       on "(<<CUSTOMER.Cust_ID>>=<<TRANSACTION.Cust_ID>>)";
```

LIST Statement

Lists the definitions of business data in the current information map. The definitions are printed to the SAS log or to the computer console.

LIST <DATAITEMS | DATASOURCES | FILTERS | _ALL_> ;

Options

DATAITEMS
lists the properties of all the data items defined in the current information map. The properties include the name, ID, folder path, description, expression text, and expression type of each data item.

DATASOURCES
lists the properties of all the data sources defined in the current information map. The properties include data source (*library.physical-table*), data source ID, table or cube name, and description.

FILTERS
lists the properties of all the filters defined in the current information map. The properties include the name, folder path, description, and the conditional expression text of each filter.

ALL
lists the properties of all the data items, filters, and data sources defined in the current information map.

Default: _ALL_ is the default if you do not specify an option.

Examples

```
open infomap "MAILORDERCUBE";
list;
run;
```

Output 2.1 Output from the LIST Statement That Is Displayed in the Log Window

```
15   open infomap "MAILORDERCUBE";
16   list;
Data source: SASMain - OLAP Schema.MAILORDERCUBE
ID: MAILORDERCUBE
Table/Cube name: MAILORDERCUBE
Description:

Data item name: COST_N
ID: COST_N
Folder: /
Description: OLAP measure COST_N
Expression: <<Measures.COST_N>>
Expression type: NUMERIC

Data item name: GEOGRAPHIC
ID: GEOGRAPHIC
Folder: /
Description: OLAP hierarchy GEOGRAPHIC
Expression: <<GEOGRAPHIC.GEOGRAPHIC>>
Expression type: CHARACTER

Data item name: PRODUCTLINE
ID: PRODUCTLINE
Folder: /
Description: OLAP hierarchy PRODUCTLINE
Expression: <<PRODUCTLINE.PRODUCTLINE>>
Expression type: CHARACTER

Data item name: SALES_COST
ID: SALES_COST
Folder: /
Description: OLAP measure SALES_COST
Expression: <<Measures.SALES_COST>>
Expression type: NUMERIC

Data item name: TIME
ID: TIME
Folder: /
Description: OLAP hierarchy TIME
Expression: <<TIME.TIME>>
Expression type: CHARACTER

17   run;
```

OPEN INFOMAP Statement

Opens the named information map, or creates the information map if it does not exist.

OPEN INFOMAP *information-map-name*
    <CREATE_TARGET_FOLDER=YES | NO>
    <INIT_CAP=YES | NO>
    <MAPPATH="*repository-path*">
    <REPLACE_UNDERSCORES=YES | NO>
    <USE_LABELS=YES | NO>;

Required Arguments

information-map-name
specifies the name of the information map to open or create.

Restriction: Information map names can be at most 60 characters. The name is truncated to 60 characters if it exceeds that limit. However, note that SAS names are restricted to 32 characters.

Restriction: The following characters are not valid:

- < > [ ]{ } \ / ^ @ ~
- null characters
- non-blank nonprintable characters

Restriction: A name can contain blank spaces, but cannot consist only of blank spaces or begin with a space.

Options

CREATE_TARGET_FOLDER=YES | NO
specifies whether to automatically create a folder. Specifying YES automatically creates a folder when you insert data items subsequently using an INSERT DATASOURCE statement with the _ALL_ option specified. The name of the folder is the name of the table specified in the INSERT DATASOURCE statement. The data items that are inserted as a result of the INSERT DATASOURCE statement are inserted into the folder that is created automatically.

Default: YES

Featured in: Example 2 on page 27.

INIT_CAP=YES | NO
specifies whether to capitalize the first letter of each word in the data item name. Specifying YES capitalizes the first letter of each word in the names of data items that you insert subsequently using one or more of the following statements:

- INSERT DATASOURCE with either the _ALL_ or the COLUMNS= option specified.
- INSERT DATAITEM with the COLUMN= option specified.

Default: YES

Tip: When you specify INIT_CAP=YES, the option replaces multiple consecutive blank spaces within a data item name with a single blank space, and it removes trailing blank spaces.

MAPPATH="*repository-path*"
specifies the path to the location within the metadata repository of the information map to open or create. The path is required unless a path has been specified in the PROC INFOMAPS statement. The repository path from the OPEN INFOMAP statement overrides the path from the PROC INFOMAPS statement.

REPLACE_UNDERSCORES=YES | NO
specifies whether to replace each underscore (_) character in the data item name with a blank space. Specifying YES replaces underscores in the names of data items that you insert subsequently using one or more of the following statements:

- INSERT DATASOURCE with either the _ALL_ or the COLUMNS= option specified.
- INSERT DATAITEM with the COLUMN= option specified.

Default: YES

USE_LABELS=YES | NO
specifies whether to create the data item name using the column label (if available) instead of the column name. Specifying YES uses the column label instead of the column name for data items that you insert subsequently using one or more of the following statements:

- INSERT DATASOURCE with either the _ALL_ or the COLUMNS= option specified.
- INSERT DATAITEM with the COLUMN= option specified.

Default: YES

Restriction: This option applies only to a relational data source.

Details

When you open an existing information map, the INFOMAPS procedure creates a copy of it in memory that can be modified. When you open an information map that does not yet exist, the INFOMAPS procedure allocates space in memory for its creation. In either case, you must save the information map with a SAVE statement for the in-memory copy to be written to the metadata repository.

Only one information map can be open at a time. If you submit one OPEN INFOMAP statement, you must save the open information map with a SAVE statement before submitting another OPEN INFOMAP or IMPORT statement. If you do not save the in-memory copy, it is not written to the metadata repository and is simply lost.

Examples

```
open infomap "my testmap"
     mappath="/BIP Tree/ReportStudio/Maps";
```

SAVE Statement

Saves the current information map.

SAVE <INFOMAP *information-map-name*>
<MAPPATH="*repository-path*">;

Options

INFOMAP *information-map-name*
specifies the name to use for saving the current information map. If the information map name does not contain a blank space (or spaces), special characters, or if you do not want to retain the case, enclosing the identifier in quotation marks is optional.

Default: If you do not specify a name, the default is the name of the current information map.

MAPPATH="*repository-path*"
specifies the path to the location within the metadata repository where the information map is to be saved.

Default: If you do not specify a path, the default is the repository path specified in the previous OPEN INFOMAP statement, if a path is specified there. If not, the default is the path specified in the previous PROC INFOMAPS statement.

Requirement: Enclose the path in quotation marks.

Examples

```
save infomap "myMap";
save mappath="/BIP Tree/ReportStudio/Maps";
save infomap "myMap" mappath="/BIP Tree/ReportStudio/Maps";
```

Examples: INFOMAPS Procedure

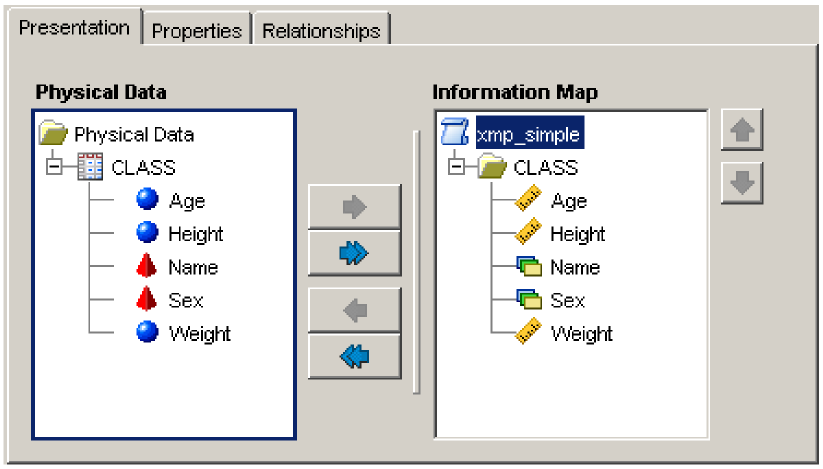
Example 1: Creating a Basic Information Map

The following example shows you how to use the INFOMAPS procedure to create an information map. Note that the program begins by attempting to delete the very information map that is being created. This ensures that the program does not create duplicate data items in an already existing information map. If the information map does not exit, the INFOMAPS procedure issues a warning message.

```
proc infomaps metauser="myUserID"
              metapass="myPassword"
              metaserver="myip.us.mycompany.com"
              metaport=8561
              metarepository="Foundation";

/* Delete the information map to avoid duplicate data items. */
delete infomap "xmp_simple" mappath="/BIP Tree/ReportStudio/Maps";

/* Open a new information map.  The path specified is where, by */
/* default, it will be saved when a SAVE statement is issued.   */
/* The information map exists only in memory and not in the     */
/* metadata repository until the SAVE statement is issued.      */
```

```
open infomap "xmp_simple"
             mappath="/BIP Tree/ReportStudio/Maps";

/* Note that an OPEN INFOMAP statement must precede the */
/* INSERT DATASOURCE statement.                         */

/* Make the specified table on the specified server accessible. */
insert datasource sasserver="SASMain"
                  table="Sample Data".CLASS _all_ ;

/* If no name is specified on the SAVE statement, then the information */
/* map that is currently open is saved with the name with which it was */
/* opened.  If no path is specified, then it is saved in the path      */
/* specified in the  OPEN statement or, secondarily, it is saved in    */
/* the path that is specified in the PROC INFOMAPS statement.          */
save;
run;
```

The following window shows the resulting information map opened in SAS Information Map Studio. Note that the folder CLASS was created automatically because the default is YES for the CREATE_TARGET_FOLDER option of the OPEN INFOMAP statement.



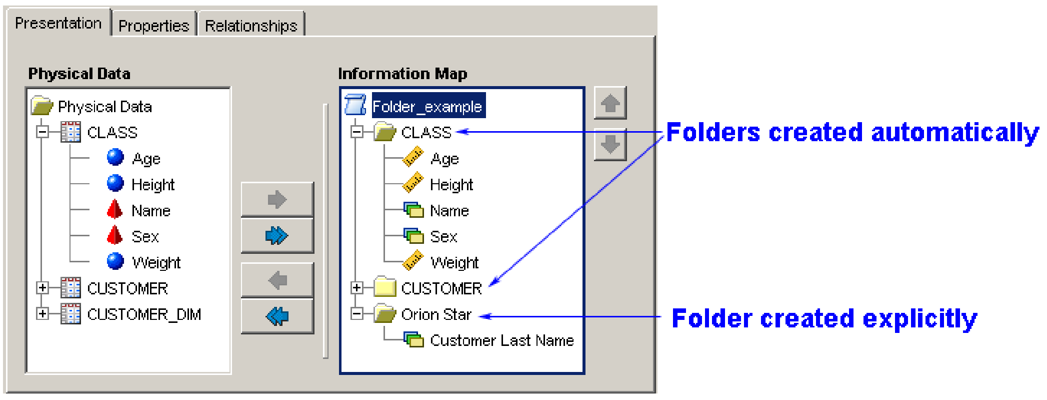
Example 2: Using Folders in an Information Map

The following example shows:

- The automatic creation of folders inside an information map. By default, a folder is created automatically when the _ALL_ option is specified on an INSERT DATASOURCE statement. In this example, two folders are created automatically to contain all the data items in two separate tables.
- The explicit creation of a folder, with the INSERT FOLDER statement, and the insertion of a data item into the folder with an INSERT DATAITEM statement.

```
proc infomaps metauser="myUserID"
              metapass="myPassword"
              metaserver="myip.us.mycompany.com"
              metaport=8561
              metarepository="Foundation";

/* By default, CREATE_TARGET_FOLDER creates a folder automatically for   */
/* every subsequent INSERT DATASOURCE statement when _ALL_ is specified. */
/* Specify CREATE_TARGET_FOLDER=NO if you do not want a folder created.  */
open infomap "Folder_example"
              mappath="/BIP Tree/ReportStudio/Maps"

/* A folder named "CLASS" will be created for all items in this table. */
insert datasource sasserver="SASMain"
                  table="Sample Data".CLASS _all_;

/* A folder named "CUSTOMER" will be created for all items in this table. */
insert datasource server="SASMain"
                  table="Basic Data".CUSTOMER _all_;

/* Make the table "Orion Star".CUSTOMER_DIM accessible to the   */
/* information map. No folder is created automatically because  */
/* _ALL_ is not specified with the INSERT DATASOURCE statement. */
insert datasource sasserver="SASMain"
                  table="Orion Star".CUSTOMER_DIM;

/* Create a folder named "Orion Star". */
insert folder "Orion Star";

/* Insert the data item "Customer Last Name" into the folder */
/* in the information map.                                    */
insert dataitem
       column="CUSTOMER_DIM".Customer_LastName
       folder="Orion Star";
save;
run;
```

The following window shows the resulting information map opened in SAS Information Map Studio.

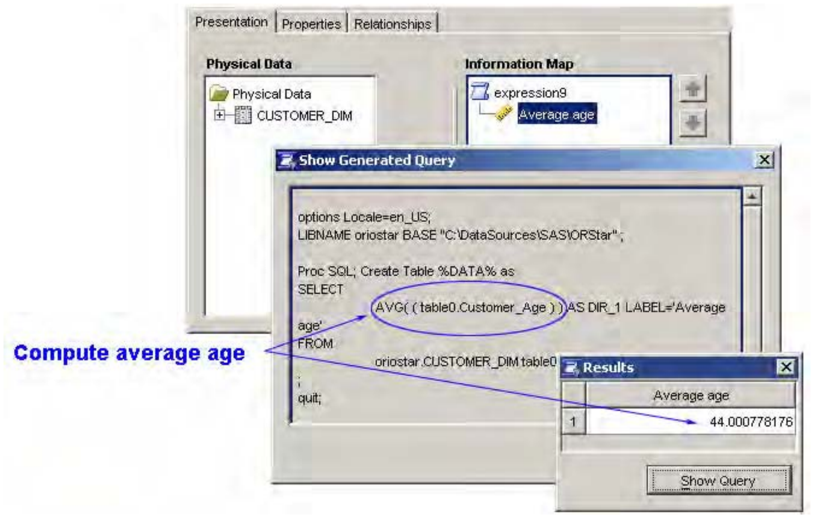


Example 3: Aggregating a Data Item

The following example shows the aggregation of data item values using the AGGREGATION= option on the INSERT DATAITEM statement.

```
proc infomaps metauser="myUserID"
              metapass="myPassword"
              metaserver="myip.us.mycompany.com"
              metaport=8561
              metarepository="Foundation";

open infomap "expression9"
              mappath="/BIP Tree/ReportStudio/Maps";

/* Make the table "Orion Star".CUSTOMER_DIM accessible */
/* to the information map.                             */
insert datasource sasserver="SASMain"
                  table="Orion Star".CUSTOMER_DIM;

/* Specify the aggregation function using the AGGREGATION= option. */
insert dataitem
      column="CUSTOMER_DIM".Customer_Age
      classification=measure
      aggregation=avg;

save;
run;
```

The following window shows the results of running a query in SAS Information Map Studio using the information map that was created with the INFOMAPS procedure. You can see that the query generated from the information map calculates an average, which is displayed in the Results window.



```
options Locale=en_US;
LIBNAME oriostar BASE "C:\DataSources\SAS\ORStar";

Proc SQL; Create Table %DATA% as
SELECT
        AVG( ( table0.Customer_Age ) ) AS DIR_1 LABEL='Average
age'
FROM
        oriostar.CUSTOMER_DIM table0
;
quit;
```

	Average age
1	44.000778176

CHAPTER

3

Using the SAS Information Maps LIBNAME Engine

What Does the Information Maps Engine Do?

An information map is a collection of data items and filters that describe and provide a view of physical data that business users understand. Information maps are defined in specific metadata repositories. The Information Maps engine enables you to retrieve data that is described by an information map. The engine provides a read-only way to access data generated from an information map and to bring it into a SAS session. Once you retrieve the data, you can run almost any SAS procedure against it.

Note that the Information Maps engine is only able to read information maps, it cannot write to or update them; nor can it modify the underlying data. If you want to add data items to an existing information map, you can use the INFOMAPS procedure. For more information, see Chapter 2, “The INFOMAPS Procedure,” on page 3. If you have SAS Information Map Studio, you can use that client application to interactively create or update information maps.

Understanding How the Information Maps Engine Works

An *engine* is a component of SAS software that reads from or writes to a file. Each engine enables SAS to access files that are in a particular format. There are several types of SAS engines.

The Information Maps engine works like other SAS data access engines. That is, you execute a LIBNAME statement to assign a libref and to specify an engine. You then use that libref throughout the SAS session where a libref is valid.

However, instead of the libref being associated with the physical location of a SAS data library, the libref is associated with a set of information maps. The information maps contain metadata that the engine uses to provide data access to users.

The following example shows a LIBNAME statement for the Information Maps engine and the output you see when you execute the statement:

```
libname mylib infomaps metauser=myUserID
                       metapass=myPassword
                       metaserver=myip.us.mycompany.com
                       metaport=8561
                       mappath="/BIP Tree/ReportStudio/Maps"
                       metarepository=Foundation;
```

Output 3.1 Output from the LIBNAME Statement That Is Displayed in the Log Window

```
1    libname mylib infomaps metauser=myUserID
2                           metapass=XXXXXXXXXX
3                           metaserver=myip.us.mycompany.com
4                           metaport=8561
5                           mappath="/BIP Tree/ReportStudio/Maps"
6                           metarepository=Foundation;
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /BIP Tree/ReportStudio/Maps
```

The DATASETS procedure can be used to display a list of available information maps.

Note: The list of available information maps will include only those that are supported by the engine. For example, there might be OLAP-based information maps available in the MAPPATH location. However, these information maps are not supported by the Information Maps engine, so they will not be displayed by the DATASETS procedure. △

The CONTENTS procedure can be used to view the data items and filters in an information map. The PRINT procedure can be used to print all of the data that the information map contains. If the map contains filters, they can be used to restrict the returned data. Here is an example:

```
/* Run the Information Maps engine to retrieve the data. */
libname mylib infomaps metauser=myUserID
                       metapass=myPassword
                       metaport=8561
                       metaserver="myip.us.mycompany.com"
                       metarepository="Foundation"
                       mappath="/BIP Tree/ReportStudio/Maps";

/* Display a list of available information maps. */
proc datasets lib=mylib;
run;

/* Allow mixed-case letters and blank spaces in information map names. */
option validvarname=any;

/* View the data items, including any filters, in the information map. */
proc contents data=mylib.InfoMaps_Demo;
run;

/* Print 5 observations from the data that the information map references. */
proc print data=mylib.InfoMaps_Demo
     (obs=5 filter=('Status is Current'n and 'Cary HQ'n));
run;
```

Output 3.2 Output from the DATASETS, CONTENTS, and PRINT Procedures That Is Displayed in the Log Window

```
1    /* Run the Information Maps engine to retrieve the data. */
2    libname mylib infomaps metauser=myUserID
3                           metapass=XXXXXXXXXX
4                           metaport=8561
5                           metaserver="myip.us.mycompany.com"
6                           metarepository="Foundation"
7                           mappath="/BIP Tree/ReportStudio/Maps";
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /BIP Tree/ReportStudio/Maps
8
9    /* Display a list of available information maps. */
10   proc datasets lib=mylib;
                                           Directory

                           Libref         MYLIB
                           Engine         INFOMAPS
                           Physical Name  /BIP Tree/ReportStudio/Maps


                                                       Member
                                    #  Name            Type

                                    1  InfoMaps_Demo   DATA
11
12   run;
13
14   /* Allow mixed-case letters and blank spaces in information map names. */
15   option validvarname=any;
WARNING: Only Base procedures and SAS/STAT procedures have been tested for use with
         VALIDVARNAME=ANY. Other use of this option is considered experimental and may cause
         undetected errors.
16
17   /* View the data items, including any filters, in the information map. */

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           13.71 seconds
      cpu time            0.86 seconds

18   proc contents data=mylib.InfoMaps_Demo;
19   run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           4.03 seconds
      cpu time            0.11 seconds


20
21   /* Print 5 observations from the data that the information map references. */
22   proc print data=mylib.InfoMaps_Demo
23   (obs=5 filter=('Status is Current'n and 'Cary HQ'n));
24   run;

NOTE: There were 5 observations read from the data set MYLIB.InfoMaps_Demo.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           12.21 seconds
      cpu time            0.71 seconds
```

Output 3.3 Output from the CONTENTS and PRINT Procedures That Is Displayed in the Output Window, Part 1 of 2

```
                                       The CONTENTS Procedure

             Data Set Name        MYLIB.InfoMaps_Demo      Observations          .
             Member Type          DATA                     Variables             9
             Engine               INFOMAPS                 Indexes               0
             Created              .                        Observation Length    0
             Last Modified        .                        Deleted Observations  0
             Protection                                    Compressed            NO
             Data Set Type                                 Sorted                NO
             Label                                         Filters               4
             Data Representation  Default
             Encoding             Default


                             Alphabetic List of Variables and Attributes

  # Variable              Type Len Format    Label

  7 Annual Salary         Num    8 DOLLAR12. /Salary Info/Annual Salary; Physical column SALARY
  5 Dept_code             Char  32           /Department Code;
  3 Division              Char  40           /Division; Physical column DIVISION
  9 Enddate               Num    8 DATE9.    /Salary Info/Enddate; Physical column ENDDATE
  4 Identification Number Num    8 SSN11.    /Identification Number; Physical column IDNUM
  1 Jobcode               Char   8           /Jobcode; Physical column JOBCODE
  2 Location              Char   8           /Location; Physical column LOCATION
  8 Monthly Salary        Num    8 DOLLAR12. /Salary Info/Monthly Salary;
  6 Title                 Char  20 $F20.     /Title; Physical column TITLE


                              Information Maps

                                Filter
FilterName                      Type       FilterDesc

Status is Current               Unp
Education and Publications      Unp        Employees in Education and Publications
Host Systems Development        Unp        Employees in Host Systems Development
Cary HQ                         Unp        Located in Cary, North Carolina HQ
```

Note: In the output from the CONTENTS procedure, the Label column gives you the folder path, the data item name, and the data item description. You can use this information to determine what each variable points to in the information map. △

Output 3.4 Output from the CONTENTS and PRINT Procedures That Is Displayed in the Output Window, Part 2 of 2

```
                              The SAS System

                                                      Identification       Dept_
         Obs    Jobcode    Location    Division                Number           Code

           1    ACT001       Cary      FINANCE                 333-44-5555         ACT
           2    APP001       Cary      SOFTWARE DEVELOPMENT    733-86-6750         APP
           3    APP002       Cary      SOFTWARE DEVELOPMENT    506-08-3698         APP
           4    APP002       Cary      SOFTWARE DEVELOPMENT    716-63-6663         APP
           5    APP002       Cary      SOFTWARE DEVELOPMENT    763-06-5513         APP

                                         Annual        Monthly
         Obs    Title                    Salary         Salary      Enddate

           1    TAX ACCOUNTANT I         $37,000         $3,083           .
           2    EXEC ASST II             $43,500         $3,625           .
           3    MEMBER OF APPL STAFF     $24,000         $2,000           .
           4    MEMBER OF APPL STAFF     $29,000         $2,417           .
           5    MEMBER OF APPL STAFF     $30,000         $2,500           .
```

For information about improving the performance of the Information Maps engine, see "Best Practices for Using the Information Maps Engine" on page 49.

Advantages of Using the Information Maps Engine

Using the Information Maps engine provides the following advantages:

- The engine is the only way to access data generated from an information map for Base SAS software.
- The engine provides a single point of access to many information maps.
- The engine enables you to take advantage of information maps, which provide you with the benefits described in "Why Are SAS Information Maps Important?" on page 2.

Operating Systems Supported by the Information Maps Engine

The following operating systems are supported:

- Windows (32–bit)
- Solaris (64–bit)
- UNIX (HP-UX, AIX 64–bit)

What Is Required to Use the Information Maps Engine?

To use the Information Maps engine, the following are required:

- access to the SAS Open Metadata Architecture
- access to the metadata repository that contains the metadata definition for the physical data and information maps
- information maps that are defined in a metadata repository
- access to the SAS Workspace Server where the physical data is located

What Is Supported?

The Information Maps engine is only able to read information maps and their data sources. If you want to update an information map directly, you can use the INFOMAPS procedure or SAS Information Map Studio.

The engine supports accessing metadata in a metadata repository to process the information map. Using the engine and SAS code, you can:

- □ read data that is generated from an information map (input processing)
- □ create a new data set by using an information map (output processing)

  Note: The new data set is created in Base SAS software, not on the data server. △

The Information Maps engine does not support the following:

- □ The engine does not pass WHERE clauses to the SAS server for processing. Therefore, all the data generated from the information map is passed back to the SAS client and the SAS client applies the WHERE clause to restrict the data for the result set.

  The Information Maps engine performance can be degraded when a large number of observations for WHERE clauses have to be processed by SAS. Note that information map filters are available for restricting the query. (A filter contains criteria for subsetting data in an information map.)
- □ The engine does not sort data in the result set for BY-group processing. BY-group processing requires that the result set be sorted; however, the engine has no control over sorting the data. This means that you will have to manually sort the data in the result set that is supplied by the engine before you use it with a BY-group statement.

  For example:

```
libname mylib infomaps ... ;

proc sort data=mylib.results_set out=work.sorted;
     by sorted_var;
run;

proc print data=work.sorted;
     by sorted_var;
run;
```

  The one exception is the SQL procedure. You can use BY-group processing with the Information Maps engine's result set because the SQL procedure automatically sorts the result set before it applies the BY-group statement.
- □ The engine does not support OLAP data.
- □ The engine does not support updating or deleting an information map, nor does it support updating the underlying data.
- □ The engine does not provide explicit SQL Pass-Through support.

What Metadata Is Used by the Information Maps Engine?

In addition to the metadata for the information map, the Information Maps engine accesses metadata about the underlying physical data. This information can include the data type, the data size, the format, and the label.

The Information Maps engine uses metadata that is stored in a specific metadata repository on the metadata server. The metadata server provides metadata management services to one or more client applications. A metadata repository is a collection of related metadata objects, such as the metadata for a set of information maps that are maintained by an application.

There are several methods to create metadata in a metadata repository. For example, from the SAS Management Console, you can use a Library wizard to create the metadata objects in a metadata repository that are necessary for the engine to construct a LIBNAME statement. SAS Data Integration Studio enables you to define the table that you want to be a member in a library and any options that you want associated with that table. You provide information to SAS Data Integration Studio, and SAS Data Integration Studio generates corresponding metadata.

CHAPTER

4

LIBNAME Statement for the Information Maps Engine

Using the LIBNAME Statement

The LIBNAME statement for the Information Maps engine associates a SAS libref with information maps that are stored in a metadata repository. The engine reads information maps and uses their metadata to access underlying data.

You must have a metadata repository available on the metadata server that contains metadata that defines the information maps to be accessed. For the necessary repository identifiers and metadata object names and identifiers, see the documentation for your application.

The metadata server, which is a multi-user server that stores metadata from one or more metadata repositories, must be running in order to execute the LIBNAME statement for the engine.

For information about defining metadata, installing and setting up a standard SAS Metadata Server, or changing the standard configuration options for the SAS Metadata Server, see the *SAS Intelligence Platform: System Administration Guide*.

LIBNAME Statement Syntax

LIBNAME *libref* **INFOMAPS** <*options*>;

Required Arguments

libref
is a valid SAS name that serves as a shortcut to associate with the information maps that are in the metadata repository. This name must conform to the rules for SAS names. A libref cannot exceed eight characters.

INFOMAPS
: is the engine name for the SAS Information Maps LIBNAME engine.

CAUTION:
The engine nickname (INFOMAPS) is not defined by default in SAS 9.1.3. If your system administrator has not set up the nickname, you can specify the actual engine name `sasioime` when you use the LIBNAME statement syntax. △

LIBNAME Statement Options for Connecting to the SAS Metadata Server

The following LIBNAME statement options establish a connection to the metadata server and identify the information maps in the metadata repository.

MAPPATH="*repository-path*"
: specifies the path to the location of the information maps within the metadata repository. The path is hierarchical and the separator character is a slash (/). This is a required option for the Information Maps engine.

For example, `mappath="/BIP Tree/ReportStudio/Maps"`. If the path does not contain a blank space (or spaces), enclosing the identifier in quotation marks is optional.

Alias: PATH=

METAPASS="*password*"
: specifies the password that corresponds to the user ID that connects to the metadata server. For example, `metapass="My Password"` or `metapass=MyPassword`. If the password is not encoded or does not contain a blank space (or spaces), enclosing the identifier in quotation marks is optional.

If this option is not specified, the value is obtained from the METAPASS= system option. See the METAPASS= system option in the *SAS Language Reference: Dictionary*.

Alias: PASSWORD= | PW=

METAPORT=*port-number*
: specifies the TCP port that the metadata server is listening to for connections. For example, `metaport=8561`. Enclosing the identifier in quotation marks is optional.

If this option is not specified, the value is obtained from the METAPORT= system option. See the METAPORT= system option in the *SAS Language Reference: Dictionary*.

Alias: PORT=

METAREPOSITORY="*repository-name*"
: specifies a name that is assigned to a particular metadata repository to use on the metadata server. For example, `metarepository="My Repository"` or `metarepository=myrepos`. If the repository name does not contain a blank space (or spaces), enclosing the identifier in quotation marks is optional.

If this option is not specified, the value is obtained from the METAREPOSITORY= system option. See the METAREPOSITORY= system option in the *SAS Language Reference: Dictionary*.

Alias: REPOSITORY= | REPOS= | REPNAME=

Restriction: If you supply a list of repositories, only the first repository name in the list is accepted by the engine.

METASERVER="*address*"
: specifies the network IP (Internet Protocol) address of the computer that hosts the metadata server. For example, `metaserver="myip.us.mycompany.com"` or

`metaserver=myip.us.mycompany.com`. Enclosing the identifier in quotation marks is optional. This is a required option for the Information Maps engine, because there is no default IP address.

If this option is not specified, the value is obtained from the METASERVER= system option. See the METASERVER= system option in the *SAS Language Reference: Dictionary*.

Alias: SERVER= | HOST= | IPADDR=

METAUSER="*user-ID*"

specifies the user ID to connect to the metadata server. For example, **`metauser="My UserID"`** or **`metauser=myUserID`**. If the user ID does not contain a blank space (or spaces), enclosing the identifier in quotation mark is optional.

If this option is not specified, the value is obtained from the METAUSER= system option. See the METAUSER= system option in the *SAS Language Reference: Dictionary*.

Alias: USER= | USERID= | ID=

Restriction: In the metadata server, you must have at least one login definition that contains a user ID that corresponds to the user ID that you specify here. For information about login definitions, see the User Manager Help for logins in the SAS Management Console.

Restriction: If your metadata server runs in a Windows environment, then you must fully qualify the user ID by using the domain or machine name that you specified when your login object was created in a SAS Metadata Repository. For example: **`metauser="Windows-domain-name\user-ID"`**.

Other LIBNAME Statement Options for the Information Maps Engine

The following LIBNAME statement options for the Information Maps engine are global options that exist for the lifetime of the libref.

EXPCOLUMNLEN=*integer*

specifies the length of the SAS character column when a data item defined with expressions is encountered.

The EXPCOLUMNLEN= option doubles as a data set option. This means that this option value can be changed during a DATA step when the Information Maps engine is used. This changed value is in effect only during the execution of the DATA step. Once the DATA step is completed, the value will revert to the setting at the time of libref creation. For more information, see the “EXPCOLUMNLEN= Data Set Option” on page 43.

Default: 32

PRESERVE_TAB_NAMES=YES | NO

YES

specifies that information map names are read with special characters, and that the exact, case-sensitive spelling of the name is preserved.

Note: To access information maps with special characters or blank spaces, you have to use SAS name literals. For more information about SAS name literals, see "Rules for Words and Names in the SAS Language" and "Avoiding Errors When Using Name Literals" in the *SAS Language Reference: Concepts*. △

NO
: specifies that when you refer to an information map, the information map name is derived from SAS member names by using SAS member-name normalization. When you use SAS to read a list of information map names (for example, in the SAS Explorer window), the information maps whose names do not conform to the SAS member-name normalization rules do not appear in the output. In SAS command-line mode, the number of information maps is not displayed by the DATASETS procedure. The restriction appears as a note:

```
NOTE: Due to the PRESERVE_TAB_NAMES=NO LIBNAME option setting,
  12 information map(s) have not been displayed.
```

You do not get this note when using the SAS Explorer window.

The SAS Explorer window displays information map names in capitalized form when PRESERVE_TAB_NAMES=NO. These information map names follow the SAS member-name normalization rules and might not represent the exact table name in the information map.

Default: YES

READBUFF=*integer*
: specifies the positive number of rows to hold in memory. This option improves performance by specifying a number of rows that can be held in memory for input into SAS. Buffering data reads can decrease network activities and increase performance. However, because SAS stores the rows in memory, higher values for READBUFF= use more memory. In addition, if too many rows are selected at once, then the rows that are returned to the SAS application might be out of date. For example, if someone else modifies the rows, you do not see the changes. When READBUFF=1, only one row is retrieved at a time.

The READBUFF= option doubles as a data set option. This means that this option value can be changed during a DATA step when the Information Maps engine is used. This changed value is in effect only during the execution of the DATA step. Once the DATA step is completed, the value will revert to the setting at the time of libref creation. For more information, see the “READBUFF= Data Set Option” on page 45.

Alias: BUFFSIZE=

Default: 1000

SPOOL=YES | NO

YES
: specifies that SAS creates a spool file into which it writes the rows of data that are read for the first time. For subsequent passes through the data, rows are read from the spool file, rather than being reread from the original data source(s). This guarantees that each pass through the data processes the same information.

NO
: specifies that the required rows for all passes through the data are read from the original data source(s). No spool file is written. There is no guarantee that each pass through the data processes the same information.

Default: NO

CHAPTER 5
SAS Data Set Options for the Information Maps Engine

Using Data Set Options

Data set options specify actions that apply only to the SAS data set with which they appear. Specify a data set option in parentheses after a SAS data set name. To specify several data set options, separate them with spaces. For example:

```
(option-1=value-1<...option-n=value-n>)
```

For more information about SAS data set options, see the *SAS Language Reference: Dictionary*.

The following data set options for the Information Maps engine exist for the lifetime of the DATA step and override the LIBNAME option values when the option can be specified in both places.

EXPCOLUMNLEN= Data Set Option

Overrides the default length of the SAS character column when a data item defined with expressions is encountered

Valid in: DATA Step

Category: Data Set Control

Restriction: Use with character column only

Default: 32

Syntax

EXPCOLUMNLEN=*integer*

integer
specifies the length of the SAS column when expressions are used. Valid values are 1 to a maximum SAS column size.

Details

When character data items are defined with expressions in an information map, the length of the resulting SAS column cannot be readily determined by the Information Maps engine. Use the EXPCOLUMNLEN= option to assign a value to the length of the column. This value can be tuned based on an understanding of the results of the expression and of the data involved.

See Also

EXPCOLUMNLEN= option in "Other LIBNAME Statement Options for the Information Maps Engine" on page 41

FILTER= Data Set Option

Determines the criteria in a query (such as an SQL WHERE clause) for subsetting a result set

Valid in: DATA Step

Category: Data Set Control

Restriction: Use only with information map filters

Restriction: Prompted filters are not available

Syntax

FILTER=(*filter-name* <AND *filter-name-n*>)

Syntax Description

filter-name
specifies the name of a filter that is defined in the information map.

filter-name-n
specifies multiple filters that are defined in the information map. The AND Boolean operator is required.

Details

A filter is criteria in a query for subsetting a result set. An example of a filter is **`gender="Male"`**. You can get a list of the filters for an information map by using the CONTENTS procedure with a libref that is created by the Information Maps engine and by using the information map name.

READBUFF= Data Set Option

Specifies the number of rows to hold in memory for input into SAS

Valid in: DATA Step and LIBNAME Statement

Category: Data Set Control

Alias: BUFFSIZE=

Default: 1000

Syntax

READBUFF=*integer*

Syntax Description

integer
specifies the positive number of rows to hold in memory.

Details

This option improves performance by specifying a number of rows that can be held in memory for input into SAS. Buffering data reads can decrease network activities and increase performance. However, because SAS stores the rows in memory, higher values for the READBUFF= option use more memory. In addition, if too many rows are selected at once, then the rows that are returned to the SAS application might be out of date. For example, if someone else modifies the rows, you do not see the changes. When READBUFF=1, only one row is retrieved at a time.

The higher the value for the READBUFF= option, the more rows the engine retrieves in one fetch operation. The effect can be greater performance, but the expense is using more memory.

See Also

READBUFF= option in "Other LIBNAME Statement Options for the Information Maps Engine" on page 41

CHAPTER 6
Examples of Using the Information Maps Engine

Example 1: Submitting a LIBNAME Statement Using the Defaults

This example shows you a LIBNAME statement that uses the defaults for the Information Maps engine. Note that the SAS Metadata Server connection information is obtained from the metadata server system options, which this example assumes have been set previously by a SAS configuration file or with the OPTIONS statement.

```
libname mylib infomaps
              mappath="/BIP Tree/ReportStudio/Maps";
```

Output 6.1 Output from the LIBNAME Statement That Is Displayed in the Log Window

```
1     libname mylib infomaps
2                   mappath="/BIP Tree/ReportStudio/Maps"
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /BIP Tree/ReportStudio/Maps
```

Example 2: Submitting a LIBNAME Statement Using All the Statement Options

This example shows you a LIBNAME statement that uses all of the engine's LIBNAME statement options in order to connect to the metadata server.

```
libname mylib infomaps metauser=myUserID
                       metapass=myPassword
                       metaserver=myip.us.mycompany.com
                       metaport=8561
                       mappath="/BIP Tree/ReportStudio/Maps"
                       metarepository=Foundation;
```

Output 6.2 Output from the LIBNAME Statement That Is Displayed in the Log Window

```
1     libname mylib infomaps metauser=myUserID
2                            metapass=XXXXXXXXXX
3                            metaserver=myip.us.mycompany.com
4                            metaport=8561
5                            mappath="/BIP Tree/ReportStudio/Maps"
6                            metarepository=Foundation;
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /BIP Tree/ReportStudio/Maps
```

CHAPTER 7 Best Practices for Using the INFOMAPS Procedure or the Information Maps Engine

Best Practices for Using the INFOMAPS Procedure

To improve the performance of the INFOMAPS procedure, consider the following:

- Use the COLUMN= option for the INSERT DATAITEM statement, unless you have a calculated data item. For more information about the INSERT DATAITEM statement and the COLUMN= option, see “INSERT DATAITEM Statement” on page 9.
- For an information map to use a table, the table must have a unique name in its SAS library (for a SAS table) or database schema (for a table from a different DBMS) in the metadata repository. If multiple tables in a SAS library or database schema have the same name, then you must perform one of the following tasks before you can use any of the tables with an information map:
  - From either SAS Data Integration Studio or the Data Library Manager in SAS Management Console, you can rename a table by changing the value of the `Name` field on the `General` tab in the properties window for the table.
  - From SAS Data Integration Studio, delete the duplicate tables.

Best Practices for Using the Information Maps Engine

Improving the Performance of the Information Maps Engine

To improve the performance of the Information Maps engine, consider the following:

- Use filters to reduce the amount of data that the engine has to return. Select only the data items that you need.
- Be careful when creating information maps for the engine to use. Consider the quality of the data and heterogeneous joins.

- If you use static data (that is, data you know will not change during the time you are using it), retrieve the data once with the Information Maps engine and then save the data to a data set that is local to your SAS session. You will save time by not having to access the static data (which could be on another server) multiple times.
- If the data is on your local machine or if you have clients on your local machine that can get to the data, then you will get the best performance from the engine. If the data or the clients are not on your local machine, then a message appears in the SAS log indicating that performance will not be optimal. Here is what the message looks like:

```
NOTE: The Information Maps LIBNAME Engine is retrieving data via
      a remote connection. Performance is not optimized.
```

Creating Information Maps that Work Well with the Information Maps Engine

Following SAS Naming Restrictions

Information maps that work well with the Information Maps engine meet the following restrictions:

- Names have a maximum length of 32 characters in Base SAS software.
- Descriptions have a maximum length of 256 characters in Base SAS software. Note that when you are using the Information Maps engine, the information map's path, the data item name, and the description for the information map are all combined into one description. If this combined description is more than 256 characters, then it will be truncated.

Note: Clients that rely on the Information Maps engine, such as SAS Enterprise Guide and SAS Add-In for Microsoft Office, are affected by these name and description length constraints. △

For more information about names in the SAS language, see "Rules for Words and Names in the SAS Language" in the *SAS Language Reference: Concepts*.

Using Calculated Data Items

Calculated data items in information maps used by the Information Maps engine or by clients that rely on the engine, such as SAS Enterprise Guide and SAS Add-In for Microsoft Office, should be created using the physical data in the expression whenever possible. Data items that are based on expressions that include either business data or summarization functions cannot be used in detailed queries. Thus, the Information Maps engine cannot use them either.

Working with Natural Language Names in SAS

Information map names, data item names, and filter names can be stored as natural language names in the metadata. Natural language names have blank spaces separating the words in the name or include symbols in the name. To be able to use natural language names in SAS, you need to do the following:

- Make sure that the PRESERVE_TAB_NAMES option is set to YES (the default) if you are using information maps with natural language names and want them to be accessible to the Information Maps engine. For more information about the PRESERVE_TAB_NAMES option, see "Other LIBNAME Statement Options for the Information Maps Engine" on page 41.

- □ To specify a name that contains any characters, including blank spaces or mixed-case letters, use the VALIDVARNAME=ANY option. This SAS system option controls the type of SAS variable names that can be created and processed during a SAS session. For more information on the VALIDVARNAME system option, see the *SAS Language Reference: Dictionary*.
- □ To specify a name that contains blank spaces or symbols, enclose the name within a SAS name literal. For more information on SAS name literals, see "Rules for Words and Names in the SAS Language" and "Avoiding Errors When Using Name Literals" in the *SAS Language Reference: Concepts*.

In the following example, a natural language name is used with the Information Maps engine:

```
libname x infomaps <connection options>;

option validvarname=any;
proc print data=x.'Results (Yearly)'n;
run;
```

The VALIDVARNAME=ANY option allows the information map name to include blank spaces, as well as the parentheses symbols. The SAS name literal surrounds the information map name in the PRINT procedure statement to allow the name **`Results (Yearly)`** to remain intact and contain the symbols that are otherwise not allowed in SAS.

Increasing Memory Usage for the Information Maps Engine

It is important that your middle-tier components be configured for efficiency and performance. This includes making sure that your Java Virtual Machine (JVM) is properly tuned and has the relevant memory settings specified correctly. The JVM's garbage collector should be configured appropriately.

For detailed information about improving the performance of your middle-tier components, see the "Best Practices for Configuring Your Middle Tier" chapter in the *SAS Intelligence Platform: Web Application Administration Guide*.

CHAPTER

8 Example: Using the INFOMAPS Procedure and the Information Maps Engine

About This Example

The example in this chapter shows you how to use the INFOMAPS procedure to create a new information map and then use the Information Maps engine to retrieve the data associated with the new information map. Once you have the data, you can use Base SAS software to analyze it.

For the example, suppose that the management team in the Human Resources (HR) department in your company wants to analyze some of the employees' salary data. The HR managers are looking for a report with statistical breakdowns that can be updated on a regular basis. Based on the output of this report, they want to be able to create additional Web-based reports on the same information.

You are part of the IT team, so you know that the analyses are updated and modified constantly (to meet the changing demands of the company). You would like to set up the environment programmatically to support the request from the HR management team. You decide to build the statistical report on top of an information map, so the information map can be used later in SAS Web Report Studio.

Step 1: Set the Metadata System Options and a Macro Variable

To get started with this example, you need to set the metadata system options and a macro variable with your site-specific data. This is a good programming technique that makes it easy for you to customize SAS code for your environment.

The following code sets the metadata system options and a macro variable:

```
/* Set the metadata system options. */
options metauser="marcel"
        metapass=AA358mk
        metaport=8561
        metaserver="server21.us.anycompany.com"
```

```
          metarepository=Foundation;

/* Assign a macro variable to the path for the information map */
/* to avoid having to set the path multiple times.             */
%LET infomap_path=/BIP Tree/ReportStudio/Maps;
```

Output 8.1 Output from the Metadata System Options and the Macro Variable That Is Displayed in the Log Window

```
/* Set the metadata system options. */
options metauser="marcel"
        metapass=XXXXXXX
        metaport=8561
        metaserver="server21.us.anycompany.com"
        metarepository=Foundation;

/* Assign a macro variable to the path for the information map */
/* to avoid having to set the path multiple times.             */
%LET infomap_path=/BIP Tree/ReportStudio/Maps;
```

For more information about macro variables or the %LET statement, see the *SAS Macro Language: Reference*.

Step 2: Register Data Using the METALIB Procedure

This example uses three tables in the **`C:\Program Files\SAS\SAS 9.1\core\sample`** directory. (The tables might be in a different directory, depending on where SAS is installed at your location). For this example, you will be using the **`empinfo`**, **`jobcodes`**, and **`salary`** tables.

Note: This example assumes that you have used SAS Management Console to create a library called **`HR`** that points to the three SAS tables. For more information about creating libraries using SAS Management Console, see the *SAS Management Console: User's Guide*. △

To register the tables in a SAS Metadata Repository, you need to use the METALIB procedure. The METALIB procedure synchronizes table definitions in a metadata repository with current information from the physical library data source. For more information about the METALIB procedure, see the *SAS Open Metadata Interface: Reference*.

The following code registers the tables using the METALIB procedure:

```
/* Use the library object defined in a SAS Metadata Repository  */
/* to obtain all accessible table metadata from the data source */
/* to create table metadata in the metadata repository.         */
proc metalib;
     omr (library="HR");
     select("empinfo" "jobcodes" "salary");

     /* Create a summary report of the metadata changes. */
     report;
run;
```

Note: If you run the METALIB procedure code more than once, your output will be different than what is shown. △

Output 8.2 Output from the METALIB Procedure That Is Displayed in the Log Window

```
12   /* Use the library object defined in a SAS Metadata Repository  */
13   /* to obtain all accessible table metadata from the data source */
14   /* to create table metadata in the metadata repository.         */
15   p
15 !  roc metalib;
16        omr (library="HR");
17        select("empinfo" "jobcodes" "salary");
18
19        /* Create a summary report of the metadata changes. */
20        report;
21   run;

NOTE: A total of 3 tables were analyzed for library "HR".
NOTE: Metadata for 0 tables was updated.
NOTE: Metadata for 3 tables was added.
NOTE: Metadata for 0 tables matched the data sources.
NOTE: 0 tables listed in the SELECT or EXCLUDE statement were not found in either the metadata
      or the data source.
NOTE: 0 other tables were not processed due to error or UPDATE_RULE.
NOTE: PROCEDURE METALIB used (Total process time):
      real time           2.03 seconds
      cpu time            0.48 seconds
```

Output 8.3 Output from the METALIB Procedure That Is Displayed in the Output Window

```
                                      The METALIB Procedure
                                  Summary Report for Library HR
                                      Repository Foundation

                                   Metadata Summary Statistics

                                Total tables analyzed          3
                                Tables Updated                 0
                                Tables Added                   3
                                Tables matching data source    0
                                Tables not found               0
                                Other tables not processed     0

    __________________________________________________________________________________________
                                          Tables Added
    __________________________________________________________________________________________

    Metadata Name                      Metadata ID          SAS Name

    EMPINFO                            A511TK91.BC0007ES    EMPINFO
    JOBCODES                           A511TK91.BC0007ET    JOBCODES
    SALARY                             A511TK91.BC0007EU    SALARY
```

For information about defining metadata, installing and setting up a standard SAS Metadata Server, or changing the standard configuration options for the SAS Metadata Server, see the *SAS Intelligence Platform: System Administration Guide*.

Step 3: Create an Information Map Using the INFOMAPS Procedure

Once the tables are registered in the metadata repository, you can create a new information map. The INFOMAPS procedure inserts multiple data sources and data items, inserts relationships to join the tables, inserts four filters, and then saves the new information map.

The following code creates the new information map:

```
/* Create a new information map using the INFOMAPS procedure. */
proc infomaps mappath="&infomap_path";

/* Delete the information map to avoid duplicate data items. */
delete infomap "InfoMaps_Demo";

/* Open a new information map. */
open infomap "InfoMaps_Demo";

/* Insert a data source and three data items using the COLUMNS= option. */
insert datasource sasserver="SASMain"
       table="Infomap_sample_code".empinfo
       columns=("JobCode" "LOCATION" "DIVISION")
       id="Empinfo";

/* Insert a data item based on a physical column.  Because the ID= option */
/* is not specified, a note with its ID value will print in the SAS log.  */
insert dataitem column="Empinfo".idnum classification=category;

/* Insert a data item with an expression. */
insert dataitem expression="SUBSTRN(<<root.Jobcode>>, 1, 3)"
       type=character
       name="Department Code"
       id="Dept_code";

/* Insert a second data source, plus a data item into the */
/* current information map.                               */
insert datasource sasserver="SASMain"
       table="Infomap_sample_code".jobcodes
       columns=( "TITLE" )
       id="Jobcodes";

/* Insert a third data source into the current information map. */
insert datasource sasserver="SASMain"
       table="Infomap_sample_code".salary
       id="Salary";

/* Add joins between the tables. */
insert relationship "Empinfo" inner join "Jobcodes"
       on "(<<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>>)";

insert relationship "Empinfo" inner join "Salary"
       on "(<<Empinfo.IDNUM>>=<<Salary.IDNUM>>)";

/* Insert a folder and more business items. */
insert folder "Salary Info";
insert dataitem column="Salary".salary
       name="Annual Salary" folder="Salary Info";

insert dataitem expression="<<Salary.SALARY>>/12" type=numeric
       name="Monthly Salary" folder="Salary Info";

insert dataitem column="Salary".enddate folder="Salary Info";
```

```
/* Insert filters. */
insert filter "Status is Current"
       condition="<<root.Enddate>> IS NULL" folder="Salary Info";

insert filter "Education and Publications"
       condition="SUBSTRN<<root.Jobcode>>, 1, 3) IN ('EDU','PUB')"
       desc='Employees in Education and Publications';

insert filter "Host Systems Development"
       condition=' <<root.Division>>="HOST SYSTEMS DEVELOPMENT" '
       desc='Employees in Host Systems Development';

insert filter "Cary HQ"
       condition=' <<root.Location>>="Cary" '
       desc='Located in Cary, North Carolina HQ';

/* List all of the filters defined in the current information map. */
list filters;

/* Save the information map. */
save;
```

Note: If you run the INFOMAPS procedure code more than once, your output will be different than what is shown. △

Output 8.4 Output from the INFOMAPS Procedure That Is Displayed in the Log Window, Part 1 of 2

```
23   /* Create a new information map using the INFOMAPS procedure. */
24   proc infomaps mappath="&infomap_path";
25
26   /* Delete the information map to avoid duplicate data items. */
27   delete infomap "InfoMaps_Demo";
WARNING: The specified map "InfoMaps_Demo" to delete does not exist in "/BIP
         Tree/ReportStudio/Maps".
28
29   /* Open a new information map. */
30   open infomap "InfoMaps_Demo";
31
32   /* Insert a data source and three data items using the COLUMNS= option. */
33   insert datasource sasserver="SASMain"
34          table="Infomap_sample_code".empinfo
35          columns=("JobCode" "LOCATION" "DIVISION")
36          id="Empinfo";
37
38   /* Insert a data item based on a physical column.  Because the ID= option */
39   /* is not specified, a note with its ID value will print in the SAS log.  */
40   insert dataitem column="Empinfo".idnum classification=category;
NOTE: A data item was successfully inserted for the physical column Empinfo.IDNUM. Its ID is
      "Identification Number".
41
42   /* Insert a data item with an expression. */
43   insert dataitem expression="SUBSTRN(<<root.Jobcode>>, 1, 3)"
44          type=character
45          name="Department Code"
46          id="Dept_code";
47
48   /* Insert a second data source, plus a data item into the */
49   /* current information map.                               */
50   insert datasource sasserver="SASMain"
51          table="Infomap_sample_code".jobcodes
52          columns=( "TITLE" )
53          id="Jobcodes";
54
55   /* Insert a third data source into the current information map. */
56   insert datasource sasserver="SASMain"
57          table="Infomap_sample_code".salary
58          id="Salary";
59
60   /* Add joins between the tables. */
61   insert relationship "Empinfo" inner join "Jobcodes"
62          on "(<<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>>)";
63
64   insert relationship "Empinfo" inner join "Salary"
65          on "(<<Empinfo.IDNUM>>=<<Salary.IDNUM>>)";
66
67   /* Insert a folder and more business items. */
68   insert folder "Salary Info";
69   insert dataitem column="Salary".salary
70          name="Annual Salary" folder="Salary Info";
71
72   insert dataitem expression="<<Salary.SALARY>>/12"  type=numeric
73          name="Monthly Salary" folder="Salary Info";
74
75   insert dataitem column="Salary".enddate folder="Salary Info";
NOTE: A data item was successfully inserted for the physical column Salary.ENDDATE. Its ID is
      "Enddate".
76
```

Output 8.5 Output from the INFOMAPS Procedure That Is Displayed in the Log Window, Part 2 of 2

```
77   /* Insert filters. */
78   insert filter "Status is Current"
79          condition="<<root.Enddate>> IS NULL" folder="Salary Info";
80
81   insert filter "Education and Publications"
82          condition="SUBSTRN(<<root.Jobcode>>, 1, 3) IN ('EDU','PUB')"
83          desc='Employees in Education and Publications';
84
85   insert filter "Host Systems Development"
86          condition=' <<root.Division>>="HOST SYSTEMS DEVELOPMENT" '
87          desc='Employees in Host Systems Development';
88
89   insert filter "Cary HQ"
90          condition=' <<root.Location>>="Cary" '
91          desc='Located in Cary, North Carolina HQ';
92
93   /* List all of the filters defined in the current information map. */
94   list filters;
Filter name: Cary HQ
ID: Cary HQ
Folder: /
Description: Located in Cary, North Carolina HQ
Expression:  <<root.Location>>="Cary"

Filter name: Education and Publications
ID: Education and Publications
Folder: /
Description: Employees in Education and Publications
Expression: SUBSTRN(<<root.Jobcode>>, 1, 3) IN ('EDU','PUB')

Filter name: Host Systems Development
ID: Host Systems Development
Folder: /
Description: Employees in Host Systems Development
Expression:  <<root.Division>>="HOST SYSTEMS DEVELOPMENT"

Filter name: Status is Current
ID: Status is Current
Folder: /Salary Info
Description:
Expression: <<root.Enddate>> IS NULL

95
96   /* Save the information map. */
97   save;
NOTE: Information map "InfoMaps_Demo" has been saved in folder "/BIP Tree/ReportStudio/Maps".
```

Step 4: Retrieve the Data Associated with the Information Map Using the Information Maps Engine

Now that you have an information map, you can use the Information Maps engine to access the metadata and then retrieve the underlying data. Once you retrieve the data, you can run almost any SAS procedure against it.

Note: The Information Maps engine nickname, INFOMAPS, is used in this example. If your systems administrator has not defined this nickname, you can specify the actual engine name **`sasioime`** when you use the LIBNAME syntax. △

The following code retrieves the data associated with the newly created information map:

```
/* If the INFOMAPS nickname for the engine is not defined, */
/* then specify the actual engine name SASIOIME when you   */
/* use the LIBNAME statement syntax.                       */

/* Run the Information Maps engine to retrieve the data. */
libname im_samp infomaps mappath="&infomap_path";

/* Allow mixed-case letters and blank spaces in information map names. */
option validvarname=any;
```

Note: Unlike running the INFOMAPS procedure code more than once, if you run the Information Maps engine code multiple times, the output should be the same as what is shown. △

Output 8.6 Output from the Information Maps Engine That Is Displayed in the Log Window

```
99   /* If the INFOMAPS nickname for the engine is not defined, */
100  /* then specify the actual engine name SASIOIME when you   */
101  /* use the LIBNAME statement syntax.                       */
102
103  /* Run the Information Maps engine to retrieve the data. */
104  libname im_samp infomaps mappath="&infomap_path";
NOTE: Libref IM_SAMP was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /BIP Tree/ReportStudio/Maps
105
106  /* Allow mixed-case letters and blank spaces in information map names. */
107  option validvarname=any;
WARNING: Only Base procedures and SAS/STAT procedures have been tested for use with
         VALIDVARNAME=ANY. Other use of this option is considered experimental and may cause
         undetected errors.
```

Step 5: View the Data Items and Filters Using the CONTENTS Procedure

You can view the data items and filters in the new information map that you just created. The following code uses the CONTENTS procedure to display information about the data items:

```
/* View the data items, including any filters, in the information map. */
proc contents data=im_samp.InfoMaps_Demo;
run;
```

Output 8.7 Output from the CONTENTS Procedure That Is Displayed in the Log Window

```
109  /* View the data items, including any filters, in the information map. */

NOTE: PROCEDURE INFOMAPS used (Total process time):
      real time           28.25 seconds
      cpu time            0.15 seconds


110  proc contents data=im_samp.InfoMaps_Demo;
111  run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           1.11 seconds
      cpu time            0.06 seconds
```

Output 8.8 Output from the CONTENTS Procedure That Is Displayed in the Output Window

```
                              The CONTENTS Procedure
               Data Set Name        IM_SAMP.InfoMaps_Demo    Observations          .
               Member Type          DATA                     Variables             9
               Engine               INFOMAPS                 Indexes               0
               Created              .                        Observation Length    0
               Last Modified        .                        Deleted Observations  0
               Protection                                    Compressed            NO
               Data Set Type                                 Sorted                NO
               Label                                         Filters               4
               Data Representation  Default
               Encoding             Default


                           Alphabetic List of Variables and Attributes

    # Variable              Type Len Format    Label

    7 Annual_Salary         Num    8 DOLLAR12. /Salary Info/Annual Salary; Physical column SALARY
    5 Department_Code       Char  32           /Department Code;
    3 Division              Char  40           /Division; Physical column DIVISION
    9 Enddate               Num    8 DATE9.    /Salary Info/Enddate; Physical column ENDDATE
    4 Identification_Number Num    8 SSN11.    /Identification Number; Physical column IDNUM
    1 Jobcode               Char   8           /Jobcode; Physical column JOBCODE
    2 Location              Char   8           /Location; Physical column LOCATION
    8 Monthly_Salary        Num    8 DOLLAR12. /Salary Info/Monthly Salary;
    6 Title                 Char  20 $F20.     /Title; Physical column TITLE


                                        Information Maps

                                Filter
  FilterName                    Type       FilterDesc

  Status is Current             Unp
  Education and Publications    Unp        Employees in Education and Publications
  Host Systems Development      Unp        Employees in Host Systems Development
  Cary HQ                       Unp        Located in Cary, North Carolina HQ
```

Step 6: Print the Data from the Information Map

You can use the PRINT procedure to print all of the data that the information map contains. If the information map contains any filters, they can be used to restrict the amount of returned data. For the purpose of this example, only the first five observations are selected.

The following code uses the PRINT procedure to display information about the data items:

```
/* Print 5 observations from the data that the information map references. */
proc print data=im_samp.InfoMaps_Demo (obs=5);
run;
```

Output 8.9 Output from the PRINT Procedure That Is Displayed in the Log Window

```
113  /* Print 5 observations from the data that the information map references. */
114  proc print data=im_samp.InfoMaps_Demo (obs=5);
115  run;

NOTE: There were 5 observations read from the data set IM_SAMP.InfoMaps_Demo.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.71 seconds
      cpu time            0.11 seconds
```

Output 8.10 Output from the PRINT Procedure That Is Displayed in the Output Window

```
                                         The SAS System

                                                           Identification_    Department_
          Obs    Jobcode    Location    Division                Number           Code

            1    FAC011       Cary      FACILITIES            333-88-1850          FAC
            2    TS0007       Cary      TECHNICAL SUPPORT     333-88-7366          TS0
            3    SAM009       Cary      SALES & MARKETING     301-97-8691          SAM
            4    ACT001       Cary      FINANCE               333-44-5555          ACT
            5    VID001       Cary      VIDEO                 333-78-0101          VID

                                              Annual_        Monthly_
          Obs    Title                         Salary          Salary      Enddate

            1    LANDSCAPING SUPV             $28,000         $2,333           .
            2    TECH SUP ANALYST II          $32,000         $2,667           .
            3    MARKETING ANALYST            $52,000         $4,333           .
            4    TAX ACCOUNTANT I             $37,000         $3,083           .
            5    VIDEO PRODUCER               $25,400         $2,117           .
```

Step 7: Analyze the Data in SAS and Produce an ODS Report

You can use the MEANS procedure to analyze the annual salary data that you have retrieved from the information map. For the purpose of this example, you will use a DATA step to apply a filter to view only the data for the employees in the Host Systems Development division. You will then use the MEANS procedure to analyze the annual salary data for the mean, the minimum, and the maximum salaries for each job code in the division. And, finally, a report will be produced with ODS (Output Delivery System).

The following code analyzes the data and produces an ODS report:

```
DATA work.HRinfo;
     set im_samp.InfoMaps_Demo(filter='Host Systems Development'n);
  keep jobcode 'Annual Salary'n;
run;

/* Produce an ODS report. */
ods html body='example-body.htm';

/* Analyze the annual salary distribution data. */
proc means data=work.HRinfo maxdec=0;
     var 'Annual Salary'n;
  class jobcode;
  title "Annual Salary by Job Code";
run;
```

```
ods html close;
```

Output 8.11 Output from the DATA Step and the MEANS Procedure That Is Displayed in the Log Window

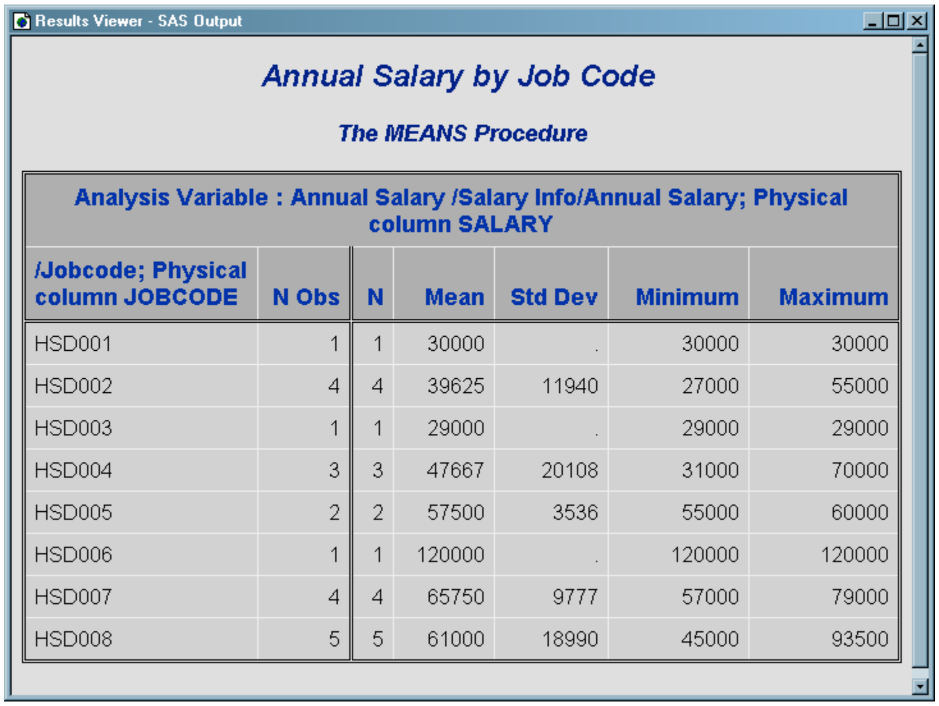
```
117  DATA work.HRinfo;
118       set im_samp.InfoMaps_Demo(filter='Host Systems Development'n);
119       keep jobcode 'Annual Salary'n;
120  run;

NOTE: There were 21 observations read from the data set IM_SAMP.InfoMaps_Demo.
NOTE: The data set WORK.HRINFO has 21 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.64 seconds
      cpu time            0.10 seconds



121
122  /* Produce an ODS report. */
123  ods html body='example-body.htm';
NOTE: Writing HTML Body file: example-body.htm
124
125  /* Analyze the annual salary distribution data. */
126  proc means data=work.HRinfo maxdec=0;
127       var 'Annual Salary'n;
128       class jobcode;
129       title "Annual Salary by Job Code";
130  run;

NOTE: There were 21 observations read from the data set WORK.HRINFO.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.17 seconds
      cpu time            0.06 seconds



131
132  ods html close;
```

Output 8.12 Output from the MEANS Procedure That Is Displayed in the Output Window

```
                              Annual Salary by Job Code

                                 The MEANS Procedure

 Analysis Variable : Annual Salary /Salary Info/Annual Salary; Physical column SALARY

/Jobcode;
Physical
column        N
JOBCODE      Obs     N            Mean         Std Dev         Minimum         Maximum
-----------------------------------------------------------------------------------------
HSD001         1     1           30000               .           30000           30000

HSD002         4     4           39625           11940           27000           55000

HSD003         1     1           29000               .           29000           29000

HSD004         3     3           47667           20108           31000           70000

HSD005         2     2           57500            3536           55000           60000

HSD006         1     1          120000               .          120000          120000

HSD007         4     4           65750            9777           57000           79000

HSD008         5     5           61000           18990           45000           93500
-----------------------------------------------------------------------------------------
```

The report that is produced by ODS should look similar to the following:

Display 8.1 Report That Is Displayed in the Results Viewer



Annual Salary by Job Code

The MEANS Procedure

Analysis Variable : Annual Salary /Salary Info/Annual Salary; Physical column SALARY						
/Jobcode; Physical column JOBCODE	N Obs	N	Mean	Std Dev	Minimum	Maximum
HSD001	1	1	30000	.	30000	30000
HSD002	4	4	39625	11940	27000	55000
HSD003	1	1	29000	.	29000	29000
HSD004	3	3	47667	20108	31000	70000
HSD005	2	2	57500	3536	55000	60000
HSD006	1	1	120000	.	120000	120000
HSD007	4	4	65750	9777	57000	79000
HSD008	5	5	61000	18990	45000	93500

APPENDIX 1 Recommended Reading

Recommended Reading

The recommended reading list for this title is:

- *The Little SAS Book: A Primer*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Intelligence Platform: System Administration Guide*
- SAS Companion that is specific to your operating environment
- Base SAS Community Web site at **support.sas.com/rnd/base**

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

aggregate function
a function that summarizes data and produces a statistic such as a sum, an average, a minimum, or a maximum.

business data
a collective term for data items in an information map. See also data item.

category
a data item whose distinct values are used to group measure data items, using an applied aggregate function.

classification
an attribute of data items that determines how they will be processed in a query. Data items can be classified as either categories or measures.

client
a computer or application that requests services, data, or other resources from a server. See also server.

column
in relational databases, a vertical component of a table. Each column has a unique name, contains data of a specific type, and has certain attributes. A column is analogous to a variable in SAS terminology.

cube
a set of data that is organized and structured in a hierarchical, multidimensional arrangement. A cube includes measures, and it can have numerous dimensions and levels of data.

data element
a general term that can include physical data (such as table columns, OLAP hierarchies, and OLAP measures) as well as data items. See also data item.

data item
in an information map, an item that represents either physical data (a table column, an OLAP hierarchy, or an OLAP measure) or a calculation. Data items are used for building queries. Data items are usually customized in order to present the physical data in a form that is relevant and meaningful to a business user.

data set
See SAS data set.

data source
the physical data (cube or table), as it is defined in a SAS Metadata Repository, that an information map consumer can query through an information map. The metadata for the physical data provides SAS Information Map Studio with the information that it needs in order to access the physical data.

DATA step
a group of statements in a SAS program that begins with a DATA statement and which ends with either a RUN statement, another DATA statement, a PROC statement, the end of the job, or the semicolon that immediately follows lines of data. The DATA step enables you to read raw data or other SAS data sets and to use programming logic to create a SAS data set, to write a report, or to write to an external file.

dimension
a group of closely related hierarchies. Hierarchies within a dimension typically represent different groupings of information that pertains to a single concept. For example, a Time dimension might consist of two hierarchies: (1) Year, Month, Date, and (2) Year, Week, Day. See also hierarchy.

engine
a component of SAS software that reads from or writes to a file. Each engine enables SAS to access files that are in a particular file format.

filter
in an information map, criteria that subset data. When a query is generated from an information map, the filter is converted to a query-language statement (for example, an SQL WHERE clause).

format
a pattern that SAS uses to determine how the values of a variable or data item should be written or displayed. SAS provides a set of standard formats and also enables you to define your own formats.

hierarchy
an arrangement of members of a dimension into levels that are based on parent-child relationships. Members of a hierarchy are arranged from more general to more specific. For example, in a Time dimension, a hierarchy might consist of the members Year, Quarter, Month, and Day. In a Geography dimension, a hierarchy might consist of the members Country, State or Province, and City. More than one hierarchy can be defined for a dimension. Each hierarchy provides a navigational path that enables users to drill down to increasing levels of detail. See also member, level.

informat
a pattern or set of instructions that SAS uses to determine how data values in an input file should be interpreted. SAS provides a set of standard informats and also enables you to define your own informats.

information map
a collection of data items and filters that describes and presents data in a form that is relevant and meaningful to a business user. A user of a query and reporting application such as SAS Web Report Studio can easily build a business report by using the parts of an information map as the building blocks for queries.

join
(1) the act of combining data from two or more tables in order to produce a single result set. (2) a specification that describes how you want data from two or more tables to be combined. The specification can be in the form of Structured Query Language (SQL) programming code, or it can be done interactively through a software user interface.

level
an element of a dimension hierarchy. Levels describe the dimension from the highest (most summarized) level to the lowest (most detailed) level. For example, possible levels for a Geography dimension are Country, Region, State or Province, and City.

libref (library reference)
a short name for the full physical name of a SAS library. In the context of the SAS Metadata Repository, a libref is associated with a SAS library when the library is defined in the metadata repository.

literal
a number or a character string that indicates a fixed value.

MDX (multidimensional expressions) language
a standardized, high-level language that is used for querying multidimensional data sources. The MDX language is the multidimensional equivalent of SQL (Structured Query Language).

measure
(1) a data item or column whose value can be used in computations or analytical expressions. Typically, these values are numeric. (2) a special dimension that usually represents numeric data values that are analyzed.

member
in a multidimensional database (or cube), a name that represents a particular data element within a dimension. For example, September 1996 might be a member of the Time dimension. A member can be either unique or non-unique. For example, 1997 and 1998 represent unique members in the Year level of a Time dimension. January represents non-unique members in the Month level, because there can be more than one January in the Time dimension if the Time dimension contains data for more than one year.

metadata
data about data. For example, metadata typically describes resources that are shared by multiple applications within an organization. These resources can include software, servers, data sources, network connections, and so on. Metadata can also be used to define application users and to manage users' access to resources. Maintaining metadata in a central location is more efficient than specifying and maintaining the same information separately for each application.

metadata object
a set of attributes that describe a table, a server, a user, or another resource on a network. The specific attributes that a metadata object includes vary depending on which metadata model is being used.

metadata server
a server that provides metadata management services to one or more client applications. A SAS Metadata Server is an example.

observation
a row in a SAS data set. All of the data values in an observation are associated with a single entity such as a customer or a state. Each observation contains either one data value or a missing-value indicator for each variable.

OLAP (online analytical processing)
a software technology that enables users to dynamically analyze data that is stored in cubes.

physical data
data values that are stored on any kind of physical data-storage media, such as disk or tape.

port
in a network that uses the TCP/IP protocol, an endpoint of a logical connection between a client and a server. Each port is represented by a unique number.

procedure
See SAS procedure.

prompted filter
a filter that is associated with a prompt, which enables the user of an information map to specify filtering criteria when a query is executed.

query
a set of instructions that requests particular information from one or more data sources.

register
to save metadata about an object to a metadata repository. For example, if you register a table, you save metadata about that table to a metadata repository.

relationship
the association, between tables in an information map, that generates a database join in a query.

repository
a location in which data, metadata, or programs are stored, organized, and maintained, and which is accessible to users either directly or through a network.

result set
the set of rows or records that a server or other application returns in response to a query.

row
in relational database management systems, the horizontal component of a table. A row is analogous to a SAS observation.

SAS data file
a type of SAS data set that contains data values as well as descriptor information that is associated with the data. The descriptor information includes information such as the data types and lengths of the variables, as well as the name of the engine that was used to create the data. See also SAS data set, SAS data view.

SAS data set
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. SAS data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data sets or from files whose contents are in other software vendors' file formats.

SAS data set option
an option that appears in parentheses after a SAS data set name. Data set options specify actions that apply only to the processing of that SAS data set.

SAS data view
a type of SAS data set that retrieves data values from other files. A SAS data view contains only descriptor information such as the data types and lengths of the variables (columns), plus other information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats. SAS data views can be created by the SAS DATA step and by the SAS SQL procedure.

SAS Information Map
See information map.

SAS library
a collection of one or more files that are recognized by SAS and that are referenced and stored as a unit. SAS libraries can be defined in a SAS Metadata Repository to provide centralized definitions for SAS applications.

SAS Metadata Repository
a repository that is used by the SAS Metadata Server to store and retrieve metadata. See also SAS Metadata Server.

SAS Open Metadata Architecture
a general-purpose metadata management facility that provides metadata services to SAS applications. The SAS Open Metadata Architecture enables applications to exchange metadata, which makes it easier for these applications to work together.

SAS procedure
a program that provides specific functionality and that is accessed with a PROC statement. For example, SAS procedures can be used to produce reports, to manage files, or to analyze data. Many procedures are included in SAS software.

SAS program
a group of SAS statements that guide SAS through a process or series of processes in order to read and transform input data and to generate output. The DATA step and the procedure step, used alone or in combination, form the basis of SAS programs.

SAS system option
an option that affects the processing of an entire SAS program or interactive SAS session from the time the option is specified until it is changed. Examples of items that are controlled by SAS system options include the appearance of SAS output, the handling of some files that are used by SAS, the use of system variables, the processing of observations in SAS data sets, features of SAS initialization, and the way SAS interacts with your host operating environment.

SAS Workspace Server
a SAS application server that provides access to Foundation SAS features such as the SAS programming language and SAS libraries.

schema
a map or model of the overall data structure of a database. An OLAP schema specifies which group of cubes an OLAP server can access.

server
a computer system that provides data or services to multiple users on a network. The term 'server' sometimes refers to the computer system's hardware and software, but it often refers only to the software that provides the data or services. In a network, users might log on to a file server (to store and retrieve data files), a print server (to use centrally located printers), or a database server (to query or update databases). In a client/server implementation, a server is a program that waits for and fulfills requests from client programs for data or services. The client programs might be running on the same computer or on other computers. See also client.

SQL (Structured Query Language)
a standardized, high-level query language that is used in relational database management systems to create and manipulate database management system objects.

statement option
a word that you specify in a particular SAS statement and which affects only the processing that that statement performs.

system option
See SAS system option.

table
a two-dimensional representation of data, in which the data values are arranged in rows and columns.

variable
a column in a SAS data set or in a SAS data view. The data values for each variable describe a single characteristic for all observations.

XML (Extensible Markup Language)
a markup language that structures information by tagging it for content, meaning, or use. Structured information contains both content (for example, words or numbers) and an indication of what role the content plays. For example, content in a section heading has a different meaning from content in a database table.

Index

Your Turn

If you have comments or suggestions about *Base SAS Guide to Information Maps*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **`yourturn@sas.com`**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **`suggest@sas.com`**