THE POWER TO KNOW.

Base SAS® 9.2 Guide to Information Maps

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *Base SAS® 9.2 Guide to Information Maps*. Cary, NC: SAS Institute Inc.

Base SAS® 9.2 Guide to Information Maps

Copyright © 2009, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-59994-001-4

All rights reserved. Produced in the United States of America.

For a hard-copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st electronic book, February 2009

2nd electronic book, December 2009

1st printing, December 2009

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contents

What's New ***v***
Overview ***v***
INFOMAPS Procedure ***v***
SAS Information Maps LIBNAME Engine ***vii***

Chapter 1 △ Overview of SAS Information Maps 1
What Is a SAS Information Map? **1**
Why Are SAS Information Maps Important? **2**
Where Can SAS Information Maps Be Used? **3**
Accessibility Features of the INFOMAPS Procedure and the SAS Information Maps LIBNAME Engine **3**

Chapter 2 △ The INFOMAPS Procedure 5
Overview: INFOMAPS Procedure **5**
Syntax: INFOMAPS Procedure **6**
Examples: INFOMAPS Procedure **57**

Chapter 3 △ Using the SAS Information Maps LIBNAME Engine 63
What Does the Information Maps Engine Do? **63**
Understanding How the Information Maps Engine Works **63**
Advantages of Using the Information Maps Engine **67**
What Is Required to Use the Information Maps Engine? **67**
What Is Supported? **67**

Chapter 4 △ LIBNAME Statement for the Information Maps Engine 69
Using the LIBNAME Statement **69**
LIBNAME Statement Syntax **69**

Chapter 5 △ SAS Data Set Options for the Information Maps Engine 75
Using Data Set Options **75**

Chapter 6 △ Examples of Using the Information Maps Engine 81
Example 1: Submitting a LIBNAME Statement Using the Defaults **81**
Example 2: Submitting a LIBNAME Statement Using Connection Options **81**

Chapter 7 △ Hints and Tips for Using the INFOMAPS Procedure or the Information Maps Engine 83
Hints and Tips for Using the INFOMAPS Procedure **83**
Hints and Tips for Using the Information Maps Engine **84**

Chapter 8 △ Example: Using the INFOMAPS Procedure and the Information Maps Engine 87
About This Example **87**
Step 1: Create a Library Definition in the SAS Metadata Server **87**
Step 2: Set the Metadata System Options and a Macro Variable **88**

Step 3: Register Data Using the METALIB Procedure **88**
Step 4: Create an Information Map Using the INFOMAPS Procedure **90**
Step 5: Retrieve the Data Associated with the Information Map Using the Information Maps Engine **96**
Step 6: View the Data Items and Filters Using the CONTENTS Procedure **96**
Step 7: Print the Data from the Information Map **98**
Step 8: Analyze the Data in SAS and Produce an ODS Report **100**

Appendix 1 △ SQL DICTIONARY Tables for the Information Maps Engine 103
Using SQL DICTIONARY Tables **103**
DICTIONARY.INFOMAPS Table **103**
DICTIONARY.DATAITEMS Table **104**
DICTIONARY.FILTERS Table **105**

Appendix 2 △ SAS Tracing and the Information Maps Engine 107
Tracing Diagnostic Messages from the Information Maps Engine **107**
Example **107**

Appendix 3 △ Recommended Reading 109
Recommended Reading **109**

Glossary 111

Index 117

What's New

Overview

The INFOMAPS procedure in Base SAS software has been enhanced for this release. You can change the definitions of any existing data item, filter, data source, folder, or relationship within an information map. You can move items between folders and associate stored processes with information maps.

The SAS Information Maps LIBNAME engine has also been enhanced for this release. The engine supports complex filter clauses and can read aggregated data as well as detailed data. You can specify an authentication domain for user logons.

INFOMAPS Procedure

The INFOMAPS procedure is now available on all operating systems for the SAS 9.2 release except OpenVMS on HP Integrity. Previously, the procedure was available only on Windows (32-bit), Solaris (64-bit), HP-UX, HP-UX on Itanium, AIX (64-bit), and z/OS.

The following statements in the INFOMAPS procedure have changes and enhancements:

- The PROC INFOMAPS statement has changed as follows:
  - The new DOMAIN= option specifies an authentication domain on the current server for user logons.
  - The METAPASS= and METAUSER= arguments are optional if the metadata server supports single sign-on.
  - The METAREPOSITORY= argument is now required only in the case where the information maps you want to use reside in a SAS 9.2 metadata repository that was converted from a custom repository in SAS 9.1.3.
- The EXPORT statement has changed. The FILE= argument is now required rather than optional.
- The INSERT DATAITEM statement has been enhanced. The new HIERARCHY= and MEASURE= options provide simpler alternatives to the EXPRESSION= option for creating data items for OLAP data sources. You can use the HIERARCHY=

option to specify a physical hierarchy in an OLAP data source. You can use the MEASURE= option to specify a physical measure in an OLAP data source.

You can use the new CREATE option to create the location for the data item if the location specified in the FOLDER= option does not already exist.

- The INSERT FILTER statement has changed. You now specify the filter name using the NAME= argument rather than as an explicit argument in the INSERT FILTER statement.

  You can use the new CREATE option to create the location for the filter if the location specified in the FOLDER= option does not already exist.

- The INSERT RELATIONSHIP statement has been enhanced. The following syntax is now obsolete:

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

  You now use the LEFT_TABLE=, RIGHT_TABLE=, JOIN=, and CONDITION= arguments to specify the relationship.

  The new DESCRIPTION= option specifies a description for a relationship.

- The LIST statement has been enhanced. The new RELATIONSHIPS option lists the identifier, left and right tables, cardinality, type of join, and join expression for each relationship defined in an information map.

  The output for the LIST statement now indicates when the information map does not contain the type of business data that has been requested.

- The MOVE DATAITEM statement is new for this release. This statement moves data items to a new location.
- The MOVE FILTER statement is new for this release. This statement moves filters to a new location.
- The MOVE FOLDER statement is new for this release. This statement moves folders to a new location.
- The OPEN INFOMAP statement has been replaced with the NEW INFOMAP statement to create a new information map and the UPDATE INFOMAP statement to update an existing information map.
- The SAVE statement has been updated. You can use the new CREATE option to create the location for the information map if it does not already exist.
- The SET STORED_PROCESS statement is new for this release. This statement associates a stored process with the current information map.
- The UPDATE DATAITEM statement is new for this release. This statement updates the properties of a specified data item in an information map.
- The UPDATE DATASOURCE statement is new for this release. This statement updates the properties of a data source in an information map.
- The UPDATE FILTER statement is new for this release. This statement updates the properties of a specified filter in an information map.
- The UPDATE FOLDER statement is new for this release. This statement updates the properties of a specified folder in an information map.
- The UPDATE RELATIONSHIP statement is new for this release. This statement updates the properties of a join relationship in an information map.

SAS Information Maps LIBNAME Engine

The Information Maps engine in Base SAS software has the following changes and enhancements:

- The Information Maps engine is now available on all operating systems for the SAS 9.2 release except OpenVMS on HP Integrity. Previously, the engine was available only on Windows (32-bit), Solaris (64-bit), HP-UX, HP-UX on Itanium, AIX (64-bit), and z/OS.
- The METAPASS= and METAUSER= arguments are optional if the metadata server supports single sign-on.
- The METAREPOSITORY= argument is now required only in the case where the information maps you want to use reside in a SAS 9.2 metadata repository that was converted from a custom repository in SAS 9.1.3.
- The SSPI= option can be used to specify Integrated Windows Authentication.
- The AGGREGATE= option for the LIBNAME statement and the AGGREGATE= data set option are new for this release. These options specify whether the engine uses detailed data or aggregated data.
- The DOMAIN= option for the LIBNAME statement is new for this release. This option specifies an authentication domain on the current server for user logons.
- The PRESERVE_MAP_NAMES= option for the LIBNAME statement has replaced the PRESERVE_TAB_NAMES option. (PRESERVE_TAB_NAMES is still supported as an alias for PRESERVE_MAP_NAMES.)
- The FILTER= data set option has been enhanced for this release. The OR and NOT Boolean operators are now supported. You can also use parentheses to signify precedence or any needed grouping within the clause.
- When you use the CONTENTS procedure in Base SAS software to show the contents of an information map referenced by the Information Maps engine, the Label column in the procedure output now shows only the data item description.
- SQL DICTIONARY tables are now available that contain information about the information maps, data items, and filters that can be accessed using the Information Maps engine.

CHAPTER

1

Overview of SAS Information Maps

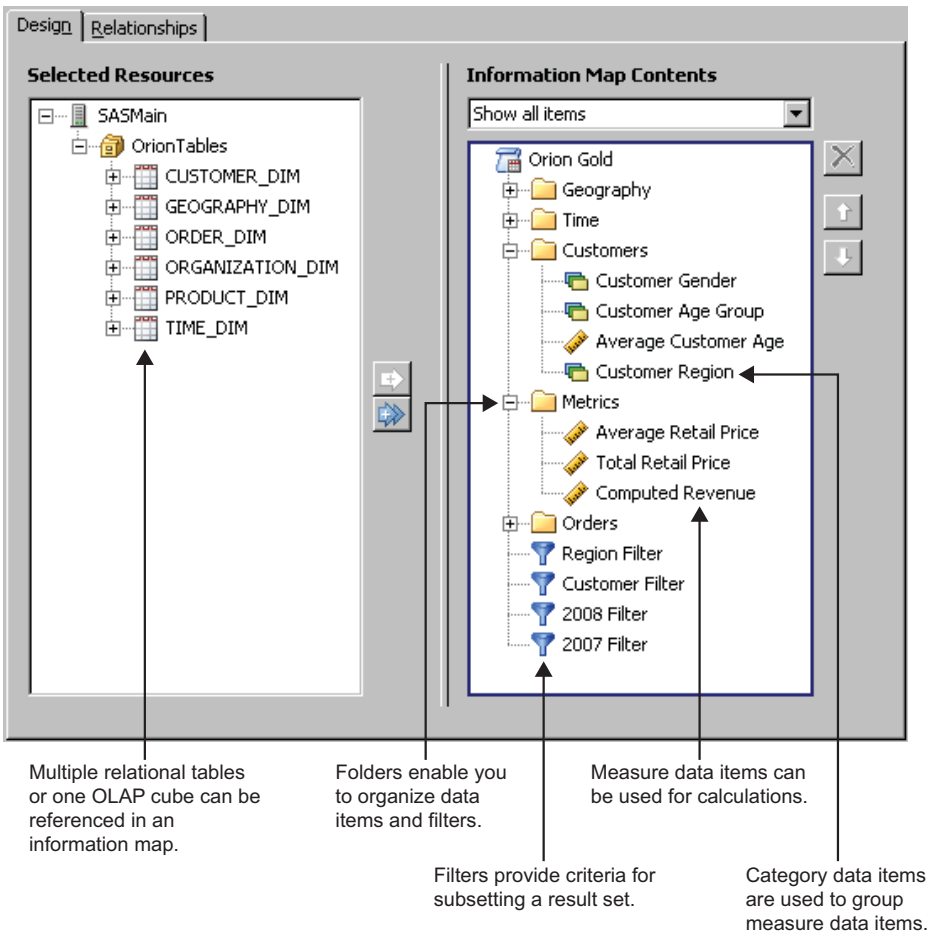
What Is a SAS Information Map? **1**
Why Are SAS Information Maps Important? **2**
Where Can SAS Information Maps Be Used? **3**
Accessibility Features of the INFOMAPS Procedure and the SAS Information Maps LIBNAME Engine **3**

What Is a SAS Information Map?

A SAS Information Map is business metadata that is applied on top of the data sources in your data warehouse. (Metadata is information about the structure and content of data. An information map does not contain any physical data.) Information maps provide business users with a user-friendly way to query data and get results for themselves.

An information map is based on one or more data sources, which can be tables or OLAP cubes. Information maps that are based on more than one table data source contain relationships that define how the data sources are joined. An information map contains data items and filters, which are used to build queries. A data item can refer to a data field or a calculation. Filters contain criteria for subsetting the data that is returned in a query. Folders can be used to organize the data items and filters so that business users can easily locate information within the information map.

To create an information map, you can use either SAS Information Map Studio, an application that provides a graphical user interface (GUI) for creating and viewing information maps, or the INFOMAPS procedure that is described in “Overview: INFOMAPS Procedure” on page 5. The following figure shows you what an information map looks like in the main window in SAS Information Map Studio.



Why Are SAS Information Maps Important?

Information maps provide a business metadata layer that enables business users to ask questions and get answers for themselves. This frees IT resources from ad hoc 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:

- Information maps shield users from the complexities of the data.
- Information maps make data storage transparent to users. It does not matter whether the data is relational or multidimensional, or whether the data is in a SAS data set or in a third-party database system.
- Information maps predefine business formulas and calculations, which makes them usable on a consistent basis.
- Information maps enable users to query data for answers to business questions without knowing query languages or being aware of the data model.

Where Can SAS Information Maps Be Used?

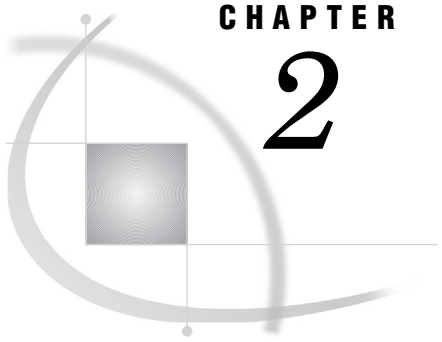
The following software can use information maps:

- Base SAS software
- SAS Add-In for Microsoft Office
- SAS Enterprise Guide
- SAS Information Delivery Portal
- SAS Marketing Automation
- SAS Web OLAP Viewer for Java
- SAS Web Report Studio

Information maps can also be used by custom applications developed with SAS AppDev Studio.

Accessibility Features of the INFOMAPS Procedure and the SAS Information Maps LIBNAME Engine

The INFOMAPS procedure and the Information Maps engine are part of Base SAS software. Base SAS is a command-based product. For this release, no features were added to address accessibility, but the product might very well be compliant to accessibility standards because it does not have a graphical user interface, and all of its features are available to anyone who can type or otherwise produce a command. If you have specific questions about the accessibility of SAS products, send them to **`accessiblity@sas.com`** or call SAS Technical Support.



CHAPTER 2

The INFOMAPS Procedure

Overview: INFOMAPS Procedure **5**
What Does the INFOMAPS Procedure Do? **5**
Syntax: INFOMAPS Procedure **6**
PROC INFOMAPS Statement **9**
DELETE INFOMAP Statement **11**
EXPORT Statement **12**
IMPORT Statement **13**
INSERT DATAITEM Statement **14**
INSERT DATASOURCE Statement **24**
INSERT FILTER Statement **27**
INSERT FOLDER Statement **29**
INSERT RELATIONSHIP Statement **31**
LIST Statement **33**
MOVE DATAITEM Statement **36**
MOVE FILTER Statement **37**
MOVE FOLDER Statement **38**
NEW INFOMAP Statement **39**
SAVE Statement **42**
SET STORED_PROCESS Statement **42**
UPDATE DATAITEM Statement **43**
UPDATE DATASOURCE Statement **48**
UPDATE FILTER Statement **49**
UPDATE FOLDER Statement **50**
UPDATE INFOMAP Statement **52**
UPDATE RELATIONSHIP Statement **55**
Examples: INFOMAPS Procedure **57**
Example 1: Creating a Basic Information Map **57**
Example 2: Creating an Information Map with Relationships and Filters **58**
Example 3: Aggregating a Data Item **60**

Overview: INFOMAPS Procedure

What Does the INFOMAPS Procedure Do?

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, data items, filters, folders, or relationships. Or you can change the definitions of any existing data item, filter, data source, folder, or relationship within an information map.

A SAS Information Map is a business metadata layer that is applied on top of the data sources in your data warehouse. (Metadata is information about the structure and content of data. An information map does not contain any physical data.) Information maps provide business users with a user-friendly way to query data and get results for themselves. 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, which are used to build queries. A data item can refer to a physical data source such as one or more columns from a table, to an OLAP hierarchy, or to an OLAP measure. It can also refer to one or more other data items in the same information map. A data item is classified as either a measure item or a category item. Measure items can be used for calculations. Category items are used to group measure items. Filters contain criteria for subsetting the data that is returned for a query. You can organize data items and filters 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 63.

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*.

Syntax: INFOMAPS Procedure

PROC INFOMAPS
<DOMAIN="*authentication-domain*">
<MAPPATH="*location*>"
<METACREDENTIALS=YES|NO>
<METAPASS="*password*">
<METAPORT=*port-number*>
<METASERVER="*address*">
<METAUSER="*user-ID*">;

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

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

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

INSERT DATAITEM
COLUMN="*data-source-ID*"."*column-name*" | EXPRESSION="*expression-text*" |
HIERARCHY="*dimension*"."*hierarchy*" | MEASURE="*OLAP-measure*"
<ACTIONS=(*actions-list*)>
<AGGREGATION=*aggregate-function*>
<AGGREGATIONS_DROP_LIST=(*aggregate-function-list*)>

<AGGREGATIONS_KEEP_LIST=(*aggregate-function-list*)>
<CLASSIFICATION=CATEGORY | MEASURE>
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<FOLDER="*folder-name*" | "*folder-location*" </CREATE>>
<FORMAT="*format-name*">
<ID="*data-item-ID*">
<NAME="*data-item-name*">
<TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP>
<VALUE_GENERATION=NONE | DYNAMIC | (*custom-value-list*)>;

INSERT DATASOURCE
SASSERVER="*application-server-name*"
TABLE="*library*"."*table*" <COLUMNS=(*column–1 <... column-n>*) | _ALL_>
<DESCRIPTION="*descriptive-text*">
<ID=*data-source-ID*>
<NAME="*data-source-name*">;

INSERT DATASOURCE
SASSERVER="*application-server-name*"
CUBE=<"*schema*".>"*cube*" <_ALL_>
<DESCRIPTION="*descriptive-text*">
<ID=*data-source-ID*>
<NAME="*data-source-name*">;

INSERT FILTER
CONDITION="*conditional-expression*"
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<FOLDER="*folder-name*" | "*folder-location*" </CREATE>>
<ID="*filter-ID*">
<NAME="*filter-name*">;

INSERT FOLDER "*folder-name*"
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<LOCATION="*parent-folder-name*" | "*parent-folder-location*" </CREATE>>;

INSERT RELATIONSHIP
CONDITION="*conditional-expression*"
LEFT_TABLE="*data-source-ID–1*"
RIGHT_TABLE="*data-source-ID-2*"
<CARDINALITY=ONE_TO_ONE | ONE_TO_MANY | MANY_TO_ONE |
MANY_TO_MANY | UNKNOWN>
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<ID="*relationship-ID*">
<JOIN=INNER | LEFT | RIGHT | FULL>;

LIST <DATAITEMS | DATASOURCES | FILTERS | RELATIONSHIPS | _ALL_> ;

MOVE DATAITEM "*data-item-ID*" | ID_LIST=("*data-item-ID-1*" <...
"*data-item-ID-n*">)
NEW_LOCATION="*new-folder-location*" </CREATE>;

MOVE FILTER "*filter-ID*" | ID_LIST=("*filter-ID-1*" <... "*filter-ID-n*">)
NEW_LOCATION="*new-folder-location*" </CREATE>;

MOVE FOLDER "*folder-name*"

NEW_LOCATION="*new-folder-location*" </CREATE>
<LOCATION="*current-folder-location*">;

NEW INFOMAP "*information-map-name*"
<AUTO_REPLACE=YES | NO>
<CREATE_TARGET_FOLDER=YES | NO>
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<INIT_CAP=YES | NO>
<MAPPATH="*location*" </CREATE>>
<REPLACE_UNDERSCORES=YES | NO>
<USE_LABELS=YES | NO>
<VERIFY=YES | NO>;

SAVE
<INFOMAP "*information-map-name*">
<MAPPATH="*location*" </CREATE>>;

SET STORED_PROCESS
NAME="*stored-process-name*"
<LOCATION="*stored-process-location*">;

UPDATE DATAITEM "*data-item-ID*"
<ACTIONS=(*actions-list*)>
<AGGREGATION=*aggregate-function*>
<AGGREGATIONS_LIST=_ALL_ | ADD (*aggregate-function-list*) |
REPLACE (*aggregate-function-list*) | REMOVE (*aggregate-function-list*)>
<CLASSIFICATION=CATEGORY | MEASURE>
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<EXPRESSION="*expression-text*">
<FORMAT="*format-name*">
<ID="*new-data-item-ID*">
<NAME="*data-item-name*">
<TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP>
<VALUE_GENERATION=NONE | DYNAMIC | ADD (*custom-value-list*) |
REPLACE (*custom-value-list*) | REMOVE (*unformatted-value-list*)>;

UPDATE DATASOURCE "*data-source-ID*"
<NAME="*data-source-name*">
<DESCRIPTION="*descriptive-text*">
<ID="new-*data-source-ID*">;

UPDATE FILTER "*filter-ID*"
<CONDITION="*conditional-expression*">
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<ID="*new-filter-ID*">
<NAME="*filter-name*">;

UPDATE FOLDER "*folder-name*"
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<LOCATION="*current-parent-folder-name*" | "*current-parent-folder-location*">
<NAME="*new-folder-name*">;

UPDATE INFOMAP "*information-map-name*"
<CREATE_TARGET_FOLDER=YES | NO>
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<INIT_CAP=YES | NO>
<MAPPATH="*location*">
<REPLACE_UNDERSCORES=YES | NO>
<USE_LABELS=YES | NO>
<VERIFY=YES | NO>;

UPDATE RELATIONSHIP "*relationship-ID*"
<CARDINALITY=ONE_TO_ONE | ONE_TO_MANY | MANY_TO_ONE |
MANY_TO_MANY | UNKNOWN>
<CONDITION="*conditional-expression*">
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<ID="*new-relationship-ID*">
<JOIN=INNER | LEFT | RIGHT | FULL>;

PROC INFOMAPS Statement

Connects to the specified metadata server.

PROC INFOMAPS
<DOMAIN="*authentication-domain*">
<MAPPATH="*location*">
<METACREDENTIALS=YES|NO>
<METAPASS="*password*">
<METAPORT=*port-number*>
<METASERVER="*address*">
<METAUSER="*user-ID*">;

Options

DOMAIN="*authentication-domain*"
specifies an authentication domain to associate the user ID and password with. If you do not specify an authentication domain, then the user ID and password are associated with the DefaultAuth authentication domain. For information about authentication, see "Understanding Authentication in the SAS Intelligence Platform" in *SAS Intelligence Platform: Security Administration Guide*.

MAPPATH="*location*"
specifies the location within the metadata server for the information map that you want to create, open, or delete. After the connection is made, the location is stored so that you do not need to specify it again on subsequent statements such as NEW INFOMAP, UPDATE INFOMAP, DELETE INFOMAP, SAVE, or EXPORT. However, if you do specify a location on a subsequent statement in the same PROC INFOMAPS step, then that location overrides the stored location.

Alias: PATH=

METACREDENTIALS=YES|NO
specifies whether the user ID and password specified in the METAUSER= and METAPASS= system options are retrieved and used to connect to the metadata server when the METAUSER= and METAPASS= options for the PROC INFOMAPS statement are omitted.

By default, or when METACREDENTIALS=YES is specified, the system option values are used if they are available when the corresponding options for the PROC INFOMAPS statement are omitted. Specify METACREDENTIALS=NO to prevent the INFOMAPS procedure from using the system option values.

A typical situation in which you would specify METACREDENTIALS=NO is when the code containing the INFOMAPS procedure is being executed on a workspace server or stored process server. In such cases, the METAUSER= and METAPASS= system options contain a one-time user ID and password that have already been used by the server. A new one-time password must be generated in this situation. Specifying METACREDENTIALS=NO enables a connection to be established under the identity of the client user using a new one-time password.

Default: YES

METAPASS="*password*"
specifies the password that corresponds to the user ID that connects to the metadata server.

You can use the METAPASS= system option to specify a default password for connecting to the metadata server for the SAS session. See the METAPASS= system option in the *SAS Language Interfaces to Metadata*.

If your metadata server supports single sign-on, you can omit the METAPASS= and METAUSER= options and connect through a trusted peer connection or through Integrated Windows Authentication. For more information, see the *SAS Intelligence Platform: Security Administration Guide*.

Alias: PASSWORD= | PW=

Examples:

```
metapass="My Password"
metapass="MyPassword"
```

METAPORT=*port-number*
specifies the TCP port that the metadata server is listening to for connections.

If this option is not specified, the value is obtained from the METAPORT= system option. See the METAPORT= system option in the *SAS Language Interfaces to Metadata*.

Alias: PORT=

Example: `metaport=8561`

METASERVER="*address*"
specifies the network IP (Internet Protocol) address of the computer that hosts the metadata server.

If this option is not specified, the value is obtained from the METASERVER= system option. See the METASERVER= system option in the *SAS Language Interfaces to Metadata*.

Alias: SERVER= | HOST=

Example: `metaserver="myip.us.mycompany.com"`

METAUSER="*user-ID*"
specifies the user ID to connect to the metadata server. The user ID is not case sensitive.

You can use the METAUSER= system option to specify a default user ID for connecting to the metadata server for the SAS session. See the METAUSER= system option in the *SAS Language Interfaces to Metadata*.

If your metadata server supports single sign-on, you can omit the METAUSER= and METAPASS= options and connect through a trusted peer connection or through Integrated Windows Authentication. For more information, see the *SAS Intelligence Platform: Security Administration Guide*.

Alias: USER= | USERID= | ID=

Example: `metauser="myUserID"`

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 the Windows environment, then you must fully qualify the user ID by specifying the domain or machine name that you specified when your login object was created in the metadata server. For example, `metauser="Windows-domain-name\user-ID"`.

Example

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

DELETE INFOMAP Statement

Deletes an information map from the SAS folders tree.

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

Required Argument

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

Option

MAPPATH="*location*"
specifies the location within the SAS folders tree for the information map to delete.

Interaction: A location in the DELETE statement overrides a location specified in a PROC INFOMAPS statement.

Examples

```
delete infomap "my testmap"
   mappath="/Users/myUserID/My Folder";

delete infomap "myMap";
```

EXPORT Statement

Exports an information map in its XML representation.

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

Required Argument

FILE=*fileref* | *"physical-location"*
specifies an external file to which to export an XML representation of the information map. If the external file already exists, it is replaced.

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 the INFOMAPS procedure or SAS Information Map Studio to import it correctly.

Options

INFOMAP *"information-map-name"*
specifies the name of the information map to export. If you do not specify the INFOMAP option, the current information map is exported.

MAPPATH=*"location"*
specifies the location within the SAS folders tree for the information map to export. Exporting fails if you specify an information map name in the EXPORT statement but no location has been specified. The location from which the information map is exported is determined according to the following order of precedence:

1. The MAPPATH specified in the EXPORT statement
2. The MAPPATH specified in the NEW INFOMAP or UPDATE INFOMAP statement
3. The MAPPATH specified in the PROC INFOMAPS statement

Examples

```
/* Export an information map to a physical location. */
/* Note that the sample locations are operating system-specific. */
export infomap "my testmap"
   file="c:\test\test.xml"
   mappath="/Users/myUserID/My Folder";
```

```
/* Export an information map to a fileref. */
filename xmlfile "c:\test\test.xml";
export infomap "my testmap"
   file=xmlfile
   mappath="/Users/myUserID/My Folder";
```

IMPORT Statement

Imports an information map from an external XML file.

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

Required Argument

FILE=*fileref* |"*physical-location*"
specifies the fileref or physical location 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 it to be imported correctly.

Details

After importing an information map, you must issue a SAVE statement to save it. If you specify a name in 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 SAS folders tree.

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

1. The MAPPATH specified in the SAVE statement
2. The MAPPATH specified in the NEW INFOMAP or UPDATE INFOMAP statement
3. The MAPPATH specified in the PROC INFOMAPS statement

CAUTION:
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. △

Example

```
/* Create a new information map from an external file. */
import file="c:\test\test.xml";
save infomap "myMap"
   mappath="/Users/myUserID/My Folder";
```

INSERT DATAITEM Statement

Inserts a data item into the current information map.

INSERT DATAITEM
COLUMN="*data-source-ID*"."*column-name*" | EXPRESSION="*expression-text*" |
HIERARCHY="*dimension*"."*hierarchy*" | MEASURE="*OLAP-measure*"
<ACTIONS=(*actions-list*)>
<AGGREGATION=*aggregate-function*>
<AGGREGATIONS_DROP_LIST=(*aggregate-function-list*)>
<AGGREGATIONS_KEEP_LIST=(*aggregate-function-list*)>
<CLASSIFICATION=CATEGORY | MEASURE>
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<FOLDER=*folder-name* | "*folder-location*" </CREATE>>
<FORMAT="*format-name*">
<ID="*data-item-ID*">
<NAME="*data-item-name*">
<TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP>
<VALUE_GENERATION=NONE | DYNAMIC | *custom-values-list*>;

Required Argument

COLUMN="*data-source-ID*"."*column-name*"
specifies a column. The *data-source-ID* is the identifier of a data source in the current information map. It must match the identifier of the table that contains the 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 argument applies only to a relational data source.

Interaction: You can specify only one of the COLUMN=, EXPRESSION=, HIERARCHY=, or MEASURE= arguments in an INSERT DATAITEM statement.

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= argument, 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 about the EXPCOLUMNLEN= option, see "Other LIBNAME Statement Options for the Information Maps Engine" on page 71 and "EXPCOLUMNLEN= Data Set Option" on page 76. △

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 data 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. You can refer explicitly to the current information map by specifying **root** as the qualifier. 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: You can specify only one of the COLUMN=, EXPRESSION=, HIERARCHY=, or MEASURE= arguments in an INSERT DATAITEM statement.

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

HIERARCHY="*dimension*"."*hierarchy*"

specifies a physical hierarchy. The *dimension* is the name of a dimension in the current OLAP data source. The *hierarchy* is the name of a hierarchy that is defined in the specified *dimension*. For example:

```
insert datasource sasserver="SASMain" cube="Simba";
insert dataitem hierarchy="MARKET"."GEOGRAPHICAL" id="Geographical";
```

The INFOMAPS procedure inserts a data item for this hierarchy into the information map.

By default, a data item inserted using the HIERARCHY= argument returns the top-level members of the hierarchy when used in a query. If you want the data item to return members from other levels, you should instead define it with the EXPRESSION= argument. In the following example, the data item Geographical1 returns the top-level members of the GEOGRAPHICAL hierarchy (for example, REGION), while the data item Geographical2 returns all members of the level.

```
insert dataitem hierarchy="MARKET"."GEOGRAPHICAL" id="Geographical1";
insert dataitem expression="<<MARKET.GEOGRAPHICAL>>"
   id="Geographical2"
   type=character;
```

If the GEOGRAPHICAL hierarchy contains another level named STATE and you want a data item to return members from the STATE level, then you should use the EXPRESSION option to create that data item. For example,

```
insert dataitem expression="<<MARKET.GEOGRAPHICAL.STATE>>.members"
   id="State"
   type=character;
```

Restriction: This option applies only to an OLAP data source.

Interaction: You can specify only one of the COLUMN=, EXPRESSION=, HIERARCHY=, or MEASURE= arguments in an INSERT DATAITEM statement.

MEASURE="*measure*"
specifies a physical measure. The *measure* is the name of a measure that is defined in the measures dimension in the OLAP data source for the current information map. For example:

```
insert datasource sasserver="SASMain" cube="SASMain - OLAP schema".Simba;
insert dataitem measure="ACTUALAVE" id="Average Actual";
```

The INFOMAPS procedure inserts a data item for this OLAP measure into the information map.

Restriction: This option applies only to an OLAP data source.

Interaction: You can specify only one of the COLUMN=, EXPRESSION=, HIERARCHY=, or MEASURE= arguments in an INSERT DATAITEM statement.

Options

ACTIONS=(*actions-list*)
tells an application (such as SAS Web Report Studio) that uses the information map what actions it can present to its users to perform on the result data set returned by the information map. For example, a user of SAS Web Report Studio can right-click a column heading of a report and select **Sort** from the pop-up menu to sort the values in that column. Specifying **actions=(nosort)** tells SAS Web Report Studio not to offer the **Sort** menu for this data item.

The following actions can be specified:

RANK | NORANK
specifies whether the following items can be ranked:

- relational data item values
- members of OLAP data items that represent hierarchies

The setting for this option does not affect the ability of the information map consumer to rank row and column values in a generated result set.

Default: RANK

SORT | NOSORT
specifies whether the following items can be sorted:

- relational data item values
- members of OLAP data items that represent hierarchies

The setting for this option does not affect the ability of the information map consumer to sort OLAP data values.

Default: SORT

FILTER | NOFILTER
specifies whether members of OLAP data items that represent hierarchies can have filters applied to them. The setting for this option does not affect the ability of the information map consumer to filter on row and column values in a generated result set, and it does not affect test queries that are run from the Test the Information Map dialog box in Information Map Studio.

Default: FILTER

Restriction: This option value applies only to non-measure OLAP data items.

NAVIGATE | NONAVIGATE
specifies whether the member of OLAP data items that represent hierarchies can be drilled up or down, or expanded and collapsed.

Default: NAVIGATE

Restriction: This option value applies only to non-measure OLAP data items.

Default: The action is enabled (RANK, SORT, FILTER, or NAVIGATE) unless it is specifically disabled (NORANK, NOSORT, NOFILTER, or NONAVIGATE).

Example: `ACTIONS=(RANK SORT NOFILTER NONAVIGATE)`

AGGREGATION=*aggregate-function*
specifies how a measure data item is aggregated when it is used in a query. Values for the AGGREGATION= option are shown in Table 2.1 on page 18. For more information about the aggregate functions (except for **`InternalAggregation`** and **`InternalAggregationAdditive`**), see "Summarizing Data: Using Aggregate Functions" in the "Retrieving Data from a Single Table" chapter in the *SAS 9.2 SQL Procedure User's Guide*.

The special value **`InternalAggregation`** specifies that the values of the measure data item are aggregated by a non-additive expression. A non-additive expression is one for which the arithmetic summation of the aggregated values of the measure data item is not equal to the arithmetic summation of all of the detail values of that data item. For example, **`SUM(COL1)/COUNT(COL1)`** is a non-additive expression. If you specify that a data item has a non-additive expression, then the total for that data item is calculated by applying the specified expression to the detail values of the data item.

The special value **`InternalAggregationAdditive`** specifies that values of the measure data item are aggregated by an additive expression. An additive expression is one for which the arithmetic summation of the aggregated values of the measure data item is equal to the arithmetic summation of all of the detail values of that data item. For example, **`SUM(COL1*COL2)`** is an additive expression.

Restriction: The AGGREGATION= option applies only to relational data items that are measures.

Restriction: If the data item is defined by an expression that references a measure data item or that contains an aggregate function, then the only valid values for the AGGREGATION= option are **`InternalAggregation`** or **`InternalAggregationAdditive`**.

Default: If you do not specify an AGGREGATION= option, then the default aggregate function is defined as follows:

SUM
if the expression type for the data item is numeric

COUNT
if the expression type for the data item is character, date, time, or timestamp

InternalAggregation
if the data item is based on a measure

If the default function is not available, then the first aggregate function in the list of functions that SAS maintains becomes the default.

Interaction: If you use the AGGREGATION= option in the same INSERT DATAITEM statement as either the AGGREGATIONS_DROP_LIST= or the AGGREGATIONS_KEEP_LIST= option, then the INFOMAPS procedure sets the AGGREGATIONS_DROP_LIST= or the AGGREGATIONS_KEEP_LIST= option first.

Table 2.1 Aggregate Functions

Function	Definition	Available to nonnumeric item that is a measure
AVG	average (mean) of values	
AvgDistinct	average (mean) of distinct values	
COUNT	number of nonmissing values	✓
CountDistinct,	number of distinct nonmissing values	✓
CountPlusNMISS	number of values (including the number of missing values)	✓
CountPlusNMISSDistinct	number of distinct values (including the number of distinct missing values)	✓
CSS	corrected sum of squares	
CSSDistinct	corrected sum of squares of distinct values	
CV	coefficient of variation (percent)	
CVDistinct	coefficient of variation (percent) of distinct values	
FREQ	number of nonmissing values	✓
FreqDistinct	number of distinct nonmissing values	✓
InternalAggregation	defined in an expression (non-additive)	✓
InternalAggregationAdditive	defined in an expression (additive)	✓
MAX	largest value	✓
MEAN	mean (average) of values	
MeanDistinct	mean (average) of distinct values	
MIN	smallest value	✓
N	number of nonmissing values	✓
NDistinct	number of distinct nonmissing values	✓
NMISS	number of missing values	✓
NMISSDistinct	number of distinct missing values	✓

Function	Definition	Available to nonnumeric item that is a measure
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	

AGGREGATIONS_DROP_LIST=(*aggregate-function-list*)
removes one or more functions from the set of aggregate functions available to a data item. See Table 2.1 on page 18 for information about aggregate functions.

Separate multiple aggregate functions in the list with a blank space. For example:

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

Note: Use AGGREGATIONS_DROP_LIST= if there are only a few aggregate functions that you want excluded from the total set. Use AGGREGATIONS_KEEP_LIST= if there are only a few aggregate functions that you want included. △

Restriction: This option applies only to relational data items that are measures.

Default: If you specify neither AGGREGATIONS_DROP_LIST= nor AGGREGATIONS_KEEP_LIST=, then all of the valid aggregate functions for the data item are available.

Interaction: If you use the AGGREGATIONS_DROP_LIST= option in the same INSERT DATAITEM statement as the AGGREGATION= option, then the INFOMAPS procedure sets the AGGREGATIONS_DROP_LIST= option first.

AGGREGATIONS_KEEP_LIST=(*aggregate-function-list*)
specifies the aggregate functions that are available to a data item. Functions not listed in *aggregate-function-list* are excluded. See Table 2.1 on page 18 for information about aggregate functions.

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

```
AGGREGATIONS_KEEP_LIST=(Freq FreqDistinct CSSDistinct)
```

Note: Use AGGREGATIONS_KEEP_LIST= if there are only a few aggregate functions that you want included. Use AGGREGATIONS_DROP_LIST= if there are only a few aggregate functions that you want excluded from the total set. △

Restriction: This option applies only to relational data items that are measures.

Default: If you specify neither AGGREGATIONS_DROP_LIST= nor AGGREGATIONS_KEEP_LIST=, then all of the valid aggregate functions for the data item are available.

Interaction: If you use the AGGREGATIONS_KEEP_LIST= option in the same INSERT DATAITEM statement as the AGGREGATION= option, then the INFOMAPS procedure sets the AGGREGATIONS_KEEP_LIST= option first.

CLASSIFICATION=CATEGORY | MEASURE
specifies whether the data item is a category or a measure. The classification of the data item determines how it is processed in a query. A data item that is a measure can be used in computations or analytical expressions. A data item that is a category is used to group measures using an applied aggregate function.

If you do not specify the CLASSIFICATION= option, the INFOMAPS procedure assigns a default classification based on the following:

- the contents of the expression if the EXPRESSION= argument is used
- the data type of the physical data if the COLUMN=, HIERARCHY=, or MEASURE= argument is used

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

For an OLAP data source, if the HIERARCHY= argument is used, then the default classification is CATEGORY. If the MEASURE= argument is used, then the default classification is MEASURE. If the EXPRESSION= argument is used, then the default classification is MEASURE if the specified TYPE= value is NUMERIC. Otherwise, the default classification is CATEGORY.

CUSTOM_PROPERTIES=(*custom-properties-list*)
specifies additional properties for the data item. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
 ...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the data item, then the INSERT DATAITEM statement will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.
For example

```
CUSTOM_PROPERTIES=(
                    ("MA_Level" "Nominal" "Descriptive text goes here.")
                    ("MA_UseInSubjectIdTop" "Subject_ID_" "Subject ID")
                  )
```

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

Alias: DESC=

Restriction: Although you can specify more than 256 characters for the data item description, SAS programs can use only the first 256 characters of the description.

FOLDER="*folder-name*" | "*folder-location*" </CREATE>
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 without an initial slash. For example, **FOLDER="CUSTOMERS"**.
- If the folder is not in the root directory, then you must specify the location of the folder beginning with a slash. For example, **FOLDER="/CUSTOMERS/Europe"**.

/CREATE
specifies that the named folder or location is created automatically if it does not already exist.

Alias: LOCATION=

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

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

Restriction: A folder name can contain blank spaces, but it 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, or if you specify an empty string as the format value, the INFOMAPS procedure sets 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 or business resource referenced in the data item expression
- the expression type of the data item

Restriction: The FORMAT= option applies only to relational data items and OLAP measures.

ID="*data-item-ID*"
specifies the ID assigned to the data item being inserted. The ID is a value that uniquely identifies 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=, HIERARCHY=, or MEASURE= argument.
  - If the data item is inserted from a physical column in one of the following ways, then the ID is generated from either the SAS name or label of the physical column:

    INSERT DATAITEM with the COLUMN= argument specified

    INSERT DATASOURCE with either the _ALL_ or the COLUMNS= option specified

    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 data sources, filters, and relationships.
  - If the data item is inserted with the HIERARCHY= or MEASURE= option, then the ID is generated from the caption of the hierarchy or measure.

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 are invalid:

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

If a name contains any of these characters, they are replaced with an underscore (_) when the ID is generated from the name.

Restriction: The first 32 characters of an ID must be unique across an information map. An error occurs if you specify an ID that is the same as an existing ID (data item, data source, filter, or other). An ID that differs only by case from another ID in the current information map is not considered unique.

NAME="*data-item-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 refer to a data item in code using its ID rather than its name. If you do not specify a name, the name defaults to one of the following, depending on how the data item is defined:

- If the COLUMN= argument is used, then the name defaults to the column name or column label (based on the setting of the USE_LABELS= option from the NEW INFOMAP or UPDATE INFOMAP statement).
- If the EXPRESSION= argument is used, then the INFOMAPS procedure provides a default name.
- If the HIERARCHY= or MEASURE= argument is used, then the name defaults to the caption of the hierarchy or measure.

Restriction: There is no limit on the length of the name of a relational data item. OLAP data item names cannot contain more than 245 characters.

Restriction: A data item name can contain blank spaces, but it cannot consist only of blank spaces. Nulls and non-blank nonprintable characters are not valid characters in a data item name. A data item name can contain the following special characters, but they are 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=, HIERARCHY=, or MEASURE= option, then you can omit the TYPE= option. In this case, the INFOMAPS procedure derives the type from the type of the corresponding data.

VALUE_GENERATION=NONE | DYNAMIC | (*custom-values-list*)
specifies what method an application (for example, SAS Web Report Studio) that uses the information map is to use in generating a list of data item values for this data item to present to a user when the user is constructing a filter or responding to a prompt. The following value generation methods can be specified:

NONE
specifies that the list of values should not be generated. The application will require its user to manually type data item values.

DYNAMIC
specifies that the list that contains all of the data item's values be dynamically generated. The list is generated by querying the data source to retrieve the data item's values.

custom-values-list
defines a custom list of values for the data item. The form of the *custom-values-list* is

```
("unformatted-value-1" <"formatted-value-1">)
...
("unformatted-value-n" <"formatted-value-n">)
```

where

unformatted-value
specifies the unformatted value for a report.

formatted-value
specifies the formatted value for a report.

Note: The formatted value is optional. It is used for display purposes only. For example, SAS Web Report Studio displays these values to the user of a filter and prompt definition dialog boxes so that the user can see what the values will look like after they are formatted for a report. △

Note: To refer to a custom value later during an update, you must specify the unformatted value rather than the formatted value.

Example:

```
VALUE_GENERATION=(
                  ("CA" "California")
```

```
        ("NC")
        ("NY" "New York")
    )
```

Examples

```
/* Use the COLUMN= option to insert a data item for a physical column. */
insert dataitem column="TRANSACTION"."Sales_Amount"
   id="Total_Sales";
```

INSERT DATASOURCE Statement

Makes the data from either a table or cube available to the current information map.

INSERT DATASOURCE
SASSERVER="*application-server-name*"
TABLE="*library*"."*table*" <COLUMNS=(*column–1* <... *column-n*>) | _ALL_ >
<DESCRIPTION="*descriptive-text*">
<ID="*data-source-ID*">
<NAME="*data-source-name*">;

or:

INSERT DATASOURCE
SASSERVER="*application-server-name*"
CUBE=<"*schema*".>"*cube*" <_ALL_>
<DESCRIPTION="*descriptive-text*">
<ID="*data-source-ID*">
<NAME="*datasource-name*">;

Required Arguments

CUBE=<"*schema*".>"*cube*"
identifies an OLAP cube as a data source for the current information map.
A cube must be both of the following:

- registered in the currently connected metadata server
- associated with a schema that is registered in the SAS OLAP Server specified by the SASSERVER= option

  Note: A SAS OLAP Server can have only one schema. A schema lists the available cubes. △

Restriction: If you use the CUBE= argument in an INSERT DATASOURCE statement, then you cannot use the TABLE= argument in any INSERT DATASOURCE statement in the same PROC INFOMAPS step. Although you can access either relational data or cube data, you cannot access both types within the same information map.

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

Restriction: Cube names are case sensitive.

SASSERVER="*application-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 both of the following:

- registered in the currently connected metadata server
- 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 server. 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 in the **`General`** tab in the properties window for the table.
- From SAS Data Integration Studio, delete the duplicate tables.

Alias: SERVER=

Restriction: If you use the TABLE= argument in an INSERT DATASOURCE statement, then you cannot use the CUBE= argument in any INSERT DATASOURCE statement in the same PROC INFOMAPS step. Although you can access either relational data or cube data, you cannot access both types 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.

Restriction: Table names are case sensitive.

Options

NAME="*data-source-name*"
enables you to specify a descriptive name for each data source inserted in an information map. If you use the INFOMAPS procedure to insert multiple data sources from the same physical table, the data sources will, by default, all have the same name. When you view the data sources in SAS Information Map Studio, they are indistinguishable because the names are used as identifiers in the graphical user interface. Use the NAME= option to customize the name for each data source.

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-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.

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

Alias: DESC

Restriction: Although you can specify more than 256 characters for the data source description, SAS programs can use only the first 256 characters of the description.

ID="*data-source-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 are invalid:

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

If a name contains any of these characters, they are replaced with an underscore (_) when the ID is generated from the name.

Restriction: Table and cube names are case sensitive.

Restriction: The first 32 characters of an ID must be unique across an information map. An error occurs if you specify an ID that is the same as an existing ID (data item, data source, filter, or other). An ID that differs only by case from another ID in the current information map is not considered unique.

Details

An inserted data source is a logical representation of a table or cube that you can query via the information map. An OLAP data source and the cube that it references have the same set of properties. A relational data source has properties that are not part of the referenced table. You can insert multiple tables as data sources into an information map. A table can be inserted as a data source multiple times in the same information map. Each of these data sources has a unique ID and its own set of properties. 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.

To view a list of all the data sources in the current information map, use the LIST DATASOURCES statement. Even though the data source name and its ID have the same value by default, you can use the ID= option to specify a different ID or use the NAME= option to assign a different name.

Examples

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

/* Insert only three 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="SASMain - 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
CONDITION="*conditional-expression*"
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<FOLDER="*folder-name*" | "*folder-location*" </CREATE>>
<ID="*filter-ID*">
<NAME="*filter-name*">;

Required Argument

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

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 data 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. You can refer explicitly to the current information map by specifying **root** as the qualifier. 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

CUSTOM_PROPERTIES=(*custom-properties-list*)
specifies additional properties for the filter. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the filter, then the INSERT FILTER will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

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

Alias: DESC=

FOLDER="*folder-name*" | "*folder-location*" </CREATE>
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, without an initial slash. For example, **FOLDER="CUSTOMERS"**.
- If the folder is not in the root directory, then you must specify the location of the folder beginning with a slash. For example, **FOLDER="/CUSTOMERS/Europe"**.

/CREATE
specifies that the named folder or location is created automatically if it does not already exist.

Alias: LOCATION=

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

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

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

ID="*filter-ID*"
specifies the ID of the filter to insert. If you do not specify an ID, the INFOMAPS procedure generates a unique ID from the filter name. You can use the LIST statement to display filter IDs.

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

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

If a name contains any of these characters, they are replaced with an underscore (_) when the ID is generated from the name.

Restriction: The first 32 characters of an ID must be unique across the information map. An error occurs if you specify an ID that is the same as an existing ID (data item, data source, filter, or other). An ID that differs only by case from another ID in the current information map is not considered unique.

NAME=*"filter-name"*
specifies the name of a filter to insert into the current information map. If the NAME= option is missing from the INSERT FILTER statement, the INFOMAPS procedure will generate a default name.

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

Examples

```
/* Insert a relational table filter. */
insert filter
   name="genderFilter"
   id="Boys"
   description="Filter for boys"
   folder="/Filters" /create
   condition='<<CLASS.sex>> = "M"';

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

/* Insert an MDX filter. */
insert filter
   name="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 folder into the current information map.

INSERT FOLDER "*folder-name*"
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<LOCATION="*parent-folder-name*" | "*parent-folder-location*" </CREATE>>;

Required Argument

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

Tip: When referring to the folder, remember that case is important.

Options

CUSTOM_PROPERTIES=(*custom-properties-list*)
specifies additional properties for the folder. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the folder, then the INSERT FOLDER statement will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

DESCRIPTION="*descriptive-text*"
specifies the description of the folder that is created.

Alias: DESC=

LOCATION="*parent-folder-name*" | "*parent-folder-location*" </CREATE>
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 without an initial slash. For example, **`FOLDER="CUSTOMERS"`**.
- If the parent folder is not in the root directory, then you must qualify it with a location that starts with a slash. For example, **`FOLDER="/CUSTOMERS/Europe"`**.

/CREATE
specifies that the named folder or location is created automatically if it does not already exist.

Alias: PARENT=

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 it cannot consist only of blank spaces.

Examples

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

INSERT RELATIONSHIP Statement

Inserts a join into the current information map.

INSERT RELATIONSHIP
CONDITION="*conditional-expression*"
LEFT_TABLE="*data-source-ID-1*"
RIGHT_TABLE="*data-source-ID-2*"
<CARDINALITY=ONE_TO_ONE | ONE_TO_MANY | MANY_TO_ONE | MANY_TO_MANY | UNKNOWN>
<CUSTOM_PROPERTIES=(*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<ID="*relationship-ID*">
<JOIN=INNER | LEFT | RIGHT | FULL>;

Required Arguments

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

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 (<< >>).

LEFT_TABLE="*data-source-ID-1*"
specifies the data source ID of the first table in the relationship.

RIGHT_TABLE="*data-source-ID-2*"
specifies the data source ID of the second table in the relationship.

Options

CARDINALITY=ONE_TO_ONE | ONE_TO_MANY | MANY_TO_ONE | MANY_TO_MANY | UNKNOWN
describes the relationship between rows in the first data source and rows in the second data source.

Default: If the CARDINALITY= option is not specified, then the cardinality defaults to UNKNOWN.

CUSTOM_PROPERTIES=(*custom-properties-list*)
specifies additional properties for the relationship. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
 ...
 ("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the relationship, then the INSERT RELATIONSHIP statement will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

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

Alias: DESC

ID="*relationship-ID*"
specifies the ID of the relationship to be inserted. If you do not specify an ID, the INFOMAPS procedure generates a unique ID.

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

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

If a name contains any of these characters, they are replaced with an underscore (_) when the ID is generated from the name.

Restriction: The first 32 characters of an ID must be unique across the information map. An error occurs if you specify an ID that is the same as an existing ID (data item, data source, filter, or other). An ID that differs only by case from another ID in the current information map is not considered unique.

JOIN=INNER | LEFT | RIGHT | FULL
specifies the type of join.

INNER
returns all the rows in one table that have one or more matching rows in the other table

LEFT
returns all the rows in the specified left table, plus the rows in the specified right table that match rows in the left table

RIGHT
returns all the rows in the specified right table, plus the rows in the specified left table that match rows in the right table

FULL
returns all the rows in both tables

Default: INNER

Details

The INSERT RELATIONSHIP statement applies only to relational tables. If a join already exists between the specified tables, then the new join replaces the old one, unless a new and unique ID is specified.

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

Example

```
insert relationship
   left_table="CUSTOMER"
   right_table="TRANSACTION"
   condition="(<<CUSTOMER.Cust_ID>>=<<TRANSACTION.Cust_ID>>)"
   join=inner
   id="join_customer_to_transaction";
```

LIST Statement

Lists the key properties 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 |RELATIONSHIPS | _ALL_>;

Options

DATAITEMS
lists the properties of all the data items defined in the current information map. The properties include the name, ID, folder location, description, expression text, expression type, classification, format, and the default aggregation (if the classification is a measure) 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, ID, folder location, description, and the conditional expression text of each filter.

RELATIONSHIPS
lists the properties of all the relationships that are defined in the current information map. The properties include the ID, left table, right table, cardinality, join type, and the join expression text.

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

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

Example

The following output shows the result of submitting the LIST statement for the information map in Chapter 8, "Example: Using the INFOMAPS Procedure and the Information Maps Engine," on page 87.

Output 2.1 Log the LIST Statement

```
1/* List all of the properties of the current information map. */
2   list;

Total datasources: 3

    Data source: HR.EMPINFO
    ID: Empinfo
    Name: EMPINFO
    Description:

    Data source: HR.JOBCODES
    ID: Jobcodes
    Name: JOBCODES
    Description:

    Data source: HR.SALARY
    ID: Salary
    Name: SALARY
    Description:

Total data items: 9

    Data item name: Annual Salary
    ID: Annual Salary
    Folder: /Salary Info
    Description: Physical column SALARY
    Expression: <<Salary.Salary>>
    Expression type: NUMERIC
    Classification: MEASURE
    Format: DOLLAR12.
    Default aggregation: Sum

    Data item name: Department Code
    ID: Dept_code
    Folder: /
    Description:
    Expression: SUBSTRN(<<root.Jobcode>>, 1, 3)
    Expression type: CHARACTER
    Classification: CATEGORY
    Format:
```

```
Data item name: Division
ID: Division
Folder: /
Description: Physical column DIVISION
Expression: <<Empinfo.DIVISION>>
Expression type: CHARACTER
Classification: CATEGORY
Format:

Data item name: Enddate
ID: Enddate
Folder: /Salary Info
Description: Physical column ENDDATE
Expression: <<Salary.ENDDATE>>
Expression type: DATE
Classification: CATEGORY
Format: DATE9.

Data item name: Identification Number
ID: Identification Number
Folder: /
Description: Physical column IDNUM
Expression: <<Empinfo.Identification Number>>
Expression type: NUMERIC
Classification: CATEGORY
Format: SSN11.

Data item name: Jobcode
ID: Jobcode
Folder: /
Description: Physical column JOBCODE
Expression: <<Empinfo.JOBCODE>>
Expression type: CHARACTER
Classification: CATEGORY
Format:

Data item name: Location
ID: Location
Folder: /
Description: Physical column LOCATION
Expression: <<Empinfo.LOCATION>>
Expression type: CHARACTER
Classification: CATEGORY
Format:

Data item name: Monthly Salary
ID: Monthly Salary
Folder: /Salary Info
Description:
Expression: <<Salary.Salary>>/12
Expression type: NUMERIC
Classification: CATEGORY
Format: DOLLAR12.

Data item name: Title
ID: Title
Folder: /
Description: Physical column TITLE
Expression: <<Jobcodes.TITLE>>
Expression type: CHARACTER
Classification: CATEGORY
Format: $F20.
```

```
Total filters: 4

    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

Total Relationships: 2

    Relationship ID: JOIN_10
    Left table: HR.EMPINFO
    Right table: HR.JOBCODES
    Cardinality: UNKNOWN
    Join type: INNER
    Join expression: (<<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>>)

    Relationship ID: JOIN_11
    Left table: HR.EMPINFO
    Right table: HR.SALARY
    Cardinality: UNKNOWN
    Join type: INNER
    Join expression: (<<Empinfo.Identification Number>>=<<Salary.Identification Number>>)

90   run;
```

MOVE DATAITEM Statement

Moves one or more data items to a new location.

MOVE DATAITEM "*data-item-ID*" | ID_LIST=("*data-item-ID-1*" <...
"*data-item-ID-n*">)
NEW_LOCATION="*new-folder-location*" </CREATE>;

Required Arguments

"data-item-ID" | ID_LIST=("*data-item-ID-1*" <... "*data-item-ID-n*">)
specifies the data items to move. You can specify a single data item or you can use the ID_LIST= option to specify multiple data items.

Tip: The data items specified in the ID_LIST do not have to reside in the same folder.

NEW_LOCATION="*new-folder-location*" </CREATE>
specifies the new location for the folder.

/CREATE
specifies that the named folder or location is created automatically if it does not already exist.

Example

```
/* Move the data item "Name" from the current folder */
/* to the "newFolder" folder. If the "newFolder" folder */
/* does not exist, then create it. */
move dataitem "Name" new_location="newFolder" /create;
```

MOVE FILTER Statement

Moves one or more filters to a new location.

MOVE FILTER "*filter-ID*" | ID_LIST=("*filter-ID-1*" <... "*filter-ID-n*">)
NEW_LOCATION="*new-folder-location*" </CREATE>;

Required Arguments

"filter-ID" | ID_LIST=("*filter-ID-1*" <... "*filter-ID-n*">)
specifies the filters to move. You can specify a single filter ID or you can use the ID_LIST= option to specify multiple filter names.

Tip: The filters specified in the ID_LIST= argument do not have to reside in the same folder.

NEW_LOCATION="*new-folder-location*" </CREATE>
specifies the new location.

/CREATE
specifies that the named folder or location is created automatically if it does not already exist.

Example

```
/* Move the filters "over60", "over40", and "over20" from */
/* the current folder to the "/Employees/AgeGroups" folder. */
/* If the "/Employees/AgeGroups" folder does not */
/* exist, then create it. */
move filter id_list=("over60" "over40" "over20")
   new_location= "/Employees/AgeGroups" /create;
```

MOVE FOLDER Statement

Moves a folder to a new location.

MOVE FOLDER *"folder-name"*
NEW_LOCATION=*"new-folder-location"* </CREATE>
<LOCATION=*"current-folder-location"*>;

Required Arguments

"folder-name"
specifies the name of the folder to move.

NEW_LOCATION="*new-folder-location*" </CREATE>
specifies the new location for the folder.

/CREATE
specifies that the named folder or location is created automatically if it does not already exist.

Option

LOCATION="*current-folder-location*"
specifies the current location of the folder to move. If you do not specify a location, then the default is the root folder location.

Example

```
/* Move the "myCompany" folder from the */
/* "NC" folder to the "CA" folder. */
move folder "myCompany" location="/State/NC" new_location="/State/CA";
```

NEW INFOMAP Statement

Creates a new information map.

NEW INFOMAP "*information-map-name*"
<AUTO_REPLACE=YES | NO>
<CREATE_TARGET_FOLDER=YES | NO>
<CUSTOM_PROPERTIES= (*custom-properties-list*)>
<DESCRIPTION="*descriptive-text*">
<INIT_CAP=YES | NO>
<MAPPATH="*location*" </CREATE>>
<REPLACE_UNDERSCORES=YES | NO>
<USE_LABELS=YES | NO>
<VERIFY=YES | NO>;

Required Argument

"*information-map-name*"
specifies the name of the new information map.

Restriction: The following characters are not valid in information map names:

- null characters
- non-blank nonprintable characters

Restriction: An information map name can contain blank spaces, but it cannot contain leading or trailing blank spaces and cannot consist of only blank spaces.

Restriction: Information map names can be up to 60 characters long. However, if you plan to access the information map in SAS programs using the Information Maps engine, then you should specify a name with no more than 32 characters, which is the maximum length for SAS names.

Options

AUTO_REPLACE=YES | NO
indicates whether the specified information map is automatically replaced if it already exists. If the AUTO_REPLACE= option is set to YES and the information map already exists, then the existing information map is replaced with a new empty information map. If the AUTO_REPLACE= option is set to NO and the information map already exists, then an error occurs.

Default: NO

CREATE_TARGET_FOLDER=YES | NO
specifies whether to automatically create a folder when inserting all data items from a data source. Specifying YES automatically creates a folder when you subsequently insert data items using an INSERT DATASOURCE statement that specifies the _ALL_ option. The ID of the data source is used as the name of the folder. All of 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

CUSTOM_PROPERTIES= (*custom-properties-list*)
specifies additional properties for an information map. The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">")
 ...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. It is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

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

Alias: DESC=

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="*location*" </CREATE>
specifies the location within the SAS folders tree for the new information map. The location is required unless a location has been specified in the PROC INFOMAPS statement.

/CREATE
specifies that the location is created automatically, if it does not already exist.

Alias: LOCATION=

Interaction: The location from the NEW INFOMAP statement overrides the location 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.

VERIFY=YES|NO
specifies whether the INFOMAPS procedure verifies the validity of data items, filters, and relationships in subsequent insert or update operations. Setting the VERIFY option to NO improves performance, but doing so introduces the risk that invalid data items, filters, or relationships could get saved into an information map.

Default: YES

Details

The NEW INFOMAP statement creates a new information map. When you open an information map that does not yet exist, the INFOMAPS procedure allocates space in memory for its creation. After that, you can start inserting business data into the copy of the information map in memory. Save the information map with a SAVE statement to write the in-memory copy to the SAS folders tree.

Only one information map can be created at a time. If you submit one NEW INFOMAP statement, you must save the new information map with a SAVE statement before submitting another NEW INFOMAP, UPDATE INFOMAP, or IMPORT statement. If you do not save the in-memory copy, it is not written to the SAS folders tree and is simply lost.

Example

```
new infomap "my testmap"
     mappath="/Users/myUserID/My Folder"
     verify=no
     description="Map for Domestic Customers";
```

SAVE Statement

Saves the current information map.

SAVE
<INFOMAP "*information-map-name*">
<MAPPATH="*location*" </CREATE>>;

Options

INFOMAP "*information-map-name*"
specifies the name to use for saving the current information map.

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

MAPPATH="*location*" </CREATE>
specifies the location within the SAS folders tree where the information map is to be saved.

/CREATE
specifies that the location is created automatically, if it does not already exist.

Alias: LOCATION=

Default: If you do not specify a location, the default is determined according to the following order of precedence:

1. The MAPPATH specified in the NEW INFOMAP or UPDATE INFOMAP statement
2. The MAPPATH specified in the PROC INFOMAPS statement

Example

```
/* Save the current information map in the location specified  */
/* when it was opened (or in the PROC INFOMAPS statement)    */
/* using the name 'myMap'  */
save infomap "myMap";

/* Save the current information map in the specified location using  */
/* its current name */
save mappath="/Users/myUserID/My Folder";

/* Save the current information map in the specified location using  */
/* the  name 'myMap'  */
save infomap "myMap" mappath="/Users/myUserID/My Folder";
```

SET STORED_PROCESS Statement

Associates a stored process with the current information map.

SET STORED_PROCESS

NAME="*stored-process-name*"
<LOCATION="*stored-process-location*">;

Required Argument

NAME="*stored-process-name*"
specifies the name of the stored process that is to be associated with the information map. If the stored process name is a null or blank string, then no stored process is associated with the information map.

Interaction: When the name of the specified stored process is a null string ("") or contains only blank spaces, then the value for the LOCATION= option is ignored.

Option

LOCATION="*stored-process-location*"
specifies the location within the SAS folders tree of the stored process that is associated with the current information map.

UPDATE DATAITEM Statement

Updates the properties of a specified data item in the current information map.

UPDATE DATAITEM "*data-item-ID*"
<ACTIONS=(*actions-list*)>
<AGGREGATION=*aggregate-function*>
<AGGREGATIONS_LIST=(_ALL_ | ADD (*aggregate-function-list*) |
REPLACE (*aggregate-function-list*) | REMOVE (*aggregate-function-list*)>
<CLASSIFICATION=CATEGORY | MEASURE>
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<EXPRESSION="*expression-text*">
<FORMAT="*format-name*">
<ID="*new-data-item-ID*">
<NAME="*data-item-name*">
<TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP>
<VALUE_GENERATION=NONE | DYNAMIC | ADD (*custom-value-list*) |
REPLACE (*custom-value-list*) | REMOVE (*unformatted-value-list*)>;

Required Argument

"*data-item-ID*"
specifies the ID of the data item to update.

Options

ACTIONS=(*actions-list*)
tells a SAS application (such as SAS Web Report Studio) that uses the information map what actions it can present to its users to perform on the result data set returned by the information map.

The following actions can be specified:

RANK | NORANK
specifies whether the following items can be ranked:

- relational data item values.
- members of OLAP data items that represent hierarchies. The setting for this option does not affect the ability of the information map consumer to rank row and column values in a generated result set.

The setting for this option does not affect the ability of the information map consumer to rank row and column values in a generated result set.

Default: RANK

SORT | NOSORT
specifies whether the following items can be sorted:

- relational data item values.
- members of OLAP data items that represent hierarchies. The setting for this option does not affect the ability of the information map consumer to sort OLAP data values.

The setting for this property does not affect the ability of the information map consumer to sort OLAP data values.

Default: SORT

FILTER | NOFILTER
specifies whether members of OLAP data items that represent hierarchies can have filters applied to them. The setting for this option does not affect the ability of the information map consumer to filter on row and column values in a generated result set, and it does not affect test queries that are run from the Test the Information Map dialog box in Information Map Studio.

Default: FILTER

Restriction: This option value applies only to non-measure OLAP data items.

NAVIGATE | NONAVIGATE
specifies whether the member of OLAP data items that represent hierarchies can be drilled up or down, or expanded and collapsed.

Default: NAVIGATE

Restriction: This option value applies only to non-measure OLAP data items.

Default: If an action is not specified with an UPDATE DATAITEM statement, then it remains as originally specified with the INSERT DATAITEM statement. By default an action is enabled unless it is specifically disabled.

Interaction: The ACTIONS= option replaces the specified action or actions but does not affect any other actions that are in effect.

AGGREGATION=*aggregate-function*
specifies how a measure data item is aggregated when it is used in a query. See Table 2.1 on page 18 for a list of functions and what types of data they are available to. For more information about the aggregate functions (except for **`InternalAggregation`** and **`InternalAggregationAdditive`**), see “Summarizing

Data: Using Aggregate Functions" in the "Retrieving Data from a Single Table" chapter in the *SAS 9.2 SQL Procedure User's Guide*.

The special value **`InternalAggregation`** specifies that the values of the measure data item are aggregated by a non-additive expression. A non-additive expression is one for which the arithmetic summation of the aggregated values of the measure data item is not equal to the arithmetic summation of all of the detail values of that data item. For example, **`SUM(COL1)/COUNT(COL1)`** is a non-additive expression. If you specify that a data item has a non-additive expression, then the total for that data item is calculated by applying the specified expression to the detail values of the data item.

The special value **`InternalAggregationAdditive`** specifies that values of the measure data item are aggregated by an additive expression. An additive expression is one for which the arithmetic summation of the aggregated values of the measure data item is equal to the arithmetic summation of all of the detail values of that data item. For example, **`SUM(COL1*COL2)`** is an additive expression.

Interaction: If you use the AGGREGATION= option in the same UPDATE statement as the AGGREGATIONS_LIST= option, then the INFOMAPS procedure sets the AGGREGATIONS_LIST= option first.

AGGREGATIONS_LIST=_ALL_ | ADD (*aggregate-function-list*) | REPLACE (*aggregate-function-list*) | REMOVE (*aggregate-function-list*)

modifies the list of aggregation functions that are available to the data item.

Interaction: If you use the AGGREGATION= option in the same UPDATE statement as the AGGREGATIONS_LIST= option, then the INFOMAPS procedure sets the AGGREGATIONS_LIST= option first.

The following actions can be specified:

ALL
places all the aggregate functions that are valid for the data item in the aggregation list.

ADD (*aggregate-function-list*)
adds the specified aggregate functions to the aggregation list. See Table 2.1 on page 18 for information about aggregate functions.

REPLACE (*aggregate-function-list*)
replaces the current aggregation list with the specified aggregate functions.

REMOVE (*aggregate-function-list*)
removes the specified aggregate functions from the aggregation list.

Requirement: Separate aggregate function names in *aggregate-function-list* values with a blank space. For example:

```
AGGREGATIONS_LIST=REPLACE(Freq FreqDistinct CSSDistinct)
```

Interaction: You can specify two AGGREGATIONS_LIST= options in the same UPDATE DATAITEM statement if one specifies _ALL and the other specifies REMOVE or if one specifies ADD and the other specifies REMOVE. If you specify both the _ALL_ and REMOVE operations, then the _ALL_ operation occurs first. If you specify both the ADD and REMOVE operations, then the REMOVE operation occurs first.

CLASSIFICATION=CATEGORY | MEASURE

specifies the usage type of the data item to be updated.

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)
specifies how custom properties for the data item are updated. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. Valid operations are the following:

NONE
removes all custom properties from the data item, if there are any.

ADD (*custom-properties-list*)
adds the specified custom properties to the data item.
The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
  ...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the data item, then the UPDATE DATAITEM statement will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

REPLACE (*custom-properties-list*)
replaces the current custom properties for the data item with the specified properties.
See the ADD operation for a description of the form of the *custom-properties-list*.

REMOVE (*property-names-list*)
removes the specified custom properties from the data item.
The form of the *property-names-list* is

```
"property-name-1" <... "property-name-n">
```

Interaction: If you specify both the ADD and REMOVE operations, then the REMOVE operation occurs first.

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

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 that contains character type data items that are 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 about the EXPCOLUMNLEN= option, see "Other LIBNAME Statement Options for the Information Maps Engine" on page 71 and "EXPCOLUMNLEN= Data Set Option" on page 76. △

Interaction: Changing the expression of an existing data item might cause changes in other property settings within the same data item.

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

ID="*new-data-item-ID*"
specifies the new ID for the data item.

NAME="*data-item-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.

TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP
specifies the data type of the data item's expression.

Interaction: Changing the type of an existing data item might cause changes in other property settings within the same data item.

VALUE_GENERATION=NONE | DYNAMIC | ADD (*custom-values-list*) | REPLACE (*custom-values-list*) | REMOVE (*unformatted-values-list*)
specifies what method an application (for example, SAS Web Report Studio) that uses the information-map is to use in generating a list of data item values for this data item to present to a user when the user is constructing a filter or responding to a prompt. Here are the value-generation methods that can be specified:

NONE
specifies that the list of values should not be generated. The application will require its user to manually type data item values.

DYNAMIC
specifies that the list that contains all of the data item's values be dynamically generated. The list is generated by querying the data source to retrieve the data item's values.

ADD (*custom-values-list*)
adds the specified custom values to the data item.

The form of the *custom-values-list* is

```
("unformatted-value-1" <"formatted-value-1" >)
  ...
("unformatted-value-n" <"formatted-value-n" >)
```

where

unformatted-value
specifies the unformatted value for a report.

formatted-value
specifies the formatted value for a report.

Note: The formatted value is optional. It is used for display purposes only. For example, SAS Web Report Studio displays these values to the user of a filter and prompt definition dialog boxes so that the user can see what the values will look like after they are formatted for a report. △

REPLACE (*custom-values-list*)
replaces the current custom values for the data item.

See the ADD operation for a description of the form of the *custom-values-list*.

REMOVE (*unformatted-values-list*)
removes the specified unformatted values and their associated formatted values from the custom values list for the data item.

The form of the *unformatted-values-list* is

```
"unformatted-value-1" <... "unformatted-value-n">
```

Interaction: If you specify both the ADD and REMOVE operations, then the REMOVE operation occurs first.

Example

```
update dataitem "custId"
       classification=category
       actions=(rank sort)
       value_generation=add(
                            ("NC" "North Carolina")
                            ("VA" "Virgina")
                            ("MD" "Maryland")
                           )
       value_generation=remove("CA" "OR" "WA");
```

UPDATE DATASOURCE Statement

Updates the properties of a data source in the current information map.

UPDATE DATASOURCE "*data-source-ID*"
<NAME="*data-source-name*">
<DESCRIPTION="*descriptive-text*">
<ID="new-*data-source-ID*">;

Required Argument

"*data-source-ID*"
specifies the ID of the data source to update.

Options

NAME="*data-source-name*"
specifies a new name for the data source. When you change the name, you create a logical representation of the physical data source. You are not duplicating any physical data.

DESCRIPTION="*descriptive-text*"
specifies the description of the data source.

ID="*new-data-source-ID*"
specifies the new ID for the data source.

Example

```
update datasource "Customer"
       name="Customer_US"
       description="Customers from the US";
```

UPDATE FILTER Statement

Updates the properties of a specified filter in the current information map.

UPDATE FILTER *"filter-ID"*
<CONDITION="*conditional-expression*">
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<ID="*new-filter-ID*">
<NAME="*filter-name*">;

Required Argument

"filter-ID"
specifies the ID of the filter to update.

Options

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

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)
specifies how custom properties for the filter are updated. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. Valid operations are the following:

NONE
removes all custom properties from the filter, if there are any.

ADD (*custom-properties-list*)
adds the specified custom properties to the filter.
The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the filter, then the UPDATE FILTER statement will fail. Therefore,

it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

REPLACE (*custom-properties-list*)
replaces the current custom properties for the filter with the specified custom propertied.
See the ADD operation for a description of the form of the *custom-properties-list*.

REMOVE (*property-names-list*)
removes the specified custom properties from the filter.
The form of the *property-names-list* is

```
"property-name-1" <... "property-name-n">
```

Interaction: If you specify both the ADD and REMOVE operations, then the REMOVE operation occurs first.

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

ID="*new-filter-ID*"
specifies the new ID for the filter.

NAME="*filter-name*"
specifies the name assigned to the filter in the information map.

Examples

```
update filter "ageFilter"
   condition="<<Class.Age>> = 10"
   description="Ten years old only"
   name="Age Filter";
```

UPDATE FOLDER Statement

Updates the properties of a folder in the current information map.

UPDATE FOLDER "*folder-name*"
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) |
REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<LOCATION="*current-parent-folder-name*" | "*current-parent-folder-location*">
<NAME="*new-folder-name*">;

Required Argument

"folder-name"
specifies the name of the map folder to be updated.

Options

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)
specifies how custom properties for the folder are updated. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. Valid operations are the following:

NONE
removes all custom properties from the folder, if there are any.

ADD (*custom-properties-list*)
adds the specified custom properties to the folder.
The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
  ...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the folder, then the UPDATE FOLDER statement will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

REPLACE (*custom-properties-list*)
replaces the current custom properties for the folder with the specified properties. See the ADD operation for a description of the form of the *custom-properties-list*.

REMOVE (*property-names-list*)
Removes the specified custom properties from the folder.
The form of the *property-names-list* is

```
"property-name-1" <... "property-name-n">
```

Interaction: If you specify both the ADD and REMOVE operations, then the REMOVE operation occurs first.

DESCRIPTION="*descriptive-text*"
specifies the description of the folder.

LOCATION="*current-parent-folder-name*" | "*current-parent-folder-location*"
specifies the current parent folder of the folder that you are updating.

- If the folder is in the root directory of the information map, then you can specify the folder by name without an initial slash. For example, **LOCATION="CUSTOMERS"**.
- If the parent folder is not in the root directory, then you must qualify it with a location that starts with a slash. For example, **LOCATION="/CUSTOMERS/Europe"**.

Restriction: The root folder cannot be updated.

NAME="*new-folder-name*"
specifies the new name of the folder.

Example

```
update folder "subsubMeasures" location="/measures/subMeasures"
```

UPDATE INFOMAP Statement

Updates an existing information map.

UPDATE INFOMAP "*information-map-name*"
<CREATE_TARGET_FOLDER=YES | NO>
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<INIT_CAP=YES | NO>
<MAPPATH="*location*">
<REPLACE_UNDERSCORES=YES | NO>
<USE_LABELS=YES | NO>
<VERIFY=YES | NO>;

Required Argument

"*information-map-name*"
specifies the name of the information map to update.

Restriction: If the specified information map does not exist, an error will occur.

Options

CREATE_TARGET_FOLDER=YES | NO
specifies whether to automatically create a folder when inserting all items from a data source. Specifying YES automatically creates a folder when you subsequently insert all data items using an INSERT DATASOURCE statement that specifies the _ALL_ option. 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

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)
specifies how custom properties for the information map are updated. Valid operations are the following:

NONE
removes all the existing custom properties from the information map, if there are any.

ADD (*custom-properties-list*)
adds the specified custom properties to the information map.
The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
   ...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the information map, then the UPDATE INFOMAP statement will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

REPLACE (*custom-properties-list*)
replaces the custom properties for the information map with the specified custom properties.
See the ADD operation for a description of the form of the *custom-properties-list*.

REMOVE (*properties-names-list*)
removes the specified custom properties from the information map.
The form of the *properties-names-list* is

```
"property-name-1" <... "property-name-n">
```

Interaction: If you specify both the ADD and REMOVE operations, then the REMOVE operation occurs first.

DESCRIPTION=“*descriptive-text*”
specifies the description of the information map, which can be viewed by the information map consumer.

Alias: DESC=

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="*location*"
specifies the location within the SAS folders tree for the information map to open or update. The location is required unless a location has been specified in the PROC INFOMAPS statement.

Alias: LOCATION=

Interaction: The location from the UPDATE INFOMAP statement overrides the location 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.

VERIFY=YES|NO
specifies whether the INFOMAPS procedure verifies the validity of data items, filters, and relationships during the update operation. Setting the VERIFY option to NO improves performance, but doing so introduces the risk that invalid data items, filters, or relationships could get saved into an information map.

Default: YES

Details

When you use the UPDATE INFOMAP statement to open an existing information map, the INFOMAPS procedure creates a working copy of the information map in memory. You must submit a SAVE statement to save the working copy before you terminate the INFOMAPS procedure or open a different information map by submitting an IMPORT, NEW INFOMAP, or UPDATE INFOMAP statement. If you do not submit a SAVE statement, then any changes that you have made to the working copy of the information map are lost.

Example

```
update infomap "my testmap"
      mappath="/Users/myUserID/My Folder"
```

```
verify=no
description="Map for Domestic Customers";
```

UPDATE RELATIONSHIP Statement

Updates the properties of a specified join relationship in the current information map.

UPDATE RELATIONSHIP "*relationship-ID*"
<CARDINALITY=ONE_TO_ONE | ONE_TO_MANY | MANY_TO_ONE | MANY_TO_MANY | UNKNOWN>
<CONDITION="*conditional-expression*">
<CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)>
<DESCRIPTION="*descriptive-text*">
<ID="*new-relationship-ID*">
<JOIN=INNER | LEFT | RIGHT | FULL>;

Required Argument

"relationship-ID"
specifies the ID of the relationship to update.

Options

CARDINALITY=ONE_TO_ONE | ONE_TO_MANY | MANY_TO_ONE | MANY_TO_MANY | UNKNOWN
specifies the cardinality of the relationship.

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

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 <<...>>.

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REPLACE (*custom-properties-list*) | REMOVE (*property-names-list*)
specifies how custom properties for the relationship are updated. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. Valid operations are the following:

NONE
removes all custom properties from the relationship, if there are any.

ADD (*custom-properties-list*)
adds to a custom list of values for the relationship.

The form of the *custom-properties-list* is

```
("property-name-1" "property-value-1" <"description-1">)
...
("property-name-n" "property-value-n" <"description-n">)
```

where

property-name
specifies the name of the property.

Requirement: Property names must be unique. If a property name already exists in the relationship, then the INSERT RELATIONSHIP statement will fail. Therefore, it is recommended that you add a prefix or suffix to the property name to ensure uniqueness.

Restriction: Property names cannot begin with an underscore (_) character.

property-value
specifies the value of the property.

description
specifies the description of the property. The description is optional.

REPLACE (*custom-properties-list*)
replaces a custom list of values for the relationship.
See the ADD operation for a description of the form of the *custom-properties-list*.

REMOVE (*property-names-list*)
removes names from a custom properties list.
The form of the *property-names-list* is

```
"property-name-1" <... "property-name-n">
```

Interaction: If you specify both the ADD and REMOVE operations, then the REMOVE operation occurs first.

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

Alias: DESC

ID="*new-relationship-ID*"
specifies the new ID for the relationship.

JOIN=INNER | LEFT | RIGHT | FULL
specifies the type of join.

INNER
returns all the rows in one table that have one or more matching rows in the other table

LEFT
returns all the rows in the specified left table, plus the rows in the specified right table that match rows in the left table

RIGHT
returns all the rows in the specified right table, plus the rows in the specified left table that match rows in the right table

FULL
returns all the rows in both tables

Example

```
update relationship "join_customer_to_transaction"
       condition="(<<CUSTOMER.Cust_ID>>=<<TRANSACTION.Cust_ID>>)"
       join=inner;
update relationship "CustTransaction"
       cardinality=one_to_one
       join=left;
```

```
update relationship "join"
       join=inner cardinality= one_to_many
       condition= "(<<CUSTOMER.Cust_ID>>=<<TRANSACTION.Cust_ID>>)
                  and (<<CUSTOMER.Cust_ID>> > 142673939)";
```

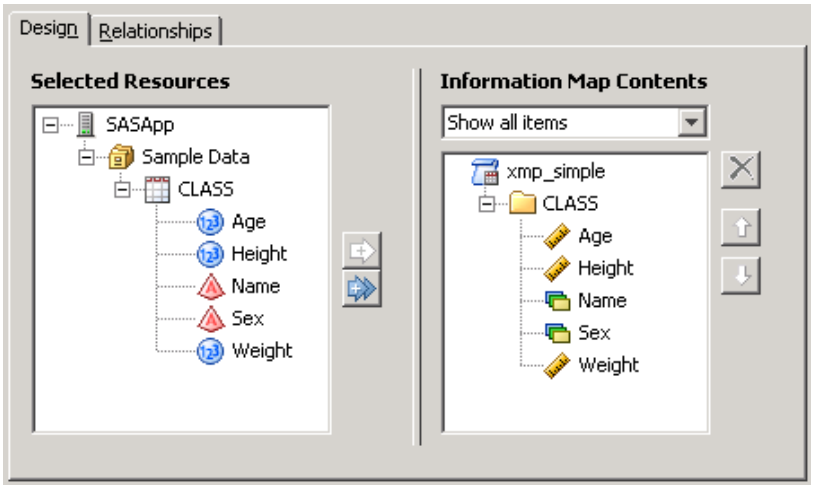
Examples: INFOMAPS Procedure

Example 1: Creating a Basic Information Map

The following example shows you how to use the INFOMAPS procedure to create an information map.

```
proc infomaps metauser="pubhdc1\sasdemo"
   metapass="Pwd123"
   metaserver="pubhdc1.na.sas.com"
   metaport=8561;

/* Open a new information map.  The specified location is */
/* where, by default, the information map is saved when a */
/* SAVE statement issued. The information map exists only */
/* in memory until a SAVE statement is issued.            */
new infomap "xmp_simple"
    mappath="/Users/sasdemo/My Folder"
    auto_replace=yes;

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

/* Save the information map that is currently open.  Because   */
/* no location is specified in the SAVE statement, it is saved */
/* in the location specified in the NEW INFOMAP 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 INSERT DATASOURCE statement includes the _ALL_ option.



Example 2: Creating an Information Map with Relationships and Filters

The following example shows:

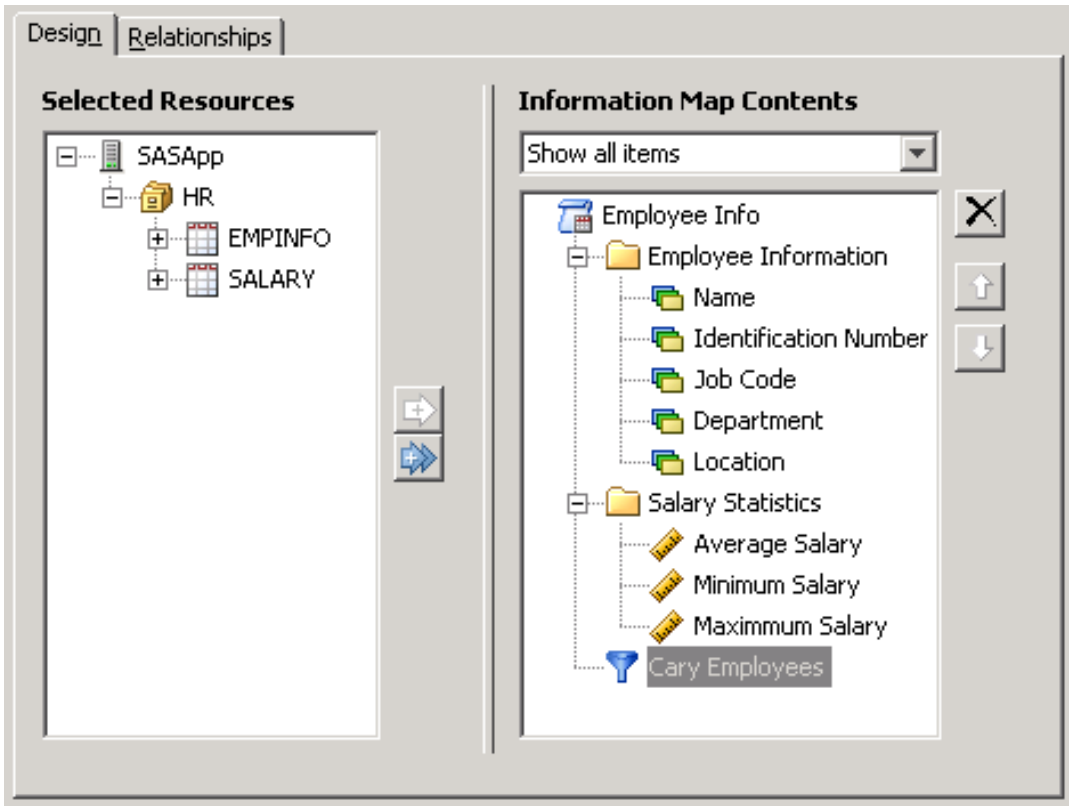
- how to create a relationship to link two data sources
- how to explicitly create folders with the INSERT FOLDER statement, and how to insert data items into the folders with the INSERT DATAITEM statement
- how to create a filter that can be used in queries to subset a data item

```
proc infomaps metauser="pubhdc1\sasdemo"
   metapass="Pwd123"
   metaserver="pubhdc1.na.sas.com"
   metaport=8561;

/* Open a new information map.  The specified location is */
/* where, by default, the information map is saved when a */
/* SAVE statement issued. The information map exists only */
/* in memory until a SAVE statement is issued.            */
new infomap "Employee Info"
   mappath="/Users/sasdemo/My Folder"
   auto_replace=yes;

/* Make the Employee Information table accessible. */
insert datasource sasserver="SASApp"
   table="HR"."EMPINFO"
   id="EmployeeInfo";

/* Make the Salary Information table accessible. */
insert datasource sasserver="SASApp"
   table="HR"."SALARY"
   id="SalaryInfo";

/* Create a relationship to link the data sources. */
insert relationship
   id="join_empinfo_to_salary"
```

```
   left_table="EmployeeInfo"
   right_table="SalaryInfo"
   cardinality=one_to_one
   condition="<<EmployeeInfo.IDNUM>>=<<SalaryInfo.IDNUM>>";

 /* Create folders for data items. */
insert folder "Employee Information";
insert folder "Salary Statistics";

/* Create data items. */
insert dataitem
   column="EmployeeInfo"."NAME"
   folder="Employee Information";

insert dataitem
   column="EmployeeInfo"."IDNUM"
   folder="Employee Information"
   classification=category;

insert dataitem
   column="EmployeeInfo"."JOBCODE"
   folder="Employee Information"
   name="Job Code";

insert dataitem
   column="EmployeeInfo"."DEPTCODE"
   folder="Employee Information"
   name="Department";

insert dataitem
   column="EmployeeInfo"."LOCATION"
   folder="Employee Information";

insert dataitem
   column="SalaryInfo"."SALARY"
   folder="Salary Statistics"
   name="Average Salary"
   aggregations_keep_list=("AVG")
   format="dollar12.2";

insert dataitem
   column="SalaryInfo"."SALARY"
   folder="Salary Statistics"
   name="Minimum Salary"
   aggregations_keep_list=("MIN")
   format="dollar12.2";

insert dataitem
   column="SalaryInfo"."SALARY"
   folder="Salary Statistics"
   name="Maximmum Salary"
   aggregations_keep_list=("MAX")
   format="dollar12.2";
```

```
 /* Create a filter for the Location data item. */
insert filter
   name="Cary Employees"
 description="Employees who work in Cary, NC"
 condition="<<EmployeeInfo.LOCATION>>='Cary'";

/* Save the information map that is currently open.  Because   */
/* no location is specified in the SAVE statement, it is saved */
/* in the location specified in the NEW INFOMAP statement.     */
save;
run;
```

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



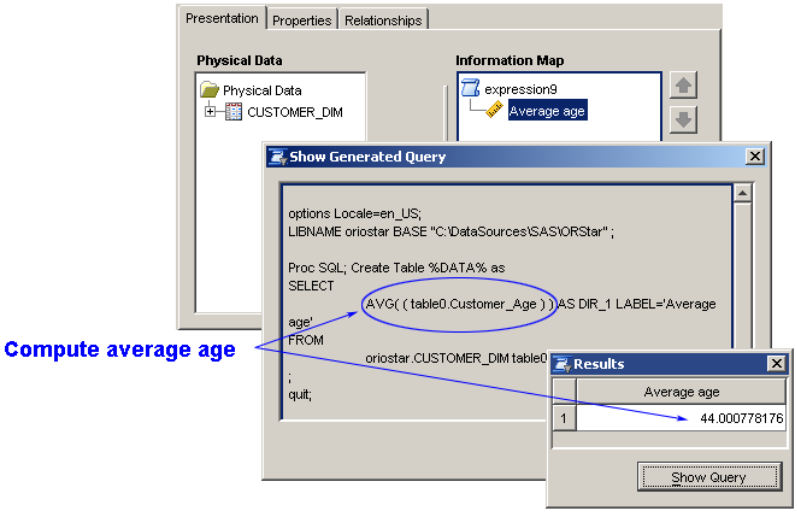
Example 3: Aggregating a Data Item

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

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

new  infomap "expression9"
     mappath="/Users/myUserID/My Folder";
```

```
/* 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.



CHAPTER

3

Using the SAS Information Maps LIBNAME Engine

What Does the Information Maps Engine Do? **63**
Understanding How the Information Maps Engine Works **63**
Advantages of Using the Information Maps Engine **67**
What Is Required to Use the Information Maps Engine? **67**
What Is Supported? **67**

What Does the Information Maps Engine Do?

An information map is a collection of data items and filters that describes and provides a view of data that business users understand. The SAS Information Maps LIBNAME 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 can only read information maps. It cannot write to or update them, nor can it modify the underlying data. If you want to update an existing information map, you can use the INFOMAPS procedure. For more information, see Chapter 2, "The INFOMAPS Procedure," on page 5. If you have SAS Information Map Studio software, 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 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 mymaps infomaps metauser=myUserID
                        metapass=myPassword
                        metaserver="myserver.mycompany.com"
```

```
                          metaport=8561
                          mappath="/Users/myUserID/My Folder";
```

Output 3.1 Output from the LIBNAME Statement

```
1  libname mymaps infomaps metauser=myUserID
2                          metapass=XXXXXXXXXX
3                          metaserver="myserver.mycompany.com"
4                          metaport=8561
5                          mappath="/Users/myUserID/My Folder";

NOTE: Libref MYMAPS was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /Users/myUserID/My Folder
```

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:

```
/* Use the Information Maps engine to retrieve the data. */
libname mymaps infomaps metauser=myUserID
                        metapass=myPassword
                        metaserver="myserver.mycompany.com"
                        metaport=8561
                        mappath="/Users/myUserID/My Folder";

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

/* 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=mymaps.'Employee Statistics Sample'n;
run;

/* Print 5 observations from the data that the information map references. */
proc print data=mymaps.'Employee Statistics Sample'n (obs=5
   filter=('Cary Employees'n));
run;
```

Output 3.2 Log for the Example Program

```
1  /* Run the Information Maps engine to retrieve the data. */
2  libname mymaps infomaps metauser=myUserID
3                          metapass=XXXXXXXXXX
4                          metaserver="myserver.mycompany.com"
5                          metaport=8561
6                          mappath="/Users/myUserID/My Folder";
NOTE: Libref MYMAPS was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /Users/myUserID/My Folder
7
8  /* Display a list of available information maps. */
9  proc datasets lib=mymaps;
                                        Directory

                           Libref         MYMAPS
                           Engine         INFOMAPS
                           Physical Name  /Users/myUserID/My Folder

                                                    Member
                             #  Name                 Type

                             1  Employee Statistics Sample  DATA
10 run;
11 quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.06 seconds
      cpu time            0.00 seconds

12
13 /* Allow mixed-case letters and blank spaces in information map names. */
14 option validvarname=any;
15
16 /* View the data items, including any filters, in the information map. */
17 proc contents data=mymaps.'Employee Statistics Sample'n;
18 run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           0.09 seconds
      cpu time            0.00 seconds

19
20 /* Print 5 observations from the data that the information map references. */
21 proc print data=mymaps.'Employee Statistics Sample'n (obs=5
22    filter=('Cary Employees'n));
23 run;

NOTE: There were 5 observations read from the data set MYMAPS.'Employee Statistics Sample'n.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.23 seconds
      cpu time            0.11 seconds
```

Output 3.3 Output from the CONTENTS and PRINT Procedures

```
                                  The SAS System

                                The CONTENTS Procedure

      Data Set Name        MYMAPS.'Employee Statistics Sample'n    Observations          .
      Member Type          DATA                                    Variables             11
      Engine               INFOMAPS                                Indexes               0
      Created              .                                       Observation Length    0
      Last Modified        .                                       Deleted Observations  0
      Protection                                                   Compressed            NO
      Data Set Type                                                Sorted                NO
      Label                                                        Filters               1
      Data Representation  Default
      Encoding             Default


                          Alphabetic List of Variables and Attributes

 # Variable            Type Len Format     Label

 4 Deptcode            Char   3            Physical column EMPINFO.DEPTCODE
10 Hire Date           Num    8 DATE9.     Hire date
 2 Identification      Num    8 SSN11.     Identification Number
   Number
 3 Jobcode             Char   8            Physical column EMPINFO.JOBCODE
 5 Location            Char   8            Physical column EMPINFO.LOCATION
 1 Name                Char  32 $32.       NAME
11 Number of Years     Num    8 COMMA6.    The number of years that the employee
   Employed                               has been employed by the company.
 7 Salary2             Num    8 DOLLAR12.2 Salary
 8 Salary3             Num    8 DOLLAR12.2 Salary
 9 Salary4             Num    8 DOLLAR12.2 Salary
 6 Salary_2            Num    8 DOLLAR12.2 Salary


                                  Information Maps

           FilterName         FilterType  FilterDesc

           Cary Employees     Unp         Employees who work in Cary, North Carolina.
                                  The SAS System


                                          Identification
     Obs   Name                                 Number        Jobcode    Deptcode    Location

       1   Bryan, Lynne C.                     000-00-0381      VID002       VID         Cary
       2   Fissel, Ronald T.                   000-00-0739      QA0005       QA0         Cary
       3   White, Frank P.                     000-00-1575      DPD003       FAC         Cary
       4   Winfree, Ambrose Y.                 000-00-1579      CCD001       CCD         Cary
       5   Blue, Kenneth N.                    000-00-1637      MIS004       QA0         Cary

                                                                                  Number of
                                                                                    Years
     Obs       Salary_2        Salary2        Salary3        Salary4   Hire Date   Employed

       1    $183,000.00    $183,000.00    $183,000.00    $183,000.00   08APR1984        25
       2     $85,000.00     $85,000.00     $85,000.00     $85,000.00   02FEB1985        24
       3     $69,000.00     $69,000.00     $69,000.00     $69,000.00   01JUN1984        24
       4    $100,000.00    $100,000.00    $100,000.00    $100,000.00   14JUN1989        19
       5     $18,000.00     $18,000.00     $18,000.00     $18,000.00   12NOV1991        17
                                                                  15:51 Monday, November 3, 2008   1

NOTE: WHERE were 277 observations read from the data set SAMPDATA.EMPINFO.LOCATION='Cary    ';
```

For information about improving the performance of the Information Maps engine, see "Hints and Tips for Using the Information Maps Engine" on page 84.

Advantages of Using the Information Maps Engine

Using the Information Maps engine provides the following advantages:

- The engine is the only way for Base SAS software to access data generated from an information map.
- 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.

What Is Required to Use the Information Maps Engine?

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

- access to the metadata server that contains the metadata definition for the data and information maps
- information maps that are defined in a metadata server
- access to the server where the physical data is located

What Is Supported?

The Information Maps engine can only 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 server to process the information map. Using the engine and SAS code, you can do the following:

- read data that is retrieved via 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 of the data that is retrieved via the information map is passed back to the SAS client. The SAS client applies the WHERE clause to restrict the data for the result set.

  Performance is degraded whenever a large number of observations have to be processed by SAS. You can use information map filters to restrict the query and reduce the number of observations that must be processed. A filter contains criteria for subsetting data in an information map. For more information about filters, see "FILTER= Data Set Option" on page 77.
- 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.

CHAPTER

4 LIBNAME Statement for the Information Maps Engine

Using the LIBNAME Statement **69**
LIBNAME Statement Syntax **69**
Required Arguments **69**
LIBNAME Statement Options for Connecting to the SAS Metadata Server **70**
Other LIBNAME Statement Options for the Information Maps Engine **71**

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 server. The engine reads information maps and uses their metadata to access underlying data.

You must have a metadata server available that contains metadata that defines the information maps to be accessed. For the necessary server 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 metadata server, see the *SAS Intelligence Platform: System Administration Guide*.

LIBNAME Statement Syntax

LIBNAME *libref* **INFOMAPS** MAPPATH="*location*" <*options*>;

Required Arguments

libref
is a SAS name that refers to the metadata server library to be accessed. A libref cannot exceed eight characters. For additional rules for SAS names, refer to *SAS Language Reference: Dictionary*.

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

MAPPATH="*location*"
specifies the path to the location of the information maps within the metadata server.

The path is hierarchical with the slash (/) as the separator character. For example, **mappath="/Users/myUserID/My Folder"**.

Alias: PATH=

LIBNAME Statement Options for Connecting to the SAS Metadata Server

The following LIBNAME statement options establish a connection to the metadata server.

DOMAIN="*authentication-domain*"
specifies an authentication domain in the metadata server that is associated with the user ID and password. If you do not specify an authentication domain, then the user ID and password are associated with the **DefaultAuth** authentication domain. For information about authentication, see "Understanding Authentication in the SAS Intelligence Platform" in *SAS Intelligence Platform: Security Administration Guide*.

Alias: AUTHDOMAIN=

METACREDENTIALS=YES|NO
specifies whether the user ID and password specified in the METAUSER= and METAPASS= system options are retrieved and used to connect to the metadata server when the METAUSER= and METAPASS= options for the LIBNAME statement are omitted.

By default, or when METACREDENTIALS=YES is specified, the system option values are used if they are available when the LIBNAME statement does not provide the corresponding options. Specify METACREDENTIALS=NO to prevent the Information Maps engine from using the system option values.

A typical situation in which you would specify METACREDENTIALS=NO is when the code containing the LIBNAME statement is being executed on a workspace server or stored process server. In such cases, the METAUSER= and METAPASS= system options contain a one-time user ID and password that have already been used by the server. A new one-time password must be generated in this situation. Specifying METACREDENTIALS=NO enables a connection to be established under the identity of the client user using a new one-time password.

Default: YES

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), then enclosing the identifier in quotation marks is optional.

If your metadata server supports single sign-on, you can omit the METAPASS= and METAUSER= options and connect through a trusted peer connection or through Integrated Windows Authentication. For more information, see the *SAS Intelligence Platform: Security Administration Guide*.

You can use the METAPASS= system option to specify a default password for connecting to the metadata server for the SAS session. For information about the METAPASS= system option, see *SAS Language Interfaces to Metadata*.

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. For information about the METAPORT= system option, see *SAS Language Interfaces to Metadata*.

Alias: PORT=

METASERVER="*address*"
specifies the network IP (Internet Protocol) address of the computer that hosts the metadata server. For example, **metaserver="myip.mycompany.com"**. Enclosing the identifier in quotation marks is optional.

If this option is not specified, the value is obtained from the METASERVER= system option. For information about the METASERVER= system option, see *SAS Language Interfaces to Metadata*.

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) or a backslash character, enclosing the identifier in quotation mark is optional.

You can use the METAUSER= system option to specify a default user ID for connecting to the metadata server for the SAS session. For information about the METAUSER= system option, see *SAS Language Interfaces to Metadata*.

If your metadata server supports single sign-on, you can omit the METAUSER= and METAPASS= options and connect through a trusted peer connection or through Integrated Windows Authentication. For more information, see the *SAS Intelligence Platform: Security Administration Guide*.

Alias: USER= | USERID= | ID=

Restriction: In the metadata server, you must have at least one login definition 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 Server. For example, **metauser="*domain-name**user-ID*"**.

SSPI=YES | NO
specifies whether Integrated Windows Authentication is used. Integrated Windows Authentication is a mechanism for a Windows client and server to exchange credentials without the user having to explicitly specify them. For more information, see "Integrated Windows Authentication" in *SAS Intelligence Platform: Security Administration Guide*.

Default: NO

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.

AGGREGATE=YES | NO
specifies whether detailed data or aggregated data is retrieved from the data source.

YES
specifies that aggregated data is retrieved.

NO
specifies that detailed data is retrieved.

Specify YES in order to see data items that do not support detailed data.

Default: NO

EXPCOLUMNLEN=*integer*
specifies the length of the SAS character column when a data item defined with an expression is encountered.

Note: The Information Maps engine also supports a EXPCOLUMNLEN= data set option that can be used to change this option setting during a DATA step when the Information Maps engine is used. The 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 specified when the libref was created. For more information, see the "EXPCOLUMNLEN= Data Set Option" on page 76. △

Default: 32

PRESERVE_MAP_NAMES=YES | NO

YES
specifies that information map names are read with special characters. 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 information map names are derived from SAS member names by using SAS member-name normalization. When you use SAS to retrieve 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. The DATASETS procedure reports the number of information maps that cannot be displayed because their names cannot be normalized, as shown in the following example:

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

This note is not displayed when you view information maps in the SAS Explorer window.

The SAS Explorer window displays information map names in capitalized form when PRESERVE_MAP_NAMES=NO. These information map names follow the SAS member-name normalization rules and might not represent the actual case of the information map name.

Default: YES

READBUFF=*integer*
specifies the number of rows to hold in memory for input into SAS. The value must be positive.

Choosing the optimum value for the READBUFF= option requires a detailed knowledge of the data that is returned from the information map and of the environment in which the SAS session runs. Buffering data reads can decrease network activities and increase performance. However, 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 modifies the rows after they are read, then you do not see the changes.

A READBUFF= data set option is also available. You can use the data set option to adjust the buffer size during a DATA step when using the Information Maps engine. This changed value is in effect only during the execution of the DATA step. Once the DATA step is completed, the value reverts to the setting specified when the libref was created. For more information, see the "READBUFF= Data Set Option" on page 78.

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: YES

CHAPTER

5 SAS Data Set Options for the Information Maps Engine

Using Data Set Options **75**
AGGREGATE= Data Set Option **75**
EXPCOLUMNLEN= Data Set Option **76**
FILTER= Data Set Option **77**
READBUFF= Data Set Option **78**

Using Data Set Options

Data set options specify actions that apply only to the SAS data set with which they appear. Because the Information Maps engine makes the data from information maps appear as SAS data sets, you can use data set options with information maps that you access through the engine. You specify data set options in parentheses after the information map name in SAS programming statements. To specify several data set options, separate them with spaces. For example:

libref.information-map-name(*option-1=value-1 < ... option-n=value-n>*)

For more information about SAS data set options, see *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.

AGGREGATE= Data Set Option

Specifies whether detailed data or aggregated data should be retrieved

Valid in: DATA Step

Category: Data Set Control

Default: NO

Syntax

AGGREGATE=YES | NO

Details

By default, or when you specify AGGREGATE=NO, aggregate data items in the information map are not accessible through the Information Maps engine. If an information map contains such data items, then a warning is displayed in the SAS log indicating how many data items are not accessible. Specify AGGREGATE=YES to retrieve aggregated data.

When you specify AGGREGATE=YES and use the CONTENTS procedure to view the contents of an information map, a column named Default Aggregation appears in the procedure output showing the default aggregation function that is assigned to the variable. If the original variable was character type, it is changed to numeric type due to applying the aggregation function. For example, a default aggregation function of COUNT on a character variable containing names produces a numeric variable that contains the number of names. A line in the heading of the CONTENTS procedure output shows the number of aggregate variables, if any.

EXPCOLUMNLEN= Data Set Option

Specifies 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=*column-length*

column-length
specifies the length of the SAS column when expressions are used. Valid values are integers from 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 71

FILTER= Data Set Option

Uses the filters that are defined in an information map to specify criteria for subsetting a result set

Valid in: DATA Step
Category: Data Set Control
Restriction: Use only with information maps that contain filters
Restriction: Filters that prompt for values at run time are not supported

Syntax

FILTER=(<NOT> *filter-1* < ... *Boolean-operator* <NOT> *filter-n*>)

Syntax Description

NOT operator
specifies that the inverse of the specified filter criteria is used to subset the data.

For example, if an information map contains a filter named **`Over_30`** that is defined as **`age > 30`**, then specifying the data set option **`FILTER=(NOT Over_30)`** retrieves rows of data in which the AGE data item has a value of 30 or less.

filter
specifies a filter that is applied when data is retrieved from the information map.

You must specify the names for filters that are assigned by SAS for use within the SAS session. The assigned names can differ from the filter names that are defined in the information map in that the assigned filter names conform to the rules for SAS variable names that are specified in the VALIDVARNAME= system option. For more information about the VALIDVARNAME= system option, see the *SAS Language Reference: Dictionary*. You can use the CONTENTS procedure to view the assigned filter names.

Requirement: If you specify more than a single filter, then parentheses are required.

Requirement: If you use the NOT operator with single filter, then parentheses are required.

Boolean-operator
combines the effects of two filters or filter clauses.

AND operator
specifies that data that satisfies the criteria defined in both filters or filter clauses is returned.

OR operator
specifies that data that specifies the criteria defined in either filter or filter clause is returned.

For more information about Boolean operators and expressions, see *SAS Language Reference: Concepts*.

Details

A filter contains criteria for subsetting data. For example, a filter named **`Males`** could be defined in an information map as **`gender="Male"`**.

An information map can contain filters that are not supported by the Information Maps engine. Only filters that are defined using static values (called unprompted filters) can be used in a FILTER= data set option. You can use the CONTENTS procedure to print a list of the filters that are supported by a libref that is created by the Information Maps engine.

Using the FILTER= data set option is similar to using a WHERE clause in a PROC SQL statement. However, filter criteria are applied as data is retrieved from the data source. As a result, a FILTER= option restricts the amount of data that is returned from the data source. In contrast, a WHERE clause is applied as data from the data source is brought into SAS. As a result, a WHERE clause does not restrict the amount of data that is retrieved.

When you specify more than one filter in the FILTER= option, you must use Boolean operators to define the logical relationships between the filters in the filter clause. The rules of precedence for Boolean operators in filter clauses follow the rules set for the SAS WHERE clause. These rules specify that the NOT operator has the highest precedence, followed by the AND and OR operators. You can use parentheses to specify explicit precedence or groupings within the clause. For more information about the rules for the SAS WHERE clause, see "Combining Expressions by Using Logical Operators" in *SAS Language Reference: Concepts*.

Example

In the following example, there are three unprompted filters: **Repeat Buyer**, **Midwest Region**, and **Southwest Region**. The retrieved data is filtered to produce new buyers from either the Midwest or Southwest regions.

```
option validvarname=any;   /* This option is  needed for names with spaces */
libname Orion infomaps ...;
proc print data=Orion.'Star Schema'n
     (filter=( (NOT('Repeat Buyer'n) ) AND
               ( ('Midwest Region'n) OR
                 ('Southwest Region'n) ) ) );
RUN;
```

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 number of rows to hold in memory input into SAS. The value must be positive.

Details

Choosing the optimum value for the READBUFF= option requires a detailed knowledge of the data that is returned from the information map and of the environment in which the SAS session runs. Buffering data reads can decrease network activities and increase performance. However, higher values for the READBUFF= option use more memory. In addition, setting a high value for the READBUFF= option could yield stale data if the data source is updated frequently.

See Also

READBUFF= option in "Other LIBNAME Statement Options for the Information Maps Engine" on page 71

CHAPTER

6

Examples of Using the Information Maps Engine

Example 1: Submitting a LIBNAME Statement Using the Defaults **81**
Example 2: Submitting a LIBNAME Statement Using Connection Options **81**

Example 1: Submitting a LIBNAME Statement Using the Defaults

This example shows a LIBNAME statement that uses the defaults for the Information Maps engine. Because both the METAUSER= and METAPASS= statement options are omitted, the values specified in the METAUSER= and METAPASS= system options are used if they have been set. Otherwise, single sign-on is used. Because the METASERVER= and METAPORT= statement options are omitted, the values specified in the METASERVER= and METAPORT= system options are used. An error occurs if the system options have not been previously specified.

```
libname mylib infomaps
              mappath="/Users/myUserID/My Folder";
```

Output 6.1 Output from the LIBNAME Statement

```
1   libname mylib infomaps
2                mappath="/Users/myUserID/My Folder";
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /Users/myUserID/My Folder
```

Example 2: Submitting a LIBNAME Statement Using Connection 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="myserver.mycompany.com"
                       metaport=8561
                       mappath="/Users/myUserID/My Folder";
```

Output 6.2 Output from the LIBNAME Statement

```
1     libname mylib infomaps metauser='myUserID'
2                            metapass=XXXXXXXXXX
3                            metaserver=myserver.mycompany.com
4                            metaport=8561
5                            mappath="/Users/myUserID/My Folder";
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /Users/myUserID/My Folder
```

CHAPTER 7

Hints and Tips for Using the INFOMAPS Procedure or the Information Maps Engine

Hints and Tips for Using the INFOMAPS Procedure **83**
Hints and Tips for Using the Information Maps Engine **84**
Improving the Performance of the Information Maps Engine **84**
Creating Information Maps That Work Well with the Information Maps Engine **84**
Following SAS Naming Restrictions **84**
Using Calculated Data Items **85**
Working with Natural Language Names in SAS **85**

Hints and Tips for Using the INFOMAPS Procedure

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

- Use the COLUMN=, HIERARCHY=, or MEASURE= option instead of the EXPRESSION= option in the INSERT DATAITEM statement, unless you have a calculated data item. For more information about the INSERT DATAITEM statement and the COLUMN=, HIERARCHY=, and MEASURE= options, see "INSERT DATAITEM Statement" on page 14.
- 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 server. 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 in the **`General`** tab in the properties window for the table.
  - From SAS Data Integration Studio, delete the duplicate tables.
- To prevent the Java Virtual Machine from running out of memory, break the task of creating an information map into smaller steps instead of using a single step. For example, if you are adding a large number of data items, add the first 100 in one PROC INFOMAPS step. Then add the next 100 in a second PROC INFOMAPS step. The number of items that can be added varies with the memory available to the Java Virtual Machine.

Hints and Tips 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.
- Use the DROP= or KEEP= data set options to select only the data items that you need.
- 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 access 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 the following message appears in the SAS log indicating that performance will not be optimal:

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

- 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 garbage collector for the JVM should be configured appropriately.

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

Creating Information Maps That Work Well with the Information Maps Engine

Following SAS Naming Restrictions

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

- Names have a maximum length of 32 bytes in Base SAS software.

  Information map names can be up to 60 bytes long, but you must use names that are no more that 32 bytes long for information maps that you access using the Information Maps engine.

  If a filter or data item in the information map has a name that is more than 32 bytes long, then the name will be reduced to a unique 32-byte name when it is used in a SAS program.
- Descriptions have a maximum length of 256 bytes in Base SAS software. If you create a description in an information map that is more than 256 bytes long, then the description will be truncated when it is used in SAS programs.

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 *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 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. Calculated data items appear only when the AGGREGATE=YES option is used.

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 use natural language names in SAS, you need to do the following:

- Make sure that the PRESERVE_MAP_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_MAP_NAMES option, see "Other LIBNAME Statement Options for the Information Maps Engine" on page 71.
- Specify the VALIDVARNAME=ANY system option to allow names that contain any character, including blank spaces or mixed-case letters. This SAS system option controls the type of SAS variable names that can be created and processed during a SAS session. For more information about the VALIDVARNAME= system option, see *SAS Language Reference: Dictionary*.
- For SAS variable names and filter names, specify natural language names (names that contain blank spaces or symbols) as 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 *SAS Language Reference: Concepts*.

The following example uses the Information Maps engine to make an information map with a natural language name available for use in the PRINT procedure:

```
libname mymap infomaps ... ;

option validvarname=any;
proc print data=mymap."Results  (Yearly)"n (drop="Tax Rate (Yearly)"n);
run;
```

The VALIDVARNAME=ANY option allows the variable name in the DROP= data set option 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 **`Tax Rate (Yearly)`** to remain intact and contain the symbols that are otherwise not allowed in SAS.

Note: The VALIDVARNAME= option applies only to variable names and filter names. **`Results (Yearly)`** is a valid information map name because the PRESERVE_MAP_NAMES= option in the LIBNAME statement for the Information Maps engine defaults to YES. △

CHAPTER

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

About This Example **87**
Step 1: Create a Library Definition in the SAS Metadata Server **87**
Step 2: Set the Metadata System Options and a Macro Variable **88**
Step 3: Register Data Using the METALIB Procedure **88**
Step 4: Create an Information Map Using the INFOMAPS Procedure **90**
Step 5: Retrieve the Data Associated with the Information Map Using the Information Maps Engine **96**
Step 6: View the Data Items and Filters Using the CONTENTS Procedure **96**
Step 7: Print the Data from the Information Map **98**
Step 8: Analyze the Data in SAS and Produce an ODS Report **100**

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 other SAS software products 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: Create a Library Definition in the SAS Metadata Server

Before you can use a data source in an information map, you must define the data source in the SAS Metadata Server. This example uses sample data sets that are provided with SAS. The sample data sets are typically located in the **`!sasroot\SASFoundation\9.2\core\sample`** directory (where **`!sasroot`** indicates the directory in which SAS is installed at your location). This example uses the EMPINFO, JOBCODES, and SALARY data sets in that directory.

Use SAS Management Console to define a library named **`HR`** that points to the directory that contains the sample data sets. For more information about creating

libraries using SAS Management Console, see the online Help for SAS Management Console.

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 2: Set the Metadata System Options and a Macro Variable

This example uses metadata system options and a macro variable to set 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 system options for connecting to the metadata server. */
options metauser="myUserID"
        metapass="myPassword"
        metaserver="myserver.mycompany.com"
        metaport=8561;
/* Assign a macro variable to store the path for the folder */
/* that contains information maps (to avoid having to set   */
/* the path multiple times).                                */
%LET infomap_path=/Shared Data;
```

Output 8.1 Log for Setting the Metadata System Options and the Macro Variable

```
1    /* Set system options for connecting to the metadata server. */
2    options metauser="myUserID"
3            metapass=XXXXXXXXXX
4            metaserver="myserver.mycompany.com"
5            metaport=8561;
6    /* Assign a macro variable to store the path for the folder */
7    /* that contains information maps (to avoid having to set   */
8    /* the path multiple times).                                */
9    %LET infomap_path=/Shared Data;
```

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

Step 3: Register Data Using the METALIB Procedure

To register the tables in a SAS Metadata Server, you need to use the METALIB procedure. The METALIB procedure synchronizes table definitions in a metadata server with current information from the physical library data source. For more information about the METALIB procedure, see the *SAS Language Interfaces to Metadata*.

The following code registers the tables using the METALIB procedure:

```
/* Use the library object defined in the SAS Metadata Server     */
/* to obtain all accessible table metadata from the data source  */
/* to create table metadata in the metadata server.              */
proc metalib;
     omr (library="Demo_V92_Doc");
     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 from what is shown. △

Output 8.2 Log for the METALIB Procedure

```
11   /* Use the library object defined in the SAS Metadata Server     */
12   /* to obtain all accessible table metadata from the data source  */
13   /* to create table metadata in the metadata server.              */
14   proc metalib;
15        omr (library="HR");
16        select("empinfo" "jobcodes" "salary");
17        /* Create a summary report of the metadata changes. */
18        report;
19   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           2.85 seconds
      cpu time            0.49 seconds
```

Output 8.3 Output from the METALIB Procedure

```
                                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
                              Tables not processed            0


    ------------------------------------------------------------------------------------
                                        Tables Added
    ------------------------------------------------------------------------------------


    Metadata Name                     Metadata ID          SAS Name

    EMPINFO                           A5U6VCF4.B80003T3    EMPINFO
    JOBCODES                          A5U6VCF4.B80003T4    JOBCODES
    SALARY                            A5U6VCF4.B80003T5    SALARY
```

Step 4: Create an Information Map Using the INFOMAPS Procedure

Once the tables are registered in the metadata server, 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";
/* Open a new information map. */
new infomap "Employee Info"
    auto_replace=yes;

/* Insert a data source and three data items using the COLUMNS= option. */
insert datasource sasserver="SASMain"
       table="HR".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="HR".jobcodes
       columns=( "TITLE" )
       id="Jobcodes";

/* Change the data item to a measure so that you can use it in computations */
/* and analytical expressions. Set the default aggregation to Count.        */
update dataitem "Title" aggregation=COUNT classification=MEASURE;

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

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

insert relationship
```

```
       left_table="Empinfo"
       right_table="Salary"
       join=inner
       condition="(<<Empinfo.IDNUM>>=<<Salary.IDNUM>>)";


/* Insert a folder and additional business items. */
insert folder "Salary Info";

insert dataitem column="Salary".salary
       name="Annual Salary" folder="Salary Info";

/* Insert a data item that contains an expression */
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 the key properties of business data in the current information map. */
list;

/* Save the information map. */
save;

/* End the INFOMAPS procedure. */
quit;
```

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

Output 8.4 Log for the INFOMAPS Procedure

```
11   /* Create a new information map using the INFOMAPS procedure. */
12   proc infomaps mappath="&infomap_path";
13   /* Open a new information map. */
14   new infomap "Employee Info"
15       auto_replace=yes;
16
17   /* Insert a data source and three data items using the COLUMNS= option. */
18   insert datasource sasserver="SASMain"
19          table="HR".empinfo
20          columns=("JobCode" "LOCATION" "DIVISION")
21          id="Empinfo";
22
23   /* Insert a data item based on a physical column. Because the ID= option */
24   /* is not specified, a note with its ID value will print in the SAS log. */
25   insert dataitem column="Empinfo".idnum classification=category;
NOTE: A data item was inserted for the physical column Empinfo.IDNUM. The data item's ID is
      "Identification Number".
26
27   /* Insert a data item with an expression. */
28   insert dataitem expression="SUBSTRN(<<root.Jobcode>>, 1, 3)"
29          type=character
30          name="Department Code"
31          id="Dept_code";
32
33   /* Insert a second data source, plus a data item into the */
34   /* current information map. */
35   insert datasource sasserver="SASMain"
36          table="HR".jobcodes
37          columns=( "TITLE" )
38          id="Jobcodes";
39
40   /* Change the data item to a measure so that you can use it in computations */
41   /* and analytical expressions. Set the default aggregation to Count.        */
42   update dataitem "Title" aggregation=COUNT classification=MEASURE;
43
44   /* Insert a third data source into the current information map. */
45   insert datasource sasserver="SASMain"
46          table="HR".salary
47          id="Salary";
48
49   /* Add joins between the tables. */
50   insert relationship
51          left_table="Empinfo"
52          right_table="Jobcodes"
53          join=inner
54          condition="(<<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>>)";
NOTE: A relationship between the tables "Empinfo" and "Jobcodes" has been inserted. The
      relationship's ID is "JOIN_10".
55
56   insert relationship
57          left_table="Empinfo"
58          right_table="Salary"
59          join=inner
60          condition="(<<Empinfo.IDNUM>>=<<Salary.IDNUM>>)";
NOTE: A relationship between the tables "Empinfo" and "Salary" has been inserted. The
      relationship's ID is "JOIN_11".
61
62
63   /* Insert a folder and additional business items. */
64   insert folder "Salary Info";
65
66   insert dataitem column="Salary".salary
67          name="Annual Salary" folder="Salary Info";
68
69   /* Insert a data item that contains an expression */
70   insert dataitem expression="<<Salary.SALARY>>/12" type=numeric
71          name="Monthly Salary" folder="Salary Info";
72
73   insert dataitem column="Salary".enddate folder="Salary Info";
NOTE: A data item was inserted for the physical column Salary.ENDDATE. The data item's ID is
      "Enddate".
74
```

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

Total datasources: 3

    Data source: HR.EMPINFO
    ID: Empinfo
    Name: EMPINFO
    Description:

    Data source: HR.JOBCODES
    ID: Jobcodes
    Name: JOBCODES
    Description:

    Data source: HR.SALARY
    ID: Salary
    Name: SALARY
    Description:

Total data items: 9

    Data item name: Annual Salary
    ID: Annual Salary
    Folder: /Salary Info
    Description: Physical column SALARY
    Expression: <<Salary.Salary>>
    Expression type: NUMERIC
    Classification: MEASURE
    Format: DOLLAR12.
    Default aggregation: Sum

    Data item name: Department Code
    ID: Dept_code
    Folder: /
    Description:
    Expression: SUBSTRN(<<root.Jobcode>>, 1, 3)
    Expression type: CHARACTER
    Classification: CATEGORY
    Format:
```

```
Data item name: Division
ID: Division
Folder: /
Description: Physical column DIVISION
Expression: <<Empinfo.DIVISION>>
Expression type: CHARACTER
Classification: CATEGORY
Format:

Data item name: Enddate
ID: Enddate
Folder: /Salary Info
Description: Physical column ENDDATE
Expression: <<Salary.ENDDATE>>
Expression type: DATE
Classification: CATEGORY
Format: DATE9.

Data item name: Identification Number
ID: Identification Number
Folder: /
Description: Physical column IDNUM
Expression: <<Empinfo.Identification Number>>
Expression type: NUMERIC
Classification: CATEGORY
Format: SSN11.

Data item name: Jobcode
ID: Jobcode
Folder: /
Description: Physical column JOBCODE
Expression: <<Empinfo.JOBCODE>>
Expression type: CHARACTER
Classification: CATEGORY
Format:

Data item name: Location
ID: Location
Folder: /
Description: Physical column LOCATION
Expression: <<Empinfo.LOCATION>>
Expression type: CHARACTER
Classification: CATEGORY
Format:
```

```
    Data item name: Monthly Salary
    ID: Monthly Salary
    Folder: /Salary Info
    Description:
    Expression: <<Salary.Salary>>/12
    Expression type: NUMERIC
    Classification: CATEGORY
    Format: DOLLAR12.

    Data item name: Title
    ID: Title
    Folder: /
    Description: Physical column TITLE
    Expression: <<Jobcodes.TITLE>>
    Expression type: CHARACTER
    Classification: MEASURE
    Format: BEST12.
    Default aggregation: Count

Total filters: 4

    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

Total Relationships: 2

    Relationship ID: JOIN_10
    Left table: HR.EMPINFO
    Right table: HR.JOBCODES
    Cardinality: UNKNOWN
    Join type: INNER
    Join expression: (<<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>>)

    Relationship ID: JOIN_11
    Left table: HR.EMPINFO
    Right table: HR.SALARY
    Cardinality: UNKNOWN
    Join type: INNER
    Join expression: (<<Empinfo.Identification Number>>=<<Salary.Identification Number>>)

93
94   /* Save the information map. */
95   save;
NOTE: The information map "Employee Info" has been saved in the folder "/Shared Data".
96
97   /* End the INFOMAPS procedure. */
98   quit;

NOTE: PROCEDURE INFOMAPS used (Total process time):
      real time           30.17 seconds
      cpu time            0.09 second
```

Step 5: 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.

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

```
/* Use the Information Maps engine to define a libref to retrieve */
/* data from the information maps. */
libname HR_Data 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 8.5 Log for the Information Maps Engine LIBNAME Statement

```
100  /* Use the Information Maps engine to define a libref to retrieve */
101  /* data from the information maps. */
102  libname HR_Data infomaps mappath="&infomap_path";
NOTE: Libref HR_DATA was successfully assigned as follows:
      Engine:        INFOMAPS
      Physical Name: /Shared Data
103  /* Allow mixed-case letters and blank spaces in information map names. */
104  option validvarname=any;
```

Step 6: 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 the default set of information about the data items and filters:

```
/* View the data items, including any filters, in the information map. */
proc contents data=HR_Data."Employee Info"n;
run;
```

Output 8.6 Log for the CONTENTS Procedure

```
106  /* View the data items, including any filters, in the information map. */
107  proc contents data=HR_Data."Employee Info"n;
108  run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           6.01 seconds
      cpu time            0.18 seconds
```

Output 8.7 Output from the CONTENTS Procedure

```
                              The CONTENTS Procedure

      Data Set Name        HR_DATA.'Employee Info'n    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.    Physical column SALARY
5    Dept_code               Char     32
3    Division                Char     40                 Physical column DIVISION
9    Enddate                 Num       8    DATE9.       Physical column ENDDATE
4    Identification Number   Num       8    SSN11.       Physical column IDNUM
1    Jobcode                 Char      8                 Physical column JOBCODE
2    Location                Char      8                 Physical column LOCATION
8    Monthly Salary          Num       8    DOLLAR12.
6    Title                   Char     20    $F20.        Physical column TITLE


                                  Information Maps

  FilterName                 FilterType  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
```

The following code uses the AGGREGATE= data set option in conjunction with the CONTENTS procedure to display additional information about the aggregations that are applied to data items that are measures:

```
/* Turn on aggregation and view the contents of the information map. */
proc contents data=HR_Data."Employee Info"n(aggregate=yes);
run;
```

Output 8.8 Log for the CONTENTS Procedure

```
110  /* Turn on aggregation and view the contents of the information map. */
111  proc contents data=HR_Data."Employee Info"n(aggregate=yes);
112  run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           4.14 seconds
      cpu time            0.06 seconds
```

Output 8.9 Output from the CONTENTS Procedure

```
                                The CONTENTS Procedure

          Data Set Name        HR_DATA.'Employee Info'n    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                     Aggregate Variables   2
          Encoding             Default


                        Alphabetic List of Variables and Attributes

                                                    Default
#    Variable                Type    Len    Format      Aggregation    Label

7    Annual Salary           Num       8    DOLLAR12.   SUM            Physical column SALARY
5    Dept_code               Char     32
3    Division                Char     40                              Physical column DIVISION
9    Enddate                 Num       8    DATE9.                     Physical column ENDDATE
4    Identification Number   Num       8    SSN11.                     Physical column IDNUM
1    Jobcode                 Char      8                               Physical column JOBCODE
2    Location                Char      8                               Physical column LOCATION
8    Monthly Salary          Num       8    DOLLAR12.
6    Title                   Num       8    BEST12.     COUNT          Physical column TITLE


                                  Information Maps

        FilterName                FilterType  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 7: 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 five observations of detailed data from the data that */
/* the information map references.                             */
proc print data=HR_Data."Employee Info"n (obs=5);
run;

title1 "Total salary for each division except Education and Publications";
title2 "and Host Systems Development";
proc print data=HR_Data."Employee Info"n
   (keep="Annual Salary"n Division
    aggregate=yes
    filter=(NOT("Education and Publications"n
```

```
                                    OR "Host Systems Development"n)));
            run;
```

Output 8.10 Log for the PRINT Procedure

```
114  /* Print five observations of detailed data from the data that */
115  /* the information map references.                            */
116  proc print data=HR_Data."Employee Info"n (obs=5);
117  run;

NOTE: There were 5 observations read from the data set HR_DATA.'Employee Info'n.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           5.57 seconds
      cpu time            0.14 seconds

118
119  title1 "Total salary for each division except Education and Publications";
120  title2 "and Host Systems Development";
121  proc print data=HR_Data."Employee Info"n
122     (keep="Annual Salary"n Division
123      aggregate=yes
124      filter=(NOT("Education and Publications"n
125                  OR "Host Systems Development"n)));
126  run;

NOTE: There were 14 observations read from the data set HR_DATA.'Employee Info'n.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           4.78 seconds
      cpu time            0.09 seconds
```

Output 8.11 Output from the PRINT Procedure

The SAS System

Obs	Jobcode	Location	Division	Identification Number	Dept_ 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

Obs	Title	Annual Salary	Monthly 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	.

Total salary for each division except Education and Publications and Host Systems Development

Obs	Division	Annual Salary
1	CALIFORNIA REGIONAL	$96,500
2	CONTRACTS	$511,000
3	EXECUTIVE	$973,000
4	FACILITIES	$596,500
5	FINANCE	$326,000
6	HUMAN RESOURCES	$824,500
7	INFORMATION SYSTEMS	$919,500
8	INTERNAL DATA BASE	$178,000
9	QUALITY ASSURANCE	$898,000
10	SALES & MARKETING	$1,254,500
11	SOFTWARE DEVELOPMENT	$2,348,000
12	TECHNICAL SUPPORT	$688,000
13	TEXAS REGIONAL	$918,000
14	VIDEO	$238,400

Step 8: 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 HR_Data."Employee Info"n(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 8.12 Log for the DATA Step and the MEANS Procedure

```
127  data work.HRinfo;
128  set HR_Data."Employee Info"n(filter="Host Systems Development"n);
129  keep jobcode "Annual Salary"n;
130  run;

NOTE: There were 21 observations read from the data set HR_DATA.'Employee Info'n.
NOTE: The data set WORK.HRINFO has 21 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           4.73 seconds
      cpu time            0.10 seconds

131
132  /* Produce an ODS report. */
133  ods html body="example-body.htm";
NOTE: Writing HTML Body file: example-body.htm
134  /* Analyze the annual salary distribution data. */
135  proc means data=work.HRinfo maxdec=0;
136             var "Annual Salary"n;
137             class jobcode;
138             title "Annual Salary by Job Code";
139  run;

NOTE: There were 21 observations read from the data set WORK.HRINFO.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.39 seconds
      cpu time            0.10 seconds

140  ods html close;
```

Output 8.13 Output from the MEANS Procedure

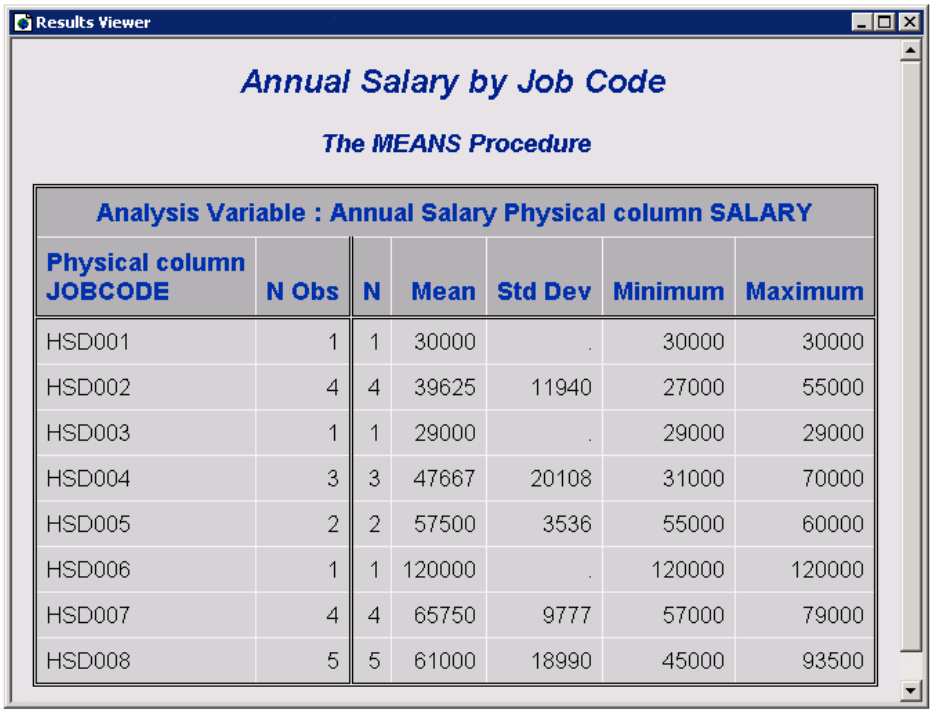
The MEANS Procedure

Analysis Variable : Annual Salary Physical column SALARY

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

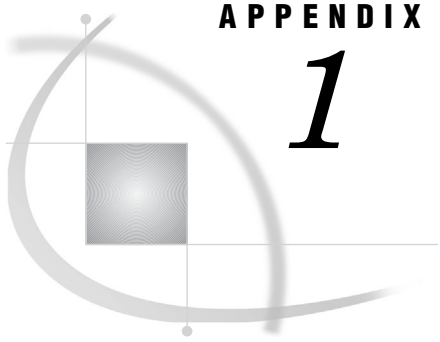
The report that is produced by ODS should look similar to the following:

Display 8.1 Report Displayed in the Results Viewer



Analysis Variable : Annual Salary Physical column SALARY

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

SQL DICTIONARY Tables for the Information Maps Engine

Using SQL DICTIONARY Tables **103**
DICTIONARY.INFOMAPS Table **103**
DICTIONARY.DATAITEMS Table **104**
DICTIONARY.FILTERS Table **105**

Using SQL DICTIONARY Tables

An SQL DICTIONARY table is a read-only SAS view that contains information about a SAS library or SAS data set. The Information Maps engine makes an information map appear like a SAS data set within a SAS library. For the engine, an information map contains one or more data items, as well as zero or more filters. The following SQL DICTIONARY tables are available for use in conjunction with the Information Maps engine:

DICTIONARY.INFOMAPS
: contains metadata about the information maps that are available in the current SAS session

DICTIONARY.DATAITEMS
: contains metadata about the data items that are defined in the available information maps

DICTIONARY.FILTERS
: contains metadata about the filters that are defined in the available information maps

You can use the SQL procedure in Base SAS to query these tables and retrieve information about the information maps.

DICTIONARY.INFOMAPS Table

The SQL DICTIONARY.INFOMAPS table contains a row for each information map that is available through the Information Maps engine. The table contains the following variables:

LIBNAME
: Information Maps engine libref for the information map

MEMNAME
: SAS name for the information map

MAPNAME
Information map name

PATH
Location of the information map within the metadata server

DESCRIPTION
Description of the information map

The following example shows how you can query the DICTIONARY.INFOMAPS table to retrieve information about the available information maps:

```
libname mymaps infomaps mappath="/Users/myUserID/My Folder";

proc sql;
   select i.mapname, i.path
      from DICTIONARY.INFOMAPS as i;
```

Output A1.1 Output from DICTIONARY.INFOMAPS Table Query

```
Information Map Name                          Information Map Path
--------------------------------------------------------------------------------
Employee Statistics Sample                    /Users/myUserID/My Folder
```

DICTIONARY.DATAITEMS Table

The SQL DICTIONARY.DATAITEMS table contains a row for each data item in all of the information maps that are available through the Information Maps engine. The table contains the following variables:

LIBNAME
Information Maps engine libref for the information maps that contains the data item

MEMNAME
SAS name for the information map that contains the data item

NAME
SAS name for the data item

DATAITEMNAME
Data item name

ID
Data item ID

PATH
Location of the data item within the metadata server

CLASS
Classification of the data item

AGGREGATION
Default aggregate function for the data item

ISCALC
Flag to indicate whether the data item contains a calculated expression (YES or NO)

ISUSABLE
Flag to indicate whether the underlying data for data item is available (YES or NO)

DESCRIPTION
Description of the data item

The following example shows how you can query the DICTIONARY.DATAITEMS table to retrieve information about the available data items:

```
libname mymaps infomaps mappath="/Users/myUserID/My Folder";

proc sql;
   select d.memname, d.dataitemname, d.id, d.class
      from DICTIONARY.DATAITEMS as d;
```

Output A1.2 Output from DICTIONARY.DATAITEMS Table Query

Member Name	Data Item Name	Data Item ID	Classification
Employee Statistics Sample	Name	Name	CATEGORY
Employee Statistics Sample	Identification Number	Identification Number	CATEGORY
Employee Statistics Sample	Job Code	Jobcode	CATEGORY
Employee Statistics Sample	Department	Deptcode	CATEGORY
Employee Statistics Sample	Location	Location	CATEGORY
Employee Statistics Sample	Average Salary	Salary_2	MEASURE
Employee Statistics Sample	Minimum Salary	Salary2	MEASURE
Employee Statistics Sample	Maximum Salary	Salary3	MEASURE
Employee Statistics Sample	Sum of Salaries	Salary4	MEASURE
Employee Statistics Sample	Hire Date	Hire Date	CATEGORY
Employee Statistics Sample	Number of Years Employed	Number of Years Employed	CATEGORY

DICTIONARY.FILTERS Table

The SQL DICTIONARY.FILTERS table contains a row for each filter in all of the information maps that are available through the Information Maps engine. The table contains the following variables:

LIBNAME
Information Maps engine libref for the information map that contains the filter

MEMNAME
SAS name for the information map that contains the filter

NAME
SAS name for the filter

FILTERNAME
Filter name

ID
Filter ID

PATH
Location of the filter within the metadata server

DESCRIPTION
Description of the filter

The following example shows how you can query the DICTIONARY.FILTERS table to retrieve information about the available filters:

```
libname mymaps infomaps mappath="/Users/myUserID/My Folder";

proc sql;
   select f.memname, f.filtername, f.id, f.description
      from DICTIONARY.FILTERS as f;
```

Output A1.3 Output from DICTIONARY.FILTERS Table Query

Member Name	Filter Name	Filter ID	Filter Description
Employee Statistics Sample	Cary Employees	Cary Employees	Employees who work in Cary, North Carolina.
Employee Statistics Sample	Which department?	Which department?	Filters based on selected departments.

APPENDIX

2

SAS Tracing and the Information Maps Engine

Tracing Diagnostic Messages from the Information Maps Engine **107**
Example **107**

Tracing Diagnostic Messages from the Information Maps Engine

You can use the SASTRACE system option to trace diagnostic messages issued by the Information Maps engine. You can specify the following parameters:

SASTRACE= ',,,<s | sa>'

',,,s'
specifies that a summary of timing information is sent to the log.

',,,sa'
specifies that detailed timing messages are sent to the log along with a summary.

You can use the SASTRACELOC system option to specify the destination of the diagnostic messages.

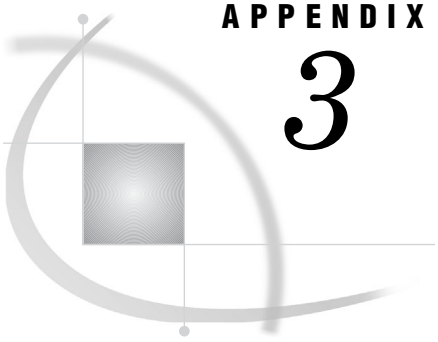
SASTRACELOC= stdout |SASLOG | FILE *'pathname'*

You can use the NOSTSUFFIX system option to simplify diagnostic messages by suppressing some nonessential output.

Example

The following statement causes both detailed and summary timing information to be written to the SAS log:

```
options sastrace=',,,sa' sastraceloc=saslog nostsuffix;
```



APPENDIX

3

Recommended Reading

Recommended Reading

The recommended reading list for this title is:

- *The Little SAS Book: A Primer*
- *Step-by-Step Programming with Base SAS Software*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *Base SAS Procedures Guide*
- *SAS Language Interfaces to Metadata*
- SAS Companion that is specific to your operating environment
- Base SAS Focus Area at **`http://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: **`http://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 classification of data items. Category data items are used to group measure data items, using an applied aggregate function. For example, a category data item that contains the names of countries could be used to group a measure data item that contains population values.

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 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 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 data in a form that is relevant and meaningful to a business user.

data set
See SAS data set.

data source
a logical representation of a table or cube that an information map retrieves data from.

DATA step
in a SAS program, a group of statements that begins with a DATA statement and that 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.

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. Short form: XML.

filter
in an information map, criteria (rules) 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 and 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 provides a user-friendly view of a data source. When you use an information map to query data for business needs, you do not have to understand the structure of the underlying data source or know how to program in a query language.

inner join
a join between two tables that returns all of the rows in one table that have one or more matching rows in the other table. See also join.

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
a short name for the full physical name of a SAS library. In the context of the SAS Metadata Server, a libref is associated with a SAS library when the library is defined in the metadata server.

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

MDX language
See multidimensional expressions language.

measure
(1) a classification of data items. The values of measure data items are aggregated (unless otherwise specified) and can be used in computations or analytical expressions. For example, a measure data item could contain age values that are grouped by gender and then averaged. (2) a member of a Measure dimension.

member
(1) a SAS file in a SAS library. (2) 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.

multidimensional expressions language
a standardized, high-level language that is used to query multidimensional data sources. The MDX language is the multidimensional equivalent of SQL (Structured Query Language). Short form: MDX language.

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
See online analytical processing.

online analytical processing
a software technology that enables users to dynamically analyze data that is stored in cubes. Short form: OLAP.

outer join
a join between two tables that returns all of the rows in one table, as well as part or all of the rows in the other table. A left or right outer join returns all of the rows in one table (the table on the left or right side of the SQL statement, respectively), as well as the matching rows in the other table. A full outer join returns all of the rows in both of the tables. See also join.

physical data
data values that are stored on any type 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 server. For example, if you register a table, you save metadata about that table to a metadata server.

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 and 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 that are stored 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 Server to provide centralized definitions for SAS applications.

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 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
See Structured Query Language.

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

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. Short form: SQL.

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
See Extensible Markup Language.

Index

A

aggregate functions
  example 60
  removing from list 19, 20, 45
  specifying default 17, 44
  table of 18
AGGREGATION= argument
  INSERT DATAITEM statement (INFOMAPS) 17
  UPDATE DATAITEM statement (INFOMAPS) 44
AGGREGATIONS_DROP_LIST= argument
  INSERT DATAITEM statement (INFOMAPS) 19, 20
  UPDATE DATAITEM statement (INFOMAPS) 45
ALL argument
  INSERT DATASOURCE statement (INFOMAPS) 25
  LIST statement (INFOMAPS) 34
analyzing data 100

B

best practices
  INFOMAPS procedure 83
  Information Maps engine 84
buffering data reads 72, 79
business data definitions 33
BY-group processing 67

C

calculated data items 85
capitalizing data item names 40, 53
CLASSIFICATION= argument
  INSERT DATAITEM statement (INFOMAPS) 20
  UPDATE DATAITEM statement (INFOMAPS) 45
COLUMN= argument
  INSERT DATAITEM statement (INFOMAPS) 14, 83
column labels
  as data item names 41, 54
columns
  inserting data items 14
  inserting data items for each column 25
  length of 72, 76
  specifying physical column names 26
COLUMNS= argument
  INSERT DATASOURCE statement (INFOMAPS) 26
CONDITION= argument
  INSERT FILTER statement (INFOMAPS) 27
  UPDATE FILTER statement (INFOMAPS) 49
CONTENTS procedure
  viewing aggregated data items 97
  viewing data items and filters 64, 96
CREATE_TARGET_FOLDER= argument
  NEW INFOMAP statement (INFOMAPS) 40
  UPDATE INFOMAP statement (INFOMAPS) 52
CUBE= argument
  INSERT DATASOURCE statement (INFOMAPS) 24
cubes
  as data source 24
  inserting data from 24

D

data analysis 100
data items 1, 6
  aggregate functions 17, 19, 20, 44, 45,
  calculated 85
  capitalizing names of 40, 53
  column labels as names 41, 54
  data type of expressions 23, 47
  descriptions of 21, 46
  folders for 21, 58
  format of 21, 47
  ID specification 22
  inserting 14
  inserting for each column or hierarchy 25
  listing properties of 33
  naming 22, 41, 47, 54
  underscores in names 41, 54
  usage type of 20, 45
  viewing with CONTENTS procedure 64, 96
data reads
  buffering 72, 79
data set options
  for Information Maps engine 75
data sources
  cubes as 24
  ID specification 26
  inserting 24
  listing properties of 33
  relational tables as 25
data types
  of expressions 23, 47
DATAITEMS argument
  LIST statement (INFOMAPS) 33
DATASETS procedure
  listing available information maps 64

DATASOURCES argument
  LIST statement (INFOMAPS) 33
DELETE INFOMAP statement
  INFOMAPS procedure 11
DESCRIPTION= argument
  INSERT DATAITEM statement (INFOMAPS) 21
  INSERT FILTER statement (INFOMAPS) 28
  INSERT FOLDER statement (INFOMAPS) 30
  UPDATE DATAITEM statement (INFOMAPS) 46
  UPDATE FILTER statement (INFOMAPS) 50
DOMAIN= argument
  PROC INFOMAPS statement 9
DOMAIN= option
  LIBNAME statement 70

E

engine
  See Information Maps engine
examples 87
  aggregate functions 60
  analyzing data 100
  creating information maps with INFOMAPS procedure 90
  INFOMAPS procedure 57
  LIBNAME statement 63, 81
  ODS reports 100
  printing information map data 98
  registering data with METALIB procedure 88
  retrieving data 96
  setting macro variable 88
  setting metadata system options 88
  viewing aggregated data items 97
  viewing data items and filters 96
EXPCOLUMNLEN= argument
  LIBNAME statement 14, 46
EXPCOLUMNLEN= data set option 76
EXPCOLUMNLEN= option
  LIBNAME statement 72
EXPORT statement
  INFOMAPS procedure 12
exporting information maps 12
EXPRESSION= argument
  INSERT DATAITEM statement (INFOMAPS) 14
expressions
  column length and 72, 76
  data type of 23, 47
  inserting data items 14

F

FILE= argument
  EXPORT statement (INFOMAPS) 12
  IMPORT statement (INFOMAPS) 13
FILTER= data set option 77
filters 1, 6
  as WHERE clause 27, 49
  conditional expression 27, 49
  description of 28, 50
  folder for inserting 28
  inserting 27
  listing properties of 33
  restricting returned data 64
  updating 49
  viewing with CONTENTS procedure 64, 96
FILTERS argument
  LIST statement (INFOMAPS) 33
FOLDER= argument
  INSERT DATAITEM statement (INFOMAPS) 21
  INSERT FILTER statement (INFOMAPS) 28
folders 1
  creating automatically 40, 52, 58
  description of 30
  inserting 30, 58
  inserting data items 21, 58
  inserting filters into 28
  parent folders 30
  updating 51
FORMAT= argument
  INSERT DATAITEM statement (INFOMAPS) 21
  UPDATE DATAITEM statement (INFOMAPS) 47
formats
  data item format 21, 47

H

hierarchies
  inserting data items for each hierarchy 25

I

ID= argument
  INSERT DATAITEM statement (INFOMAPS) 22
  INSERT DATASOURCE statement (INFOMAPS) 26
IMPORT statement
  INFOMAPS procedure 13
importing information maps 13
INFOMAP argument
  EXPORT statement (INFOMAPS) 12
  SAVE statement (INFOMAPS) 42
INFOMAPS argument
  LIBNAME statement 69
INFOMAPS procedure 5
  aggregating data items 60
  best practices 83
  creating information maps 39, 57, 90
  DELETE INFOMAP statement 11
  examples 57
  EXPORT statement 12
  folders in information maps 58
  IMPORT statement 13
  INSERT DATAITEM statement 14
  INSERT DATASOURCE statement 24
  INSERT FILTER statement 27
  INSERT FOLDER statement 30
  INSERT RELATIONSHIP statement 31
  LIST statement 33
  NEW INFOMAP statement 39
  PROC INFOMAPS statement 9
  SAVE statement 42
  syntax 9
  UPDATE FILTER statement 49
  UPDATE FOLDER statement 51
  UPDATE INFOMAP statement 52
  UPDATE RELATIONSHIP statement 55
  updating information maps 52
information maps 1
  benefits of 2
  business data definitions 33
  calculated data items 85
  creating 39, 57, 90

deleting from metadata server 11
exporting in XML 12
folders in 58
importing from XML file 13
inserting data items 14
inserting data sources 24
inserting filters 27
inserting folders 30
inserting joins 31
list of available maps 64
names with special characters 72, 85
naming 39, 52
naming restrictions and 84
opening 39
printing data 64, 98
restricting returned data 64
SAS Information Map Studio and 6
saving 42
updating 52
updating filters 49
updating folders 51
updating joins 55
Information Maps engine 63
advantages of 67
best practices 84
data set options 75
how it works 63
memory usage 84
nickname 69, 96
performance improvement 84
requirements for 67
retrieving data 96
submitting LIBNAME statement using connection options 81
submitting LIBNAME statement using defaults 81
what is supported 67
INIT_CAP= argument
NEW INFOMAP statement (INFOMAPS) 40
UPDATE INFOMAP statement (INFOMAPS) 53
input
number of rows to hold in memory 72, 79
INSERT DATAITEM statement
INFOMAPS procedure 14
INSERT DATASOURCE statement
INFOMAPS procedure 24
INSERT FILTER statement
INFOMAPS procedure 27
INSERT FOLDER statement
INFOMAPS procedure 30
INSERT RELATIONSHIP statement
INFOMAPS procedure 31
IP address
of metadata server host 10, 71

J

joins
inserting 31
updating 55

L

LIBNAME statement, Information Maps engine 69
connection options for metadata server 70
examples 63, 81
global options 71
submitting, using connection options 81
submitting, using defaults 81
syntax 69
librefs 63, 69
LIST statement
INFOMAPS procedure 33

M

macro variable 88
map folders
See folders
MAPPATH= argument 9
DELETE INFOMAP statement (INFOMAPS) 11
EXPORT statement (INFOMAPS) 12
NEW INFOMAP statement (INFOMAPS) 40
PROC INFOMAPS statement 9
SAVE statement (INFOMAPS) 42
UPDATE INFOMAP statement (INFOMAPS) 54
MAPPATH= option
LIBNAME statement 69
MEANS procedure 100
member-name normalization 72
memory
for Information Maps engine 84
number of rows to hold in 72, 79
METACREDENTIALS= argument
LIBNAME statement 70
PROC INFOMAPS statement 10
metadata 1
metadata server
connecting to 9, 70
deleting information maps from 11
IP address of host 10, 71
LIBNAME statement connection options 70
passwords for 10, 70
path 9
TCP port 10
user ID for connecting 10, 71
metadata system options 88
METALIB procedure 88
METAPASS= argument
PROC INFOMAPS statement 10
METAPASS= option
LIBNAME statement 70
METAPORT= argument
PROC INFOMAPS statement 10
METAPORT= option
LIBNAME statement 70
METASERVER= argument
PROC INFOMAPS statement 10
METASERVER= option
LIBNAME statement 71
METAUSER= argument
PROC INFOMAPS statement 10
METAUSER= option
LIBNAME statement 71

N

NAME= argument
INSERT DATAITEM statement (INFOMAPS) 22
UPDATE DATAITEM statement (INFOMAPS) 47
name literals 72
names
data items 22, 40, 41, 47, 53,

information maps 39, 52, 72, 85
natural language names 85
nickname for Information Maps engine 69
physical column names 26
restrictions on 84
special characters in information map names 72, 85
tables 83
natural language names 85
NEW INFOMAP statement
INFOMAPS procedure 39
nickname for Information Maps engine 69, 96

O

ODS reports 100
OLAP cubes
as data source 24

P

PARENT= argument
INSERT FOLDER statement (INFOMAPS) 30
parent folders 30
passwords
metadata server 10, 70
path
See also MAPPATH= argument
metadata server 9
performance
Information Maps engine 84
PRESERVE_MAP_NAMES= option
LIBNAME statement 72, 85
PRINT procedure
printing information map data 64, 98
PROC INFOMAPS statement 9
properties
listing 33

Q

queries
for subsetting result sets 77

R

READBUFF= data set option 79
READBUFF= option
LIBNAME statement 72
registering data 88
relational databases
filter as WHERE clause 27, 49
relational tables
as data source 25
inserting joins 31
updating joins 55
REPLACE_UNDERSCORES= argument
NEW INFOMAP statement (INFOMAPS) 41
UPDATE INFOMAP statement (INFOMAPS) 54
result sets
subsetting 27, 49, 77
retrieving data 96
rows
number to hold in memory 72, 79

S

SAS Information Map Studio 6, 63
SAS Information Maps
See information maps
SAS server
inserting data sources and 25
SASSERVER= argument
INSERT DATASOURCE statement (INFOMAPS) 25
SAVE statement
INFOMAPS procedure 42
saving information maps 42
special characters
information map names 72, 85
spool file 73
SPOOL= option
LIBNAME statement 73
SSPI= argument
LIBNAME statememt 71
subsetting result sets 27, 49, 77
system options
setting metadata system options 88
VALIDVARNAME= 85

T

TABLE= argument
INSERT DATASOURCE statement (INFOMAPS) 25
tables
inserting data from 24
naming 83
TCP port 10, 70
TYPE= argument
INSERT DATAITEM statement (INFOMAPS) 23
UPDATE DATAITEM statement (INFOMAPS) 47

U

underscores
in data item names 41, 54
UPDATE FILTER statement
INFOMAPS procedure 49
UPDATE FOLDER statement
INFOMAPS procedure 51
UPDATE INFOMAP statement
INFOMAPS procedure 52
UPDATE RELATIONSHIP statement
INFOMAPS procedure 55
USE_LABELS= argument
NEW INFOMAP statement (INFOMAPS) 41
UPDATE INFOMAP statement (INFOMAPS) 54
user ID
connecting to metadata server 10, 71

V

VALIDVARNAME= system option 85

W

WHERE clauses 67
filters as 27, 49

X

XML files
  exporting information maps in 12
  importing information maps from 13

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.