

Base SAS[®] 9.3 Guide to Information Maps



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *Base SAS® 9.3 Guide to Information Maps*. Cary, NC: SAS Institute Inc.

Base SAS® 9.3 Guide to Information Maps

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

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

For a hardcopy 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.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

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, July 2011

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

<i>What's New in the INFOMAPS Procedure and the Information Maps LIBNAME Engine for SAS 9.3</i>	v
<i>Accessibility Features of the INFOMAPS Procedure and the SAS Information Maps LIBNAME Engine</i>	vii
<i>Recommended Reading</i>	ix
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
Chapter 2 • INFOMAPS Procedure	5
Overview: INFOMAPS Procedure	6
Syntax: INFOMAPS Procedure	7
Examples: INFOMAPS Procedure	75
Chapter 3 • Using the SAS Information Maps LIBNAME Engine	81
What Does the Information Maps Engine Do?	81
Understanding How the Information Maps Engine Works	81
Advantages of Using the Information Maps Engine	85
What Is Required to Use the Information Maps Engine?	85
What Is Supported?	85
Chapter 4 • LIBNAME Statement for the Information Maps Engine	87
Using the LIBNAME Statement	87
Dictionary	87
Examples	93
Chapter 5 • SAS Data Set Options for the Information Maps Engine	95
Using Data Set Options	95
Dictionary	95
Chapter 6 • Hints and Tips for Using the INFOMAPS Procedure or the Information Maps Engine	101
Hints and Tips for Using the INFOMAPS Procedure	101
Hints and Tips for Using the Information Maps Engine	102
Chapter 7 • Example: Using the INFOMAPS Procedure and the Information Maps Engine	105
About This Example	105
Step 1: Create a Library Definition in the SAS Metadata Server	106
Step 2: Set the Metadata System Options and a Macro Variable	106
Step 3: Register Data Using the METALIB Procedure	107
Step 4: Create an Information Map Using the INFOMAPS Procedure	108
Step 5: Retrieve the Data Associated with the Information Map Using the Information Maps Engine	115
Step 6: View the Data Items and Filters Using the CONTENTS Procedure	115
Step 7: Print the Data from the Information Map	117
Step 8: Analyze the Data in SAS and Produce an ODS Report	119

Appendix 1 • SQL Dictionary Tables for the Information Maps Engine	123
Using SQL DICTIONARY Tables	123
DICTIONARY.INFOMAPS Table	123
DICTIONARY.DATAITEMS Table	124
DICTIONARY.FILTERS Table	125
Appendix 2 • SAS Tracing and the Information Maps Engine	127
SAS System Options for Diagnostic Messages	127
Dictionary	127
Displaying Detailed and Summary Timing Information	128
Glossary	129
Index	135

What's New in the INFOMAPS Procedure and the Information Maps LIBNAME Engine for SAS 9.3

Overview

The INFOMAPS procedure in Base SAS software has the following changes and enhancements:

- Support for specifying data sources that are used in every query that is generated from an information map.
- Support for using an advanced model for the join strategy during query generation when the information map contains more than one measure data item derived from more than one data source.
- Support for assigning filters that are applied to data sources before they are used in the current information map, including assigning authorization-based filters for a specific user or group.
- Support for SAS identity properties that enable user-specific information to be evaluated in filters.
- Support for changing the access permissions of the information map.
- Support for localizing information map properties for multiple locales.
- Support for updating the information map currently in memory without reloading the information map definition from the metadata server and for closing the information map currently in memory without ending the procedure.
- Support for controlling whether information maps created by previous SAS releases are updated when saved.
- Support for halting the procedure when an error occurs during batch processing.

The Information Maps LIBNAME Engine in Base SAS software has been enhanced to recognize the access permissions of the information map when data is accessed using the engine.

INFOMAPS Procedure Features

The following statements are new:

CLOSE INFOMAP

enables you to close the current information map.

EXPORT LOCALIZABLE_PROPERTIES and **IMPORT LOCALIZED_PROPERTIES** support localizing information map properties for multiple locales.

INSERT IDENTITY_PROPERTY and **DELETE IDENTITY_PROPERTY**
enable you to insert a SAS identity property into the current information map and to remove one or more SAS identity properties from the current information map.

SET ASSIGNED_FILTERS
enables you to assign filters that are applied to data sources before they are used in the current information map.

UPDATE CURRENT_INFOMAP
enables you to update the information map in memory without reloading the information map definition from the metadata server.

UPDATE MAP_PERMISSIONS
enables you to change the access permissions of the information map and to assign authorization-based filters for a specific user or group.

The following statements have been enhanced:

PROC INFOMAPS
has a new **ERRORSTOP** option that enables you to control whether the procedure halts when an error occurs during batch processing.

INSERT DATASOURCE and **UPDATE DATASOURCE**
have a new **REQUIRED_DATASOURCE=** option that enables you to specify that the data source is used in every query that is generated from an information map.

UPDATE INFOMAP
has a new **REQUIRED_DATASOURCES=** option that enables you to manage the list of required data sources for an information map.

INSERT FILTER and **UPDATE FILTER**
have a new **HIDDEN=** option that enables you to specify whether the filter is hidden from users of the information map.

NEW INFOMAP and **UPDATE INFOMAP**
have a new **JOIN_MODEL=** option that enables you to control whether a basic or advanced model is used for the join strategy during query generation.

SAVE
has new **ALLOW_MAJOR_VERSION_UPGRADE=** and **ALLOW_MINOR_VERSION_UPGRADE=** options that enable you to control the migration of information maps created by previous SAS releases.

Information Maps LIBNAME Engine Features

The Information Maps LIBNAME Engine now honors the user's Read permission setting for the information map and its data sources. A user is not allowed to access data via the Information Maps engine if the user's Read permission for the information map or its data sources in the metadata server is DENY.

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 accessibility@sas.com or call SAS Technical Support.

Recommended Reading

- *The Little SAS Book: A Primer*
- *Step-by-Step Programming with Base SAS Software*
- *SAS Language Reference: Concepts*
- *SAS Statements: Reference*
- *SAS Data Set Options: Reference*
- *SAS System Options: Reference*
- *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, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

x *Recommended Reading*

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

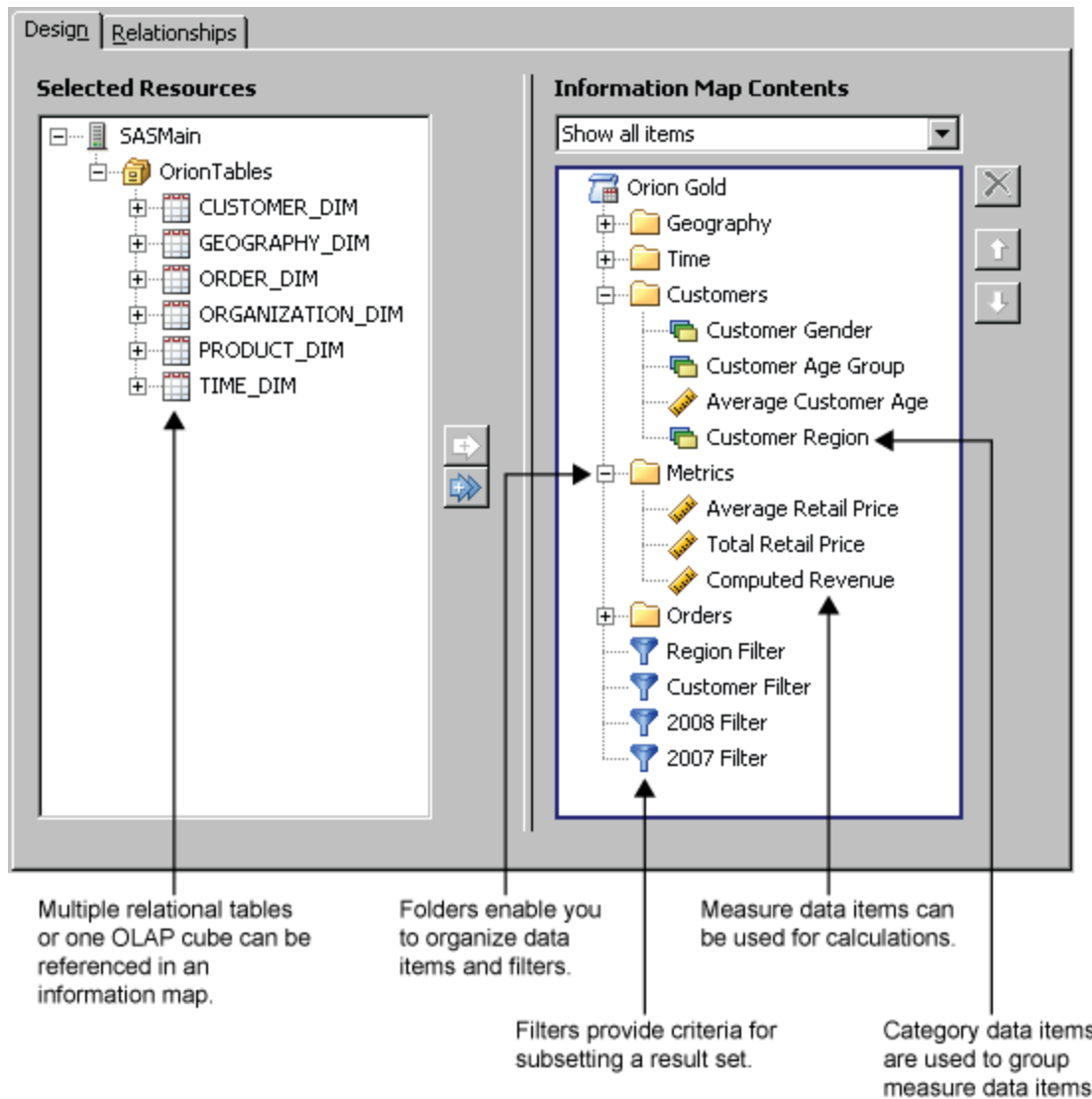
What Is a SAS Information Map?

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 [Chapter 2](#), “[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 Report Studio

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

Chapter 2

INFOMAPS Procedure

Overview: INFOMAPS Procedure	6
Syntax: INFOMAPS Procedure	7
PROC INFOMAPS Statement	7
CLOSE INFOMAP Statement	10
DELETE IDENTITY_PROPERTY Statement	11
DELETE INFOMAP Statement	11
EXPORT Statement	12
EXPORT LOCALIZABLE_PROPERTIES Statement	13
IMPORT Statement	14
IMPORT LOCALIZED_PROPERTIES Statement	15
INSERT DATAITEM Statement	17
INSERT DATASOURCE Statement	28
INSERT FILTER Statement	32
INSERT FOLDER Statement	35
INSERT IDENTITY_PROPERTY Statement	37
INSERT RELATIONSHIP Statement	39
LIST Statement	41
MOVE DATAITEM Statement	44
MOVE FILTER Statement	44
MOVE FOLDER Statement	45
NEW INFOMAP Statement	45
SAVE Statement	49
SET ASSIGNED_FILTERS Statement	51
SET STORED_PROCESS Statement	52
UPDATE CURRENT_INFOMAP Statement	52
UPDATE DATAITEM Statement	56
UPDATE DATASOURCE Statement	62
UPDATE FILTER Statement	63
UPDATE FOLDER Statement	65
UPDATE INFOMAP Statement	66
UPDATE MAP_PERMISSIONS Statement	70
UPDATE RELATIONSHIP Statement	72
Examples: INFOMAPS Procedure	75
Example 1: Creating a Basic Information Map	75
Example 2: Creating an Information Map with Relationships and Filters	76
Example 3: Aggregating a Data Item	78

Overview: INFOMAPS Procedure

The INFOMAPS procedure enables you to create information maps programmatically. You can also use the procedure to modify an existing information map by adding new data sources, 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 that 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 [“Using the LIBNAME Statement” on page 87](#).

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 <options>;
CLOSE INFOMAP;
DELETE IDENTITY_PROPERTY _ALL_ | ID="identity-property-ID";
DELETE INFOMAP "information-map-name" <options>;
EXPORT FILE=fileref | "physical-location" <options>;
EXPORT LOCALIZABLE_PROPERTIES FILE="physical-location" <option>;
IMPORT FILE="physical-location";
IMPORT LOCALIZED_PROPERTIES FILE="base-location"
  LOCALES=(locale-1<...>locale-n);
INSERT DATAITEM <options>;
INSERT DATASOURCE <options>;
INSERT FILTER CONDITION="conditional-expression" <options>;
INSERT FOLDER "folder-name" <options>;
INSERT IDENTITY_PROPERTY PROPERTY=property-keyword
  <ID="identity_property-ID">;
INSERT RELATIONSHIP <options>;
LIST <options>;
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> <option>;
NEW INFOMAP "information-map-name" <options>;
SAVE <options>;
SET ASSIGNED_FILTERS DEFINITION=(<data-source-filters-1
  <...>data-source-filters-n>>);
SET STORED_PROCESS NAME="stored-process-name" <option>;
UPDATE CURRENT_INFOMAP <options>;
UPDATE DATAITEM "data-item-ID" <options>;
UPDATE DATASOURCE "data-source-ID" <options>;
UPDATE FILTER "filter-ID" <options>;
UPDATE FOLDER "folder-name" <options>;
UPDATE INFOMAP "information-map-name" <options>;
UPDATE MAP_PERMISSIONS GROUP="identity" | USER="identity"
  permission-specification-1<...>permission-specification-n>;
UPDATE RELATIONSHIP "relationship-ID" <options>;

```

PROC INFOMAPS Statement

Connects to the specified metadata server.

Syntax

PROC INFOMAPS <options>;

Summary of Optional Arguments

DOMAIN="authentication-domain"

specifies an authentication domain to associate the user ID and password with.

ERRORSTOP | **NOERRORSTOP**

MAPPATH="location"

specifies the location within the metadata server for the information map that you want to create, open, or delete.

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.

METAPASS="password"

specifies the password that corresponds to the user ID that connects to the metadata server.

METAPORT=port-number

specifies the TCP port that the metadata server is listening to for connections.

METASERVER="address"

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

METAUSER="user-ID"

specifies the user ID to connect to the metadata server.

Optional Arguments

DOMAIN="authentication-domain"

specifies an authentication domain to associate the user ID and password with.

Default: 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*.

ERRORSTOP | **NOERRORSTOP**

specifies whether the INFOMAPS procedure terminates when a syntax or run-time error is encountered while the procedure is executed in batch mode. By default, the procedure continues to execute subsequent statements as it does in interactive mode, which can lead to the creation of invalid information maps. The **ERRORSTOP** option causes the procedure to exit with the **SYSERR** automatic macro variable set to **XXEXITERROR** when an error is encountered.

Default: NOERRORSTOP

Restriction: The **ERRORSTOP** option has no effect when the INFOMAPS procedure is used in interactive mode.

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 *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.

Alias: PORT=

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

Example:

```
metaport=8561
```

METASERVER="address"

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

Alias:

SERVER=
HOST=

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

Example:

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

METAUSER="user-ID"

specifies the user ID to connect to the metadata server. 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 *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=

Restrictions:

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.

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"**.

Note: The user ID is not case sensitive.

Example:

```
metauser="myUserID"
```

Example

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

CLOSE INFOMAP Statement

Closes the current information map.

Syntax

CLOSE INFOMAP;

Details

The current information map is automatically closed when you submit a NEW INFOMAP or UPDATE INFOMAP statement. The CLOSE INFOMAP statement enables you to explicitly close the current information map without ending the INFOMAPS procedure. This is useful when you want to perform operations that would otherwise affect the current information map, such as submitting an EXPORT LOCALIZED_PROPERTIES statement.

DELETE IDENTITY_PROPERTY Statement

Deletes one or more SAS identity properties from the current information map.

Syntax

```
DELETE IDENTITY_PROPERTY _ALL_ | ID="identity-property-ID";
```

Required Arguments

ALL

removes all SAS identity properties from the current information map.

ID="*identity-property-ID*"

removes the specified SAS identity property from the current information map.

Details

If an SAS identity property is currently being referenced by a business object, then the DELETE IDENTITY_PROPERTY statement is ignored and a warning message is written to the log.

For more information about SAS identity properties, see [INSERT IDENTITY_PROPERTY Statement on page 37](#).

DELETE INFOMAP Statement

Deletes an information map from the SAS folders tree.

Syntax

```
DELETE INFOMAP "information-map-name" <options>;
```

Required Argument

"*information-map-name*"

specifies the name of the information map to delete.

Optional Argument

MAPPATH="*location*"

specifies the location within the SAS folders tree for the information map to delete.

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

Examples

Example 1

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

Example 2

```
delete infomap "myMap";
```

EXPORT Statement

Exports an information map in its XML representation.

Note: 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.

Syntax

```
EXPORT FILE=fileref | "physical-location" <options>;
```

Summary of Optional Arguments

INFOMAP "*information-map-name*"

specifies the name of the information map to export.

MAPPATH="*location*"

specifies the location within the SAS folders tree for the information map to export.

Required Argument

FILE=*fileref*

FILE="*physical-location*"

specifies an external file to which to export an XML representation of the information map.

Note: If the specified external file already exists, it is replaced.

Optional Arguments

INFOMAP "*information-map-name*"

specifies the name of the information map to export.

Default: 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.

Details

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

Example 1

```
/* 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";
```

Example 2

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

EXPORT LOCALIZABLE_PROPERTIES Statement

Exports the localizable properties of one or more information maps to an external file.

Alias: EXPORT LOC_PROPERTIES

Syntax

```
EXPORT LOCALIZABLE_PROPERTIES FILE="physical-location" <option>;
```

Required Argument

FILE="*physical-location*"

specifies the external file to which the localizable properties of one or more information maps are exported. The extension **.locprop** is added to the specified pathname if it is not included in the specified value.

Note: If the specified external file already exists, it is replaced.

Optional Argument

INFOMAP="*location*" <*keep-drop-list*>

INFOMAP=("*location-1*" <*keep-drop-list-1*> <... "location-n" <*keep-drop-list-n*>>

specifies the location and names of the information maps for which localized properties are exported.

location

specifies a location within the SAS folders tree that contains the information maps for which you want to export properties.

keep-drop-list

specifies the names of information maps from the specified location that you want to include in or exclude from the export process. By default, properties are exported for all information maps at the specified location. The *keep-drop-list* value has the following form:

```
(information-map-name-1 <...information-map-name-n>) </KEEP | /DROP>
```

By default, the list specifies the information maps from the specified location for which properties are exported. If you specify the `/DROP` argument, then the specified information maps are excluded and properties for all other information maps from the specified location are exported.

Alias: INFOMAPS=

Default: If you do not specify the `INFOMAP=` option, then the localizable properties for the current information map are exported. When exporting the properties for the current information map, it is a good practice to use a `SAVE` statement before the `EXPORT` statement. This ensures that all pending changes are saved.

Note: An error occurs if you specify the `INFOMAP=` option while an information map is open. You can use the `CLOSE INFOMAP` statement to close the current information map if you want to export the properties of a different information map.

Details

The `EXPORT LOCALIZABLE_PROPERTIES` statement generates text files that can be used as source by translators who want to localize the properties of an information map for other languages. The localized versions of the properties files can then be read back into the information maps using the `IMPORT LOCALIZED_PROPERTIES` statement. For more information, see [IMPORT LOCALIZED_PROPERTIES Statement on page 15](#).

Example

```
/* Export the localizable properties of the current */
/* information map to an external file */
export localizable_properties file ="c:\test";

/* Export the localizable properties of two information maps */
/* (Flights and Passengers) in the /Shared Data/Travel folder. */
export localizable_properties
    file="c:\myProp"
    infomaps ="/Shared Data/Travel" ("Flights" "Passengers");
```

IMPORT Statement

Imports an information map from an external XML file.

Note: 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.

Syntax

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

Required Argument

FILE=*fileref*

FILE="*physical-location*"

specifies the fileref or physical location of an XML file from which an information map is imported.

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";
```

IMPORT LOCALIZED_PROPERTIES Statement

Imports localized properties from external files into one or more existing information maps.

Alias: IMPORT LOC_PROPERTIES

Syntax

```
IMPORT LOCALIZED_PROPERTIES FILE="base-location"
```

```
  LOCALES=(locale-1<...locale-n>);
```

Required Arguments

FILE=*base-location*

specifies the path and base name for one or more external files that contain localized properties for one or more information maps. The complete names for the imported files are formed by adding the locale values specified in the LOCALES= argument and the file extension `.locprop` to the specified value. For example, if the filename in the *base-location* value is `mapprops` and `fr_FR` is specified as one of the LOCALES= values, then one of the files that the import process attempts to read is `mapprops_fr_FR.locprop`.

Note: You can use the EXPORT LOCALIZABLE_PROPERTIES statement to create files that contain the localizable properties for one or more information maps. Those property files can then serve as the source for creating the localized properties files that this statement imports.

LOCALES=(*locale-1* <...*locale-n*>

specifies one or more locales for the localized properties files to import. For more information about locale values, see *SAS National Language Support (NLS): Reference Guide*.

Details

For each locale value specified in the LOCALES= argument, the import process expects to find a localized properties file for that locale at the path and with the base name specified in the FILE= argument. A localized properties file is a text file containing localized versions of information map properties such as object labels and descriptions. You can use the EXPORT LOCALIZABLE_PROPERTIES statement to create a file that contains the localizable properties of one or more information maps. You can then create new copies of that file in which the values of the properties are translated for a given locale.

The set of localized properties files must use a common base name with the locale added to each file. Localized properties files must use the file extension `.locprop`. Thus, the French version in a set of localized properties files with the base name `localmap` would be `localmap_fr_FR.locprop` and the German version would be `localmap_de_DE.locprop`.

The import process reads the localized properties files and applies the specified localized properties to the information maps specified in the files. If an information map is open when the import process is started, then the procedure imports the localized properties for only the current information map. You should issue a Save command after the import to store the imported properties. If you want to import localized properties for all of the information maps that are specified in the external files, then you must close the current information map first. You can use the CLOSE INFOMAP statement to close the current information map.

Example

```
/* Update an information map by adding French and      */
/* Canadian French versions of localizable properties */
update infomap "my testmap"
      mappath="/Users/myUserID/My Folder";
import localized_properties
      file="c:/locale/mapprops"
      locales=(fr_FR fr_CA);
save;
```

In this example, the import process expects to find the following localized properties files to import:

- c:/locale/mapprops_fr_FR.locprop
- c:/locale/mapprops_fr_CA.locprop

INSERT DATAITEM Statement

Inserts a data item into the current information map.

Syntax

- Form 1: **INSERT DATAITEM** COLUMN="*data-source-ID*".*column-name* <*options*>;
- Form 2: **INSERT DATAITEM** EXPRESSION="*expression-text*" <*options*>;
- Form 3: **INSERT DATAITEM** HIERARCHY="*dimension*".*hierarchy* <*options*>;
- Form 4: **INSERT DATAITEM** MEASURE="*OLAP-measure*" <*options*>;

Summary of Optional Arguments

ACTIONS=(*actions-list*)

tells an application 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.

AGGREGATION=*aggregate-function*

specifies how a measure data item is aggregated when it is used in a query.

AGGREGATIONS_DROP_LIST=(*aggregate-function-list*)

removes one or more functions from the set of aggregate functions available to a data item.

AGGREGATIONS_KEEP_LIST=(*aggregate-function-list*)

specifies the aggregate functions that are available to a data item.

CLASSIFICATION=CATEGORY | MEASURE

specifies whether the data item is a category or a measure.

CUSTOM_PROPERTIES=(*custom-properties-list*)

specifies additional properties for the data item.

DESCRIPTION="*descriptive-text*"

specifies the description of the data item, which can be viewed by the information map consumer.

FOLDER="*folder-name*" <CREATE>

FOLDER="*folder-location*" <CREATE>

specifies the folder in the information map into which to insert the data item.

FORMAT="*format-name*"

specifies the SAS format of the data item.

ID="*data-item-ID*"

specifies the ID assigned to the data item being inserted.

NAME="*data-item-name*"

specifies the name assigned to the data item in the information map.

TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP

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

VALUE_GENERATION=NONE | DYNAMIC | (*custom-values-list*)

specifies what method an application that uses the information map is to use in generating a list of values for this data item to present to a user.

Required Arguments

The INSERT DATAITEM statement must include one of the following arguments:

COLUMN="data-source-ID"."column-name"

specifies a column.

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_";
```

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. The form of the *expression-text* value depends on the type of data item being defined:

For relational data items

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>>.

For OLAP data items

Expressions 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]
```

Note: If you use 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 and the EXPCOLUMNLEN= data set option. By default, the Information Maps engine sets the data length for columns of these data items to 32 characters. You can use the EXPCOLUMNLEN= statement option or data set option to change the default length. For more information about the EXPCOLUMNLEN= statement option, see “LIBNAME Statement” on page 87. For more information about the EXPCOLUMNLEN= data set option, see “EXPCOLUMNLEN= Data Set Option” on page 96.

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.

dimension

is the name of a dimension in the current OLAP data source.

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 argument 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="OLAP-measure"

specifies a physical measure.

OLAP-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 argument 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.

Optional Arguments**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 selection 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: All actions are enabled (RANK, SORT, FILTER, or NAVIGATE) unless 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 *aggregate-function* value are shown in [Table 2.1 on page 21](#). 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 SQL Procedure User's Guide*.

The special value **InternalAggregation** specifies that the values of the measure data item are aggregated by a nonadditive expression. A nonadditive 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 nonadditive expression. If you specify that a data item has a nonadditive 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.

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

- If the expression type for the data item is numeric, then the default aggregate function is SUM.
- If the expression type for the data item is character, date, time, or timestamp, then the default aggregate function is COUNT.
- If the data item is based on a measure, then the default aggregate function is InternalAggregation.

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	✓

Function	Definition	Available to nonnumeric item that is a measure
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 (nonadditive)	✓
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	✓

Function	Definition	Available to nonnumeric item that is a measure
NMISSDistinct	number of distinct missing values	✓
PRT	probability of a greater absolute value of Student's t	
PRTDistinct	probability of a greater absolute value of Student's t of distinct values	
RANGE	range of values	
RANGEDistinct	range of distinct values	
STD	standard deviation	
STDDistinct	standard deviation of distinct values	
STDERR	standard error of the mean	
STDERRDistinct	standard error of the mean of distinct values	
SUM	sum of values	
SumDistinct	sum of distinct values	
T	Student's t value for testing the hypothesis that the population mean is zero	
TDistinct	Student's t value for testing the hypothesis that the population mean of distinct values is zero	
USS	uncorrected sum of squares	
USSDistinct	uncorrected sum of squares for distinct values	
VAR	variance	
VarDistinct	variance of distinct values	

Restrictions:

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

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**.

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.

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 21](#) 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.

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.

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

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 21](#) 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.

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

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

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* value 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.

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

Requirement: Property names must be unique. If a specified 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.

property-value

specifies the value of the property.

description

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

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*" </CREATE>

FOLDER="*folder-location*" </CREATE>

specifies the folder in the information map into which to insert the data item.

The following rules apply:

- 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=

Restrictions:

The following characters are not valid in a folder name:

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

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, as follows:
 - If the data item is inserted from a physical column using INSERT DATAITEM with the COLUMN= argument specified or using INSERT DATASOURCE with either the `_ALL_` or the COLUMNS= option specified, then the ID is generated from either the SAS name or label of the physical column.

The settings of the USE_LABELS=, REPLACE_UNDERSCORES=, and INIT_CAP= options determine the exact value and casing of the ID.

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

Restrictions:

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.

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.

Restrictions:

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.

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* value 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")
)
```

Example

```
/* 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.

Syntax

Form 1: **INSERT DATASOURCE** SASSERVER="*application-server-name*"
TABLE="*library*".*table* <*options*>;

Form 2: **INSERT DATASOURCE** SASSERVER="*application-server-name*"
 CUBE=<"*schema*".>"*cube*" <*options*>;

Summary of Optional Arguments

ALL

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

COLUMNS=(*column-1* <...*column-n*>)

specifies one or more physical column names as defined in the specified table.

DESCRIPTION="*descriptive-text*"

specifies the description of the data source, which can be viewed by the information map consumer.

ID="*data-source-ID*"

specifies the ID assigned to the data source.

NAME="*data-source-name*"

enables you to specify a descriptive name for each data source inserted in an information map.

REQUIRED_DATASOURCE=YES | NO

specifies whether the data source is added to the list of required data sources for the information map.

Required Arguments

The INSERT DATASOURCE statement must include the SASSERVER= argument and either the CUBE= or TABLE= argument.

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.

Restrictions:

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.

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

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=

Restrictions:

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.

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.

Table names are case sensitive.

Optional Arguments

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.

Restrictions:

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.

Table and cube names are case sensitive.

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-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.

REQUIRED_DATASOURCE=YES | NO

specifies whether the data source is added to the list of required data sources for the information map. By default, the data source and its assigned filters are included in a query only when a data item that references the data source is explicitly selected for the query. Specify YES if you want the data source and any associated assigned filters to be used in every query that is generated from the information map.

Default: NO

Note: After the data source is created, you can use the REQUIRED_DATASOURCES= option in the UPDATE CURRENT_INFOMAP or UPDATE INFOMAP statements to control whether it is required.

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

Example 1

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

Example 2

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

Example 3

```
/* 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.

Note: For relational databases, a filter is a WHERE clause.

Syntax

```
INSERT FILTER CONDITION="conditional-expression" <options>;
```

Summary of Optional Arguments

CUSTOM_PROPERTIES=(*custom-properties-list*)

specifies additional properties for the filter.

DESCRIPTION="*descriptive-text*"

specifies the description of the filter to be inserted.

FOLDER="*folder-name*" <CREATE>

FOLDER="*folder-location*" <CREATE>

specifies the folder in the information map into which to insert the filter.

HIDDEN=YES | NO

specifies whether to hide the filter from users of the information map.

ID="*filter-ID*"

specifies the ID of the filter to insert.

`NAME="filter-name"`

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

Required Argument

`CONDITION="conditional-expression"`

The following rules apply to the *conditional-expression* value:

For 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>>.

For 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.

Optional Arguments

`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* value 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.

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

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

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

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" </CREATE>

FOLDER="folder-location" </CREATE>

specifies the folder in the information map into which to insert the filter. The following rules apply:

- 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=

Restrictions:

The following characters are not valid in a folder name:

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

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

HIDDEN=YES | NO

specifies whether to hide the filter from users of the information map. By default, the filter is available to users of the information map. Specify HIDDEN=YES if you want to hide the filter from users (for example, when the filter is used as an assigned filter for a data source and therefore should not be applied again).

Default: NO

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.

Restrictions:

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.

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

Example 1

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

Example 2

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

Example 3

```
/* 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.

Syntax

```
INSERT FOLDER "folder-name" <options>;
```

Summary of Optional Arguments

CUSTOM_PROPERTIES=(*custom-properties-list*)

specifies additional properties for the folder.

DESCRIPTION="descriptive-text"

specifies the description of the folder that is created.

LOCATION="parent-folder-name" </CREATE>

LOCATION="parent-folder-location" </CREATE>

specifies the parent folder of the folder that you are inserting into the information map.

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.

Optional Arguments

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* value 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.

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

Requirement: Property names must be unique. If a specified 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.

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*" </CREATE>

LOCATION="*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=

Restrictions:

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

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

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

Examples

Example 1

```
insert folder "measures";
```

Example 2

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

Example 3

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

INSERT IDENTITY_PROPERTY Statement

Adds a SAS identity property or all available SAS identity properties to the current information map.

Syntax

```
INSERT IDENTITY_PROPERTY PROPERTY=property-keyword
<ID="identity_property-ID">;
```

Required Argument

PROPERTY=*property-keyword*

inserts a specified SAS identity property (or all available SAS identity properties) into the current information map. Once the properties are inserted into the information map, they can be used in filter definitions that are created with the INSERT FILTER or UPDATE FILTER statements.

The *property-keyword* value can be one of the following:

ALL

inserts all of the following SAS identity properties into the current information map.

EXTERNAL_IDENTITY

inserts the SAS.ExternalIdentity property. When a filter that uses this property is executed, the connected client's identity value (for example, employee ID) is substituted in the filter expression. This property is often useful because its values are likely to match user information in your data.

Notes:

An identity can have more than one external identity value. However, only the first value is returned.

Unlike the values for other SAS identity properties, values for this property are not always populated in the metadata. If an identity has no external identity values, then no value is returned for this property.

IDENTITY_GROUP_NAME

inserts the SAS.IdentityGroupName property. When a filter that uses this property is executed, the connected client's group name is substituted in the filter expression. If a user logs on with an ID that is stored in a login on a group definition, then the name of the group that owns that login is returned. If a user logs on with a user ID that is not stored in the metadata, then the PUBLIC group is returned.

Restriction: This property is not supported if client-side pooling is used.

Note: This property is useful only in the unusual circumstance where a user logs on with the user ID that is defined for a group login. In almost all cases, a user logs on with a user ID that is defined for an individual user definition. Not all applications allow a group to log on.

IDENTITY_GROUPS

inserts the SAS.IdentityGroups property. When a filter that uses this property is executed, a list of the names of the user groups and roles to which the connected client belongs (directly, indirectly, or implicitly) is substituted in the filter expression.

IDENTITY_NAME

inserts the SAS.IdentityName property. When a filter that uses this property is executed, the connected client's user name or group name is substituted in the filter expression. This property is a generalization of SAS.PersonName and SAS.IdentityGroupName.

PERSON_NAME

inserts the SAS.PersonName property. When a filter that uses this property is executed, the connected client's name is substituted in the filter expression.

Note: Users who belong only to the PUBLIC group do not have PersonNames. For these users, no value is returned for this property.

USERID

inserts the SAS.Userid property. When a filter that uses this property is executed, the connected client's authenticated user ID, normalized to the uppercase format *USERID* or *USERID@DOMAIN*, is substituted in the filter expression.

Note: If you attempt to insert an SAS identity property that already exists in the information map, then the INSERT IDENTITY_PROPERTY statement is ignored and a warning message is written to the log.

Optional Argument

ID="*identity_property-ID*"

specifies the identifier that is assigned to the specified SAS identity property in the information map. To reference an SAS identity property in a conditional expression in an INSERT FILTER or UPDATE FILTER statement, you must specify its ID.

Note:

Default: If you omit the ID= option, then a default identifier is assigned to the inserted SAS identity property and a warning message is written to the log.

Note: When you specify PROPERTY=_ALL_, the specified ID value is used as the base name for the default identifiers that are assigned to the properties.

Details

It is often necessary to make per-person access distinctions. You can make a separate filter for each user (such as **where name="joe"**). However, if you have more than a few users, this approach quickly becomes cumbersome. The more efficient alternative is to create a dynamic filter (such as **where name="&name;"**) that can discover and insert the correct, user-specific value into the WHERE expression each time access is requested.

To create a dynamic filter, use an SAS identity property as the value against which values in the target data are compared. This list explains how the substitution works:

1. Each SAS identity property corresponds to a characteristic (such as name, user ID, or external identity).
2. Each user's values for these characteristics (such as `joe`, `WinXP\joe`, or `607189`) are stored in the metadata.
3. The SAS identity property is aware of the user ID with which a client authenticated and can locate information that is stored in the metadata for that user ID.
4. Each time it receives a request, the SAS identity property substitutes a user-specific value into the filter expression.

Note: In certain circumstances, a connecting identity might not have a value for the SAS identity property that you are using. This can happen with the `ExternalIdentity` property (sometimes), the `IdentityGroupName` property (almost always), or the `PersonName` property (rarely). When a connecting user doesn't have a value for the property that a query uses, an empty string is returned.

For more information about SAS identity properties and when to use them, see the section on fine-grained controls in the “Authorization Model” chapter in the *SAS Intelligence Platform: Security Administration Guide*. For information about using SAS identity properties to implement row-level access control to the data in an information map, see *SAS Guide to BI Row-Level Permissions*.

Example

```
insert identity_property
  property=person_name
  id="SasPersonName";

insert filter "myFilter"
  condition="<<SECURITY.USER_ID>>=(<<SasPersonName>>";
```

INSERT RELATIONSHIP Statement

Inserts a join into the current information map.

Syntax

```
INSERT RELATIONSHIP CONDITION="conditional-expression"
LEFT_TABLE="data-source-ID-1" RIGHT_TABLE="data-source-ID-2" <options>;
```

Summary of Optional Arguments

`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.

`CUSTOM_PROPERTIES=(custom-properties-list)`

specifies additional properties for the relationship.

`DESCRIPTION="descriptive-text"`

specifies the description of the relationship that is created.

`ID="relationship-ID"`

specifies the ID of the relationship to be inserted.

JOIN=INNER | LEFT | RIGHT | FULL

specifies the type of join.

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.

Optional Arguments

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* value 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.

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

Requirement: Property names must be unique. If a specified 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.

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.

Restrictions:

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.

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. The type can be one of the following:

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.

Syntax

LIST <object-types>;

Optional Argument

object-types

specifies one or more of the following information map object types for which properties are listed:

ALL

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

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, description, and whether the data source is designated as required.

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.

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

Example

The following are representative sections of the LIST statement results for the information map in “[Example: Using the INFOMAPS Procedure and the Information Maps Engine](#)” on page 105.

Log 2.1 LIST Statement Information for Data Sources

```
Total datasources: 3

Data source: SAS Sample Data.EMPINFO
ID: Empinfo
Name: EMPINFO
Description:
Required data source: NO

Data source: SAS Sample Data.JOBCODES
ID: Jobcodes
Name: JOBCODES
Description:
Required data source: NO
...
```

Log 2.2 LIST Statement Information for Data Items

```

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:
  ...

```

Log 2.3 LIST Statement Information for Filters

```

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")
  ...

```

Log 2.4 LIST Statement Information for Relationships

```

Total relationships: 2

  Relationship ID: JOIN_10
  Left data source: SAS Sample Data.EMPINFO
  Right data source: SAS Sample Data.JOBCODES
  Cardinality: UNKNOWN
  Join type: INNER
  Join expression: (<<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>>)

  Relationship ID: JOIN_11
  Left data source: SAS Sample Data.EMPINFO
  Right data source: SAS Sample Data.SALARY
  Cardinality: UNKNOWN
  Join type: INNER
  Join expression: (<<Empinfo.Identification Number>>=<<Salary.Identification
Number>>)

```

MOVE DATAITEM Statement

Moves one or more data items to a new location.

Syntax

```
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= argument 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.

Syntax

```
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 or you can use the ID_LIST= argument to specify multiple filters.

Tip: The filters 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 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.

Syntax

```
MOVE FOLDER "folder-name" NEW_LOCATION="new-folder-location" </CREATE>
<option>;
```

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.

Optional Argument

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.

Syntax

```
NEW INFOMAP "information-map-name" <options>;
```

Summary of Optional Arguments

AUTO_REPLACE=YES | NO

indicates whether the specified information map is automatically replaced if it already exists.

CREATE_TARGET_FOLDER=YES | NO

specifies whether to automatically create a folder when inserting all data items from a data source.

CUSTOM_PROPERTIES=(*custom-properties-list*)

specifies additional properties for an information map.

DESCRIPTION="*descriptive-text*"

specifies the description of the information map that is created.

INIT_CAP=YES | NO

specifies whether to capitalize the first letter of each word in the data item name.

JOIN_MODEL=BASIC | ADVANCED

specifies the join strategy that is used during query generation.

MAPPATH="*location*" </CREATE>

specifies the location within the SAS folders tree for the new information map.

REPLACE_UNDERSCORES=YES | NO

specifies whether to replace each underscore (`_`) character in the data item name with a blank space.

USE_LABELS=YES | NO

specifies whether to create the data item name using the column label (if available) instead of the column name.

VERIFY=YES | NO

specifies whether the INFOMAPS procedure verifies the validity of data items, filters, and relationships in subsequent Insert or Update operations.

Required Argument

"information-map-name"

specifies the name of the new information map.

Restrictions:

The following characters are not valid in information map names:

- null characters
- non-blank nonprintable characters

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

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.

Optional Arguments

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

Note: The information map is not replaced until you successfully save the map.

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 INSERT DATASOURCE statements 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. Custom properties are supported by specific SAS applications such as SAS Marketing Automation. The form of the *custom-properties-list* value 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.

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

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

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

JOIN_MODEL=BASIC | ADVANCED

specifies the join strategy that is used during query generation. By default, the INFOMAPS procedure applies a basic join strategy that is appropriate when the information map is based on a simple star schema (a single fact table with multiple dimensions). Specify JOIN_MODEL=ADVANCED if your information map contains more than one measure data item that is derived from more than one data source (multiple fact tables). The advanced join model prevents inflated query results when there is a MANY-to-MANY cardinality between selected measures that exist in two or more tables.

Default: BASIC

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

Note: The Save operation fails if the information map has no valid data source.

Syntax

SAVE <options>;

Summary of Optional Arguments

ALLOW_MAJOR_VERSION_UPGRADE=YES | NO

specifies whether the Save operation can overwrite an existing information map that has a lower major version number.

ALLOW_MINOR_VERSION_UPGRADE=YES | NO

specifies whether the Save operation can overwrite an existing information map that has a lower minor version number.

INFOMAP "information-map-name"

specifies the name to use for saving the current information map.

MAPPATH="location" </CREATE>

specifies the location within the SAS folders tree where the information map is to be saved.

Optional Arguments

ALLOW_MAJOR_VERSION_UPGRADE=YES | NO

specifies whether the Save operation can overwrite an existing information map that has a lower major version number. Information maps store major and minor version numbers that indicate the release of SAS software with which they were created. This enables applications to determine whether the information maps might include features that they do not support. By default, an error occurs when you attempt to replace an existing information map that has a lower major version number. Specify ALLOW_MAJOR_VERSION_UPGRADE=YES to enable the SAVE statement to replace an existing map that has a lower major version number. After a major version upgrade, an application that does not support information maps with the new major version number will no longer be able to access the information map.

Default: NO

ALLOW_MINOR_VERSION_UPGRADE=YES | NO

specifies whether the Save operation can overwrite an existing information map that has a lower minor version number. Information maps store major and minor version numbers that indicate the release of SAS software with which they were created. This enables applications to determine whether the information maps might include features that they do not support. By default, an error occurs when you attempt to replace an existing information map that has a lower minor version number. Specify `ALLOW_MINOR_VERSION_UPGRADE=YES` to enable the `SAVE` statement to replace an existing map that has a lower minor version number. After a minor version upgrade, an application that supports information maps with a lower minor version number but the same major version number is allowed to read in the information map. However, the application might ignore or incorrectly handle features of the newer version. It is up to the application to check the version number and decide whether to read the information map.

Default: NO

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.

Note: If you specified the `AUTO_REPLACE=YES` option when you created the information map, then the `SAVE` command will overwrite the existing information map without warning.

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

Examples

Example 1

```
/* 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";
```

Example 2

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

Example 3

```

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

```

SET ASSIGNED_FILTERS Statement

Assigns filters that are applied whenever the associated data sources are referenced in a query.

Syntax

SET ASSIGNED_FILTERS

DEFINITION=(*<data-source-filters-1<...data-source-filters-n>>*);

Required Argument

DEFINITION=(*<data-source-filters-1<...data-source-filters-n>>*)

defines the filter assignments for one or more data sources. The *data-source-filters* value has the following form:

```

"data-source-ID" (
  <PREFILTERS=("filter-ID-1<... "filter-ID-n">)>
  <RESULTS_FILTERS=("filter-ID-1<... "filter-ID-n">)>
)

```

data-source-ID

specifies the identifier of the data source to which the assigned filters are applied.

PREFILTERS=

specifies one or more filters that are applied before the specified data source is used.

RESULTS_FILTERS=

specifies one or more filters that are applied after query results are generated.

filter-ID

specifies the identifier of a filter defined in the information map.

Alias: DEF=

Note: To clear an existing assigned filters list, specify a blank list: DEFINITION=()

Details

Assigned filters are filters that are always applied to queries that reference the data source with which the filter is associated and to queries for which the associated data source is marked as required. There are the following two types of assigned filters:

prefilters

are applied before the data source is used. These filters subset the data in its associated data source before any other part of a query is run. The generated query contains a subquery that ensures that no data is consumed without the filter being applied.

results filters

are applied after a data source is used. These filters subset the query results after a join takes place. Any extra tables required by the filters are joined to the query. No sub-query is generated for these filters.

Note: Assigned filters can also be applied only for specific users or groups. For more information about assigning authorization-based filters that apply to a user or group, see [UPDATE MAP_PERMISSIONS Statement on page 70](#).

Examples

Example 1

```
/* Define assigned filters for the current information map by assigning the */
/* filter 'ageLessThan30' and prefilter 'FemaleOnly' to the Customer table */
/* and the filter '2008Q2' to the Order table. */
set assigned_filters
    definition=("Customer" (PREFILTERS=("FemaleOnly"))
               (RESULTS_FILTERS=("ageLessThan30"))
               "Order" (RESULTS_FILTERS=("2008Q2")));
```

Example 2

```
/* Reset the assigned filters of the current information map */
set assigned_filters def=();
```

SET STORED PROCESS Statement

Associates a stored process with the current information map.

Syntax

```
SET STORED PROCESS NAME="stored-process-name" <option>;
```

Required Argument

NAME="*stored-process-name*"

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

Optional Argument

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 CURRENT_INFOMAP Statement

updates the current information map.

Alias: UPDATE CURRENT_MAP

Syntax

UPDATE CURRENT_INFOMAP <options>;

Summary of Optional Arguments

CREATE_TARGET_FOLDER=YES | NO

specifies whether to automatically create a folder when inserting all items from a data source.

CUSTOM_PROPERTIES=NONE | ADD (custom-properties-list) | REMOVE (property-names-list) | <REPLACE> (custom-properties-list)

specifies how custom properties of the current information map are updated.

DESCRIPTION="descriptive-text"

specifies the description of the current information map.

INIT_CAP=YES | NO

specifies whether to capitalize the first letter of each word in the data item name.

JOIN_MODEL=BASIC | ADVANCED

specifies the join strategy that is used during query generation.

REPLACE_UNDERSCORES=YES | NO

specifies whether to replace each underscore (_) character in the data item name with a blank space.

REQUIRED_DATASOURCES=_ALL_ | NONE | ADD (data-source-list) | <REPLACE> (data-source-list) | REMOVE (data-source-list)

specifies how the list of required data sources for the current information map is updated.

USE_LABELS=YES | NO

specifies whether to create data item names using the column label (if available) instead of the column name.

VERIFY=YES | NO

specifies whether the INFOMAPS procedure verifies the validity of data items, filters, and relationships during the Update operation.

Optional Arguments

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) | REMOVE (property-names-list) | <REPLACE> (custom-properties-list)

specifies how custom properties of the current information map 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 current information map, if there are any.

ADD (*custom-properties-list*)

adds the specified custom properties to the current information map.

The form of the *custom-properties-list* value 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.

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

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

property-value

specifies the value of the property.

description

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

REMOVE (*property-names-list*)

removes the specified custom properties from the current information map.

The form of the *property-names-list* value is

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

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

<REPLACE> (*custom-properties-list*)

replaces the current custom properties for the current information map with the specified properties.

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

Default: REPLACE (if a custom properties list is specified with no operation keyword)

DESCRIPTION="descriptive-text"

specifies the description of the current information map.

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 INSERT DATASOURCE statements with either the `_ALL_` or the `COLUMNS=` option specified or INSERT DATAITEM statements 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.

JOIN_MODEL=BASIC | ADVANCED

specifies the join strategy that is used during query generation. By default, the INFOMAPS procedure applies a basic join strategy that is appropriate when the

information map is based on a simple star schema (a single fact table with multiple dimensions). Specify JOIN_MODEL=ADVANCED if your information map contains more than one measure data item that is derived from more than one data source (multiple fact tables). The advanced join model prevents inflated query results when there is a MANY-to-MANY cardinality between selected measures that exist in two or more tables.

Default: BASIC

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

Default: YES

REQUIRED_DATASOURCES= _ALL_ | NONE | ADD (data-source-list) | <REPLACE> (data-source-list) | REMOVE (data-source-list)

specifies how the list of required data sources for the current information map is updated. If you want a data source and any associated assigned filters to be used in every query that is generated from the current information map, then designate the data source as required for the current information map. If you do not designate a data source as required, then the data source and its assigned filters are included in a query only when a data item that references the data source is explicitly selected for the query.

Valid operations are the following:

ALL

specifies that all data sources in the current information map are required data sources.

NONE

removes the entire list of data sources, if any, that were previously assigned to the current information map.

ADD (data-source-list)

adds one or more data sources to the list of required data sources for the current information map. The *data-source-list* value has the following form:

("data-source-ID-1" <... "data-source-ID-n" >)

REMOVE (data-source-list)

removes one or more data sources from the list of required data sources for the current information map.

See the ADD operation for a description of the form of the *data-source-list* value.

<REPLACE> (data-source-list)

replaces the existing list of required data sources for the current information map, if any, with the specified list of required data sources.

See the ADD operation for a description of the form of the *data-source-list* value.

Note: Using the REPLACE operation changes the processing order of the required data sources.

Default: REPLACE (if a data sources list is specified with no operation keyword)

USE_LABELS=YES | NO

specifies whether to create data item names 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 INSERT DATASOURCE statements with either the `_ALL_` or the `COLUMNS=` option specified or INSERT DATAITEM statements 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 the current information map.

Default: YES

Details

The difference between the UPDATE CURRENT_INFOMAP statement and the UPDATE INFOMAP statement is that the UPDATE CURRENT_INFOMAP statement applies the specified changes directly to the information map currently in memory, whereas the UPDATE INFOMAP statement reloads the information map from the metadata server before applying the specified updates.

Example

```
update current_map
  description="Map for Domestic Customers"
  required_datasources=add('Product' 'Customer');
```

UPDATE DATAITEM Statement

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

Syntax

UPDATE DATAITEM "*data-item-ID*" <*options*>;

Summary of Optional Arguments

ACTIONS=*(actions-list)*

tells an application 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.

AGGREGATION=*aggregate-function*

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

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

CLASSIFICATION=`CATEGORY` | `MEASURE`

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

CUSTOM_PROPERTIES=`NONE` | **ADD** (*custom-properties-list*) | **REMOVE** (*property-names-list*) | **<REPLACE>** (*custom-properties-list*)

specifies how custom properties for the data item are updated.

DESCRIPTION="*descriptive-text*"

specifies the description of the data item.

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.

FORMAT="*format-name*"

specifies the SAS format of the data item.

ID="*data-item-ID*"

specifies the ID of the data item to update.

NAME="*data-item-name*"

specifies the name assigned to the data item in the information map.

TYPE=NUMERIC | CHARACTER | DATE | TIME | TIMESTAMP

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

VALUE_GENERATION=NONE | DYNAMIC | ADD (*custom-values-list*) |

REMOVE (*unformatted-values-list*) | <REPLACE> (*custom-values-list*)

specifies what method an application that uses the information map is to use in generating a list of values for this data item to present to a user.

Required Argument

"*data-item-ID*"

specifies the ID of the data item to update.

Optional Arguments

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 selection 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: 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 21](#) for a list of *aggregate-function* values 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 SQL Procedure User's Guide*.

The special value **InternalAggregation** specifies that the values of the measure data item are aggregated by a nonadditive expression. A nonadditive 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 nonadditive expression. If you specify that a data item has a nonadditive 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 DATAITEM statement as the AGGREGATIONS_LIST= option, then the INFOMAPS procedure sets the AGGREGATIONS_LIST= option first.

AGGREGATIONS_LIST= ALL | ADD (*aggregate-function-list*) | REMOVE (*aggregate-function-list*) | <REPLACE> (*aggregate-function-list*)

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

REMOVE (*aggregate-function-list*)

removes the specified aggregate functions from the aggregation list.

<REPLACE> (*aggregate-function-list*)

replaces the current aggregation list with the specified aggregate functions.

Default: REPLACE (if an aggregate function list is specified with no other keyword)

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

```
aggregations_list=replace(Freq FreqDistinct CSSDistinct)
```

Interactions:

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

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*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-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* value 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.

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

Requirement: Property names must be unique. If a specified 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.

property-value

specifies the value of the property.

description

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

REMOVE (*property-names-list*)

removes the specified custom properties from the data item.

The form of the *property-names-list* value is

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

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

<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* value.

Default: REPLACE (if a custom properties list is specified with no operation keyword)

DESCRIPTION=" *descriptive-text*"

specifies the description of the data item, which can be viewed by the information map consumer.

Alias: DESC=

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 use 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 and the EXPCOLUMNLEN= data set option. By default, the Information Maps engine sets the data length for columns of these data items to 32 characters. You can use the EXPCOLUMNLEN= statement option or data set option to change the default length. For more information about the EXPCOLUMNLEN= statement option, see [“LIBNAME Statement” on page 87](#). For more information about the EXPCOLUMNLEN= data set option, see [“EXPCOLUMNLEN= Data Set Option” on page 96](#).

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.

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

ID=" *data-item-ID*"

specifies the ID of the data item to update.

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*) | REMOVE (*unformatted-values-list*) | <REPLACE> (*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.

ADD (*custom-values-list*)

adds the specified custom values to the data item. The form of the *custom-values-list* value 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.

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* value is

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

<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* value.

Default: REPLACE (if a values list is specified with no operation keyword)

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" "Virginia")
```

```

                                ("MD" "Maryland")
                                )
value_generation=remove("CA" "OR" "WA");

```

UPDATE DATASOURCE Statement

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

Syntax

```
UPDATE DATASOURCE "data-source-ID" <options>;
```

Summary of Optional Arguments

DESCRIPTION="*descriptive-text*"

specifies the description of the data source.

ID="*data-source-ID*"

specifies the ID of the data source to update.

NAME="*data-source-name*"

specifies a new name for the data source.

REQUIRED_DATASOURCE=YES | NO

specifies whether the data source is added to the list of required data sources for the information map.

Required Argument

"data-source-ID"

specifies the ID of the data source to update.

Optional Arguments

DESCRIPTION="*descriptive-text*"

specifies the description of the data source.

Alias: DESC=

ID="*data-source-ID*"

specifies the ID of the data source to update.

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.

REQUIRED_DATASOURCE=YES | NO

specifies whether the data source is added to the list of required data sources for the information map. By default, the data source and its assigned filters are included in a query only when a data item that references the data source is explicitly selected for the query. Specify YES if you want the data source and any associated assigned filters to be used in every query that is generated from the information map.

Default: NO

Note: After the data source is updated, you can use the

REQUIRED_DATASOURCES= option in the UPDATE CURRENT_MAP or UPDATE INFOMAP statements to control whether it is required.

Example

```
update datasource "Customer"
  name="Customer_US"
  description="Customers from the US"
  required_datasource=no;
```

UPDATE FILTER Statement

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

Syntax

```
UPDATE FILTER "filter-ID" <options>;
```

Summary of Optional Arguments

CONDITION="*conditional-expression*"

specifies a conditional expression that is used to filter the data.

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-list*)

specifies how custom properties for the filter are updated.

DESCRIPTION="*descriptive-text*"

specifies the description of the filter.

HIDDEN=YES | NO

specifies whether to hide the filter from users of the information map.

ID="*filter-ID*"

specifies the ID of the filter to update.

NAME="*filter-name*"

specifies the name assigned to the filter in the information map.

Required Argument

"*filter-ID*"

specifies the ID of the filter to update.

Optional Arguments

CONDITION="*conditional-expression*"

specifies a conditional expression that is used to filter the data.

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-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* value 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.

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

Requirement: Property names must be unique. If a specified 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.

property-value

specifies the value of the property.

description

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

REMOVE (*property-names-list*)

removes the specified custom properties from the filter.

The form of the *property-names-list* value is

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

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

<REPLACE> (*custom-properties-list*)

replaces the current custom properties for the filter with the specified properties.

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

Default: REPLACE (if a custom properties list is specified with no operation keyword)

DESCRIPTION="*descriptive-text*"

specifies the description of the filter.

Alias: DESC=

HIDDEN=YES | NO

specifies whether to hide the filter from users of the information map. By default, the filter is available to users of the information map. Specify HIDDEN=YES if you want to hide the filter from users (for example, when the filter is used as an assigned filter for a data source and therefore should not be applied again).

Default: NO

ID="*filter-ID*"

specifies the ID for the filter to update.

NAME="*filter-name*"

specifies the name assigned to the filter in the information map.

Example

```
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.

Syntax

UPDATE FOLDER "*folder-name*" <*options*>;

Summary of Optional Arguments

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-list*)
 specifies how custom properties for the folder are updated.

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

LOCATION="*current-parent-folder-name*"

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

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

Required Argument

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

Optional Arguments

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-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* value 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.

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

Requirement: Property names must be unique. If a specified 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.

property-value
specifies the value of the property.

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

REMOVE (*property-names-list*)
removes the specified custom properties from the folder.

The form of the *property-names-list* value is

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

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

<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* value.

Default: REPLACE (if a custom properties list is specified with no operation keyword)

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.

Alias: DESC=

LOCATION=" *current-parent-folder-name*"

LOCATION=" *current-parent-folder-path*"

specifies the current parent folder of the folder that you are updating. The following rules apply:

- 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.

Syntax

UPDATE INFOMAP "*information-map-name*" <*options*>;

Summary of Optional Arguments

CREATE_TARGET_FOLDER=YES | NO

specifies whether to automatically create a folder when inserting all items from a data source.

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-list*)

specifies how custom properties for an information map are updated.

DESCRIPTION="*descriptive-text*"

specifies the description of an information map.

INIT_CAP=YES | NO

specifies whether to capitalize the first letter of each word in the data item name.

JOIN_MODEL=BASIC | ADVANCED

specifies the join strategy that is used during query generation.

MAPPATH="*location*"

specifies the location within the SAS folders tree for the information map to open or update.

REPLACE_UNDERSCORES=YES | NO

specifies whether to replace each underscore (`_`) character in the data item name with a blank space.

REQUIRED_DATASOURCES= `_ALL_` | NONE | ADD (*data-source-list*) | <REPLACE> (*data-source-list*) | REMOVE (*data-source-list*)

specifies how the list of required data sources for the information map is updated.

USE_LABELS=YES | NO

specifies whether to create the data item name using the column label (if available) instead of the column name.

VERIFY=YES | NO

specifies whether the INFOMAPS procedure verifies the validity of data items, filters, and relationships during the Update operation.

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.

Optional Arguments

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*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-list*)

specifies how custom properties for an information map 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 an information map, if there are any.

ADD (*custom-properties-list*)

adds the specified custom properties to an information map.

The form of the *custom-properties-list* value 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.

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

Requirement: Property names must be unique. If a specified property name already exists in an 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.

property-value

specifies the value of the property.

description

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

REMOVE (*property-names-list*)

removes the specified custom properties from an information map.

The form of the *property-names-list* value is

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

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

<REPLACE> (*custom-properties-list*)

replaces the current custom properties for an information map with the specified properties.

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

Default: REPLACE (if a custom properties list is specified with no operation keyword)

DESCRIPTION="descriptive-text"

specifies the description of an information map.

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 INSERT DATASOURCE statements with either the `_ALL_` or the `COLUMNS=` option specified or INSERT DATAITEM statements 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.

JOIN_MODEL=BASIC | ADVANCED

specifies the join strategy that is used during query generation. By default, the INFOMAPS procedure applies a basic join strategy that is appropriate when the information map is based on a simple star schema (a single fact table with multiple dimensions). Specify JOIN_MODEL=ADVANCED if your information map contains more than one measure data item that is derived from more than one data source (multiple fact tables). The advanced join model prevents inflated query results when there is a MANY-to-MANY cardinality between selected measures that exist in two or more tables.

Default: BASIC

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

Default: YES

REQUIRED_DATASOURCES= _ALL_ | NONE | ADD (data-source-list) | <REPLACE> (data-source-list) | REMOVE (data-source-list)

specifies how the list of required data sources for the information map is updated. If you want a data source and any associated assigned filters to be used in every query that is generated from an information map, then designate the data source as required for the information map. If you do not designate a data source as required, then the data source and its assigned filters are included in a query only when a data item that references the data source is explicitly selected for the query.

Valid operations are the following:

`_ALL_`

specifies that all data sources in the information map are required data sources.

NONE

removes the entire list of data sources, if any, that were previously assigned to the information map.

ADD (data-source-list)

adds one or more data sources to the list of required data sources for the information map. The data-source-list value has the following form:

("data-source-ID-1" <... "data-source-ID-n" >)

REMOVE (data-source-list)

removes one or more data sources from the list of required data sources for the information map.

See the ADD operation for a description of the form of the *data-source-list* value.

<REPLACE> (*data-source-list*)

replaces the existing list of required data sources for the information map, if any, with the specified list of required data sources.

See the ADD operation for a description of the form of the *data-source-list* value.

Note: Using the REPLACE operation changes the processing order of the required data sources.

Default: REPLACE (if a data sources list is specified with no operation keyword)

Note: You can use the REQUIRED_DATASOURCE=YES option in the INSERT DATASOURCE and UPDATE DATASOURCE statements to designate a data source as required for the information map.

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 INSERT DATASOURCE statements with either the *_ALL_* or the COLUMNS= option specified or INSERT DATAITEM statements 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 MAP_PERMISSIONS Statement

Changes the access permissions to the information map and optionally assigns authorization-based prefilters for a specific user or group.

Note: The UPDATE MAP_PERMISSIONS statement immediately updates the permissions settings for the information map when it is submitted. This is different from the other UPDATE statements in the INFOMAPS procedure, for which the specified changes are not stored until the information map is saved.

Syntax

- Form 1: **UPDATE MAP_PERMISSIONS** GROUP="*identity*" *permission-specification-1* <...*permission-specification-n*>;
- Form 2: **UPDATE MAP_PERMISSIONS** USER="*identity*" *permission-specification-1* <...*permission-specification-n*>;

Required Arguments

GROUP="*identity*"

USER="*identity*"

specifies the name of a group or user whose permissions to the information map are updated.

permission-specification

specifies the user's or group's access permissions settings to the information map. The specification value has the following forms:

DENY (READ | READMETADATA | WRITEMETADATA)

denies the user or group one or more of the following access permissions to the information map.

READ sets the Read access permission for the information map.

READMETADATA sets the ReadMetadata access permission for the information map.

WRITEMETADATA sets the WriteMetadata access permission for the information map.

GRANT (READ</CONDITION=(*<data-source-filters-1*<... *data-source-filters-n*>>)| READMETADATA | WRITEMETADATA)

grants the user or group one or more access permissions to the information map. See the DENY option for descriptions of the permission values.

When the READ option is specified, you can add the /CONDITION= option to assign authorization-based prefilters for the user or group. The *data-source-filters* values for the /CONDITION= option have the following form:

"*data-source-ID*" (PREFILTERS=("filter-ID-1<... "filter-ID-n">)

data-source-ID

specifies the identifier of the data source to which the assigned filters are applied.

filter-ID

specifies the identifier of a filter defined in the information map that is applied before the specified data source is used.

Notes:

When the /CONDITION= option is used with the READ option, the Read permission for the user or group is granted immediately, but the specified filter assignments are not written to the information map until it is saved.

When a Read permission is denied or removed for the user or group, any associated permission condition is deleted.

REMOVE (READ | READMETADATA | WRITEMETADATA)

removes one or more access permissions to the information map for the specified user or group. See the DENY option for descriptions of the permission values.

See: For more information about metadata permissions, see “Introduction to Access Management” in *SAS Management Console: Guide to Users and Permissions*.

Details

Authorization-based assigned prefilters are filters that are applied to queries by the specified user or group that reference the data source with which the filter is associated and to queries for which the associated data source is marked as required. Prefilters are applied before the data source is used by the specified user or group. These filters subset the data in its associated data source before any other part of a query is run. The generated query contains a subquery that ensures that no data is consumed without the filter being applied.

Note: Assigned prefilters can also be applied for all users of the information map. For more information about assigning prefilters that apply to all users, see [SET ASSIGNED_FILTERS Statement on page 51](#).

Example

```

/* For user "jsmith", grant both WRITEMETADATA and READ permissions */
/* and assign authorization-based prefilters to the identity.      */
update map_permissions
    user="jsmith"
    grant(writemetadata
        read/condition=("Customer"
            (prefilters=("ageLessThan30" "FemaleOnly"))));

/* For group "PUBLIC", deny WRITEMETADATA permission and */
/* grant READMETADATA permission to the information map */
update map_permissions
    group="PUBLIC"
    deny (writemetadata)
    grant(readmetadata);

/* Remove the READ permission of the identity "jsmith". Any */
/* permission condition associated with the READ permission */
/* is deleted and the identity resumes whatever permissions */
/* it obtains from group memberships or inheritance.      */
update map_permissions user="jsmith" remove(read);

```

Note: The specified identity must be a registered user or group on the metadata server.

UPDATE RELATIONSHIP Statement

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

Syntax

UPDATE RELATIONSHIP "*relationship-ID*" <*options*>;

Summary of Optional Arguments

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.

CUSTOM_PROPERTIES=NONE | ADD (*custom-properties-list*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-list*)

specifies how custom properties for the relationship are updated.

DESCRIPTION="*descriptive-text*"

specifies the description of the relationship.

ID="*relationship-ID*"

specifies the ID of the relationship to update.

JOIN=INNER | LEFT | RIGHT | FULL

specifies the type of join.

Required Argument

"*relationship-ID*"

specifies the identifier of the relationship to update.

Optional Arguments

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*) | REMOVE (*property-names-list*) | <REPLACE> (*custom-properties-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 the specified custom properties to the relationship.

The form of the *custom-properties-list* value 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.

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

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

property-value

specifies the value of the property.

description

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

REMOVE (*property-names-list*)

removes the specified custom properties from the relationship.

The form of the *property-names-list* value is

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

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

<REPLACE> (*custom-properties-list*)

replaces the current custom properties for the relationship with the specified properties.

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

Default: REPLACE (if a custom properties list is specified with no operation keyword)

DESCRIPTION="*descriptive-text*"

specifies the description of the relationship.

Alias: DESC=

ID="*relationship-ID*"

specifies the ID of the relationship to update.

JOIN=INNER | LEFT | RIGHT | FULL

specifies the type of join. The type can be one of the following:

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

Examples

Example 1

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

Example 2

```
update relationship "CustTransaction"
  cardinality=one_to_one
  join=left;
```

Example 3

```
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="your-user-ID"
  metapass="your-password"
  metaserver="your-server-name"
  metaport=8561;

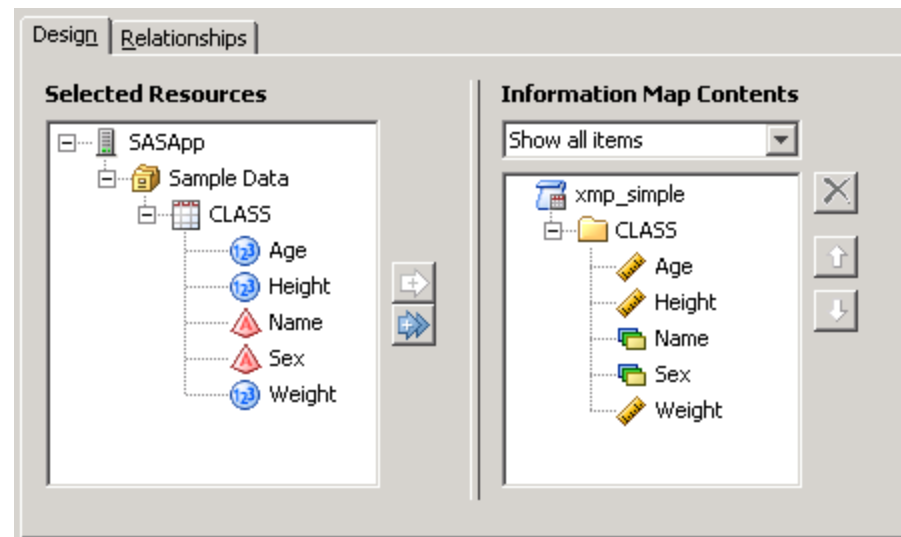
  /* Open a new information map. The specified location is */
  /* where, by default, the information map is saved when a */
  /* SAVE statement issued. The information map exists only */
  /* in memory until a SAVE statement is issued.          */
  new infomap "xmp_simple"
    mappath="/Users/sasdemo/My 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="your-user-ID"
  metapass="your-password"
  metaserver="your-server-name"
  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="Maximum Salary"
  aggregations_keep_list=("MAX")
  format="dollar12.2";
```

```

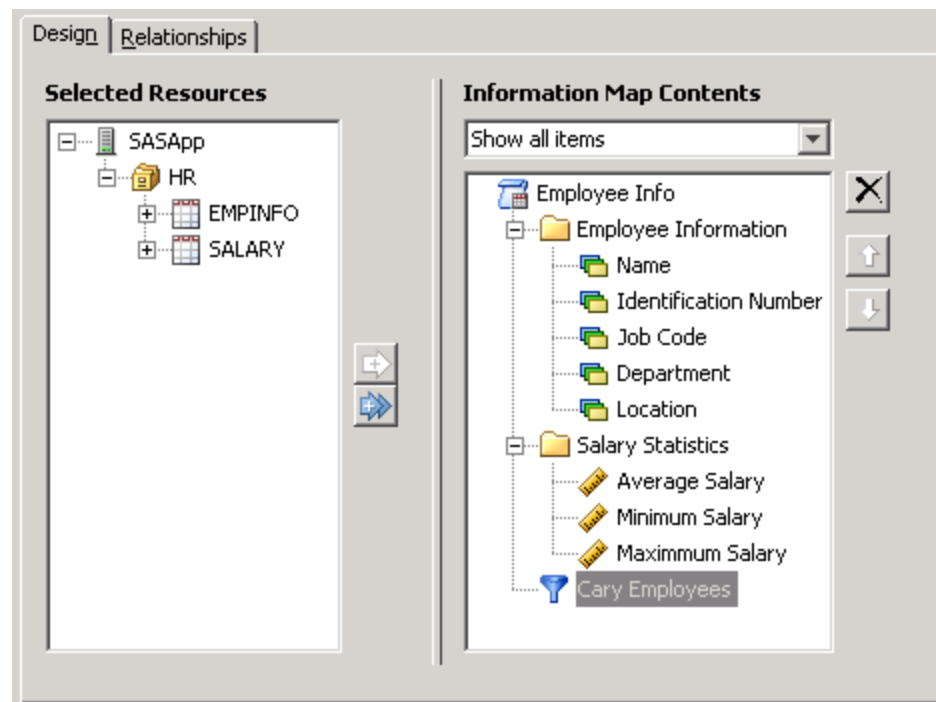
/* Create a filter for the Location data item. */
insert filter
  name="Cary Employees"
  description="Employees who work in Cary, NC"
  condition="<<EmployeeInfo.LOCATION>>='Cary'";

/* 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="your-user-ID"
  metapass="your-password"
  metaserver="your-server-name"
  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.

The screenshot displays the SAS Information Map Studio interface. The 'Physical Data' pane shows the 'CUSTOMER_DIM' table. The 'Information Map' pane shows an 'Average age' expression. A 'Show Generated Query' dialog box is open, showing the following SQL code:

```

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

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

```

The 'AVG((table0.Customer_Age))' expression in the SQL code is circled in blue. A blue arrow points from the text 'Compute average age' to this expression. Another blue arrow points from the 'Results' window to the value '44.000778176' in the table below.

Average age	
1	44.000778176

A 'Show Query' button is visible at the bottom of the Results window.

Chapter 3

Using the SAS Information Maps LIBNAME Engine

What Does the Information Maps Engine Do?	81
Understanding How the Information Maps Engine Works	81
Advantages of Using the Information Maps Engine	85
What Is Required to Use the Information Maps Engine?	85
What Is Supported?	85

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 only reads 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, “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. (The Information Maps engine is read-only.) 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.

Note: The Information Maps engine honors the permission settings defined in the metadata for the information map and its data sources. A user is not allowed to access data via the Information Maps engine if the user's Read permission for the information map or its data sources in the metadata server is DENY.

The following example shows a LIBNAME statement for the Information Maps engine and the output that 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";
```

Log 3.1 Log for 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;

```

Log 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

      #   Name                                     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 3.1 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 Number	Num	8	SSN11.	Identification 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 Employed	Num	8	COMMA6.	The number of years that the employee 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.				

Obs	Name	Identification 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	Winfrey, Ambrose Y.	000-00-1579	CCD001	CCD	Cary
5	Blue, Kenneth N.	000-00-1637	MIS004	QA0	Cary

Obs	Salary_2	Salary2	Salary3	Salary4	Hire Date	Number of Years 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

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

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 only reads 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 are returned, especially from a remote data source, for processing 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 97. For additional information about improving the performance of the Information Maps engine, see [Chapter 6, “Hints and Tips for Using the INFOMAPS Procedure or the Information Maps Engine,”](#) on page 101.

- 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	87
Dictionary	87
LIBNAME Statement	87
Examples	93
Example 1: Submitting a LIBNAME Statement Using the Defaults	93
Example 2: Submitting a LIBNAME Statement Using Connection Options	93

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*.

Dictionary

LIBNAME Statement

Associates a SAS libref with information maps.

Syntax

```
LIBNAME libref INFOMAPS MAPPATH="location" <options>
```

Summary of Optional Arguments

Options for Connecting to the SAS Metadata Server

DOMAIN="authentication-domain"

specifies an authentication domain in the metadata server that is associated with the user ID and password.

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.

METAPASS="password"

specifies the password that corresponds to the user ID that connects to the metadata server.

METAPORT=port-number

specifies the TCP port that the metadata server is listening to for connections.

METASERVER="address"

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

METAUSER="user-ID"

specifies the user ID to connect to the metadata server.

SSPI=YES | NO

specifies whether Integrated Windows Authentication is used.

Other Options for the Information Maps Engine

AGGREGATE=YES | NO

specifies whether detailed data or aggregated data is retrieved from the data source.

EXPCOLUMNLEN=integer

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

PRESERVE_MAP_NAMES=YES | NO

specifies how information map names are handled.

READBUFF=integer

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

SPOOL=YES | NO

specifies whether a spool file is created.

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: Concepts*.

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=

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=

Default: DefaultAuth

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. 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*.

Specifying a plain-text password can compromise security. You can use the PWENCODE procedure to encode the password value. For information about the PWENCODE procedure, see *Base SAS Procedures Guide*.

Alias:

PASSWORD=

PW=

Note: If the password is not encoded or does not contain a blank space (or spaces), then enclosing the identifier in quotation marks is optional.

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

Alias:

SERVER=

HOST=

IPADDR=

Note: Enclosing the identifier in quotation marks is optional.

Example:

```
metaserver="myip.mycompany.com"
```

METAUSER="*user-ID*"

specifies the user ID to connect to the metadata server. 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=

Restrictions:

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.

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"**.

Note: If the user ID does not contain a blank space (or spaces) or a backslash character, enclosing the identifier in quotation mark is optional.

Examples:

```
metauser="My UserID"
```

```
metauser=myUserID
```

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 the *SAS Intelligence Platform: Security Administration Guide*.

Default: NO

Other 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.

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.

Default: NO

Note: The Information Maps engine also supports an AGGREGATE= data set option that you can use 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 reverts to the setting specified when the libref was created. For more information, see "[AGGREGATE= Data Set Option](#)" on page 95.

EXPCOLUMNLEN=*integer*

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

Default: 32

Note: The Information Maps engine also supports an EXPCOLUMNLEN= data set option that you can use 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 reverts to the setting specified when the libref was created. For more information, see "[EXPCOLUMNLEN= Data Set Option](#)" on page 96.

PRESERVE_MAP_NAMES=YES | NO

specifies how information map names are handled.

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 must 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 *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. 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.

Alias: BUFFSIZE=

Default: 1000

Restriction: The specified value must be a positive integer.

Note: The Information Maps engine also supports a READBUFF= data set option that you can use 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 "[READBUFF= Data Set Option](#)" on page 98.

SPOOL=YES | NO

specifies whether a spool file is created.

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

Examples

Example 1: Submitting a LIBNAME Statement Using the Defaults

The following 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";
```

Log 4.1 Log for 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

The following example shows 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";
```

Log 4.2 Log for 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 5

SAS Data Set Options for the Information Maps Engine

Using Data Set Options	95
Dictionary	95
AGGREGATE= Data Set Option	95
EXPCOLUMNLEN= Data Set Option	96
FILTER= Data Set Option	97
READBUFF= Data Set Option	98

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

For more information about SAS data set options, see *SAS Data Set Options: Reference*.

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.

Dictionary

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.

See Also

[AGGREGATE=](#) option in the LIBNAME statement for the Information Maps engine

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
Default:	32
Restriction:	Use with character column only.

Syntax

EXPCOLUMNLEN=*column-length*

Required Argument

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 the LIBNAME statement for the Information Maps engine

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
Restrictions:	Use only with information maps that contain filters. Filters that prompt for values at run time are not supported.
Requirement:	If you specify more than a single filter, then parentheses are required.

Syntax

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

Required Argument

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 System Options: Reference*. You can use the CONTENTS procedure to view the assigned filter names.

Optional Arguments

NOT

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.

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

Boolean-operator

combines the effects of two filters or filter clauses.

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

OR 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, LIBNAME Statement
Category:	Data Set Control
Alias:	BUFSIZE=
Default:	1000

Syntax

READBUFF=integer

Required Argument

integer

specifies the number of rows to hold in memory input into SAS.

Restriction: The value must be a positive integer.

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 the LIBNAME statement for the Information Maps engine

Chapter 6

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

Hints and Tips for Using the INFOMAPS Procedure	101
Hints and Tips for Using the Information Maps Engine	102
Improving the Performance of the Information Maps Engine	102
Creating Information Maps That Work Well with the Information Maps Engine .	102

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 17](#).
- 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 in 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 that 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 than 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 is 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 is 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 Options for the Information Maps Engine” on page 91](#).
- 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 System Options: Reference*.
- 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 7

Example: Using the INFOMAPS Procedure and the Information Maps Engine

About This Example	105
Step 1: Create a Library Definition in the SAS Metadata Server	106
Step 2: Set the Metadata System Options and a Macro Variable	106
Step 3: Register Data Using the METALIB Procedure	107
Step 4: Create an Information Map Using the INFOMAPS Procedure	108
Step 5: Retrieve the Data Associated with the Information Map Using the Information Maps Engine	115
Step 6: View the Data Items and Filters Using the CONTENTS Procedure	115
Step 7: Print the Data from the Information Map	117
Step 8: Analyze the Data in SAS and Produce an ODS Report	119

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\9.3\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 SAS Sample Data 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.

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

The following code defines a macro variable for the information map location and sets system options for the connection to the server:

```
/* Use traditional listing output. */
ods html close;
ods listing;

/* Assign a macro variable to store the path to the folder */
/* that contains information maps (to avoid having to set */
/* the path multiple times). */
%LET infomap_path=/Shared Data/Infomap Example;

/* Set system options for connecting to the metadata server. */
options metauser="your-user-ID"
        metapass="your-password"
        metaserver="your-server-name"
        metaport=8561;
```

Log 7.1 Setting the Metadata System Options and the Macro Variable

```

1  /* Use traditional listing output. */
2  ods html close;
3  ods listing;
4
5  /* Assign a macro variable to store the path to the folder */
6  /* that contains information maps (to avoid having to set */
7  /* the path multiple times). */
8  %LET infomap_path=/Shared Data/Infomap Example;
9
10 /* Set system options for connecting to the metadata server. */
11 options metauser="myUserID"
12         metapass=XXXXXXXX
13         metaserver="myserver.mycompany.com"
14         metaport=8561;

```

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 *SAS Language Interfaces to Metadata*.

The following code registers the tables using the METALIB procedure:

```

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

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

```

Log 7.2 Log for the METALIB Procedure

```

16 /* Use the library object defined in the SAS Metadata Repository */
17 /* to obtain all accessible table metadata from the data source */
18 /* to create table metadata in the metadata repository. */
19 proc metalib;
20     omr (library="SAS Sample Data");
21     select("empinfo" "jobcodes" "salary");
22
23     /* Create a summary report of the metadata changes. */
24     report;
25 run;

```

NOTE: A total of 3 tables were analyzed for library "SAS Sample Data".
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	0.30 seconds
cpu time	0.20 seconds

Output 7.1 Output from the METALIB Procedure

The METALIB Procedure		
Summary Report for Library SAS Sample Data 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	A5Y81OV0.BH000028	empinfo
JOBCODES	A5Y81OV0.BH000029	jobcodes
SALARY	A5Y81OV0.BH00002A	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="SASApp"
    table="SAS Sample Data".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="SASApp"
    table="SAS Sample Data".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="SASApp"
    table="SAS Sample Data".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.

Log 7.3 Log for the INFOMAPS Procedure

```

27 /* Create a new information map using the INFOMAPS procedure. */
28 proc infomaps mappath="&infomap_path";
29 /* Open a new information map. */
30 new infomap "Employee Info"
31     auto_replace=yes;
32
33 /* Insert a data source and three data items using the COLUMNS= option. */
34 insert datasource sasserver="SASApp"
35     table="SAS Sample Data".empinfo
36     columns=("JobCode" "LOCATION" "DIVISION")
37     id="Empinfo";
NOTE: A data item was inserted for the physical column "EMPINFO.JOBCODE". Its
      ID is "Jobcode".
NOTE: A data item was inserted for the physical column "EMPINFO.LOCATION". Its
      ID is "Location".
NOTE: A data item was inserted for the physical column "EMPINFO.DIVISION". Its
      ID is "Division".

```

```

38
39 /* Insert a data item based on a physical column. Because the ID= option */
40 /* is not specified, a note with its ID value will print in the SAS log. */
41 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".

42
43 /* Insert a data item with an expression. */
44 insert dataitem expression="SUBSTRN(<<root.Jobcode>>, 1, 3)"
45       type=character
46       name="Department Code"
47       id="Dept_code";
48
49 /* Insert a second data source, plus a data item into the */
50 /* current information map. */
51 insert datasource sasserver="SASApp"
52       table="SAS Sample Data".jobcodes
53       columns=( "TITLE" )
54       id="Jobcodes";
NOTE: A data item was inserted for the physical column "JOBCODES.TITLE". Its ID is
      "Title".

55
56 /* Change the data item to a measure so that you can use it in computations */
57 /* and analytical expressions. Set the default aggregation to Count. */
58 update dataitem "Title" aggregation=COUNT classification=MEASURE;
59
60 /* Insert a third data source into the current information map. */
61 insert datasource sasserver="SASApp"
62       table="SAS Sample Data".salary
63       id="Salary";
64
65 /* Add joins between the tables. */
66 insert relationship
67       left_table="Empinfo"
68       right_table="Jobcodes"
69       join=inner
70       condition="( <<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>> )";
NOTE: A relationship between the data sources "Empinfo" and "Jobcodes" has been
      inserted. The relationship's ID is "JOIN_10".

71
72 insert relationship
73       left_table="Empinfo"
74       right_table="Salary"
75       join=inner
76       condition="( <<Empinfo.IDNUM>>=<<Salary.IDNUM>> )";
NOTE: A relationship between the data sources "Empinfo" and "Salary" has been
      inserted. The relationship's ID is "JOIN_11".

77
78 /* Insert a folder and additional business items. */
79 insert folder "Salary Info";
80
81 insert dataitem column="Salary".salary
82       name="Annual Salary" folder="Salary Info";
83
84 /* Insert a data item that contains an expression */
85 insert dataitem expression="<<Salary.SALARY>>/12" type=numeric
86       name="Monthly Salary" folder="Salary Info";
87

```

```
88 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".
```

```
89
90 /* Insert filters. */
91 insert filter "Status is Current"
92     condition="<<root.Enddate>> IS NULL" folder="Salary Info";
93
94 insert filter "Education and Publications"
95     condition='SUBSTRN(<<root.Jobcode>>, 1, 3) IN ("EDU","PUB")'
96     desc="Employees in Education and Publications";
97
98 insert filter "Host Systems Development"
99     condition='<<root.Division>>="HOST SYSTEMS DEVELOPMENT" '
100    desc="Employees in Host Systems Development";
101
102 insert filter "Cary HQ"
103     condition='<<root.Location>>="Cary" '
104     desc="Located in Cary, North Carolina HQ";
105
106 /* List the key properties of business data in the current information map. */
107 list;
```

Total datasources: 3

```
Data source: SAS Sample Data.EMPINFO
ID: Empinfo
Name: EMPINFO
Description:
Required data source: NO
```

```
Data source: SAS Sample Data.JOBCODES
ID: Jobcodes
Name: JOBCODES
Description:
Required data source: NO
```

```
Data source: SAS Sample Data.SALARY
ID: Salary
Name: SALARY
Description:
Required data source: NO
```

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 data source: SAS Sample Data.EMPINFO
Right data source: SAS Sample Data.JOBCODES
Cardinality: UNKNOWN
Join type: INNER
Join expression: (<<Empinfo.JOBCODE>>=<<Jobcodes.JOBCODE>>)

Relationship ID: JOIN_11
Left data source: SAS Sample Data.EMPINFO
Right data source: SAS Sample Data.SALARY
Cardinality: UNKNOWN
Join type: INNER
Join expression: (<<Empinfo.Identification Number>>=<<Salary.Identification Number>>)

```

108

109 /* Save the information map. */

110 save;

NOTE: The information map "Employee Info" has been saved in the folder
"/Shared Data/Infomap Example".

111

112 /* End the INFOMAPS procedure. */

113 quit;

NOTE: PROCEDURE INFOMAPS used (Total process time):

real time	10.13 seconds
cpu time	0.04 seconds

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:

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

/* Use the Information Maps engine to define a libref that retrieves data */
/* from the 'Employee Info' information map created with PROC INFOMAPS. */
libname HR_Data infomaps mappath="&infomap_path";
```

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.

Log 7.4 Log for the Information Maps Engine LIBNAME Statement

```
115 /* Allow mixed-case letters and blank spaces in information map names. */
116 option validvarname=any;
117
118 /* Use the Information Maps engine to define a libref that retrieves data */
119 /* from the 'Employee Info' information map created with PROC INFOMAPS. */
120 libname HR_Data infomaps mappath="&infomap_path";
NOTE: Libref HR_DATA was successfully assigned as follows:
      Engine:          INFOMAPS
      Physical Name:  /Shared Data/Infomap Example
```

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;
```

Log 7.5 Log for the CONTENTS Procedure

```

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

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time          0.46 seconds
      cpu time           0.03 seconds
    
```

Output 7.2 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;
    
```

Log 7.6 Log for the CONTENTS Procedure

```

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

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time          0.22 seconds
      cpu time           0.01 seconds

```

Output 7.3 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						
#	Variable	Type	Len	Format	Default 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;

/* Use aggregation and a filter to produce a report of total */
/* salary for selected divisions.                             */
title1 "Total salary for each division except PUBS and EDU";
title2 "and the Host Systems division";
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;

```

Log 7.7 Log for the PRINT Procedure

```

130 /* Print five observations of detailed data from the data that */
131 /* the information map references.                               */
132 proc print data=HR_Data."Employee Info"n (obs=5);
133 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          0.44 seconds
      cpu time           0.07 seconds

135
136 /* Use aggregation and a filter to produce a report of total */
137 /* salary for selected divisions.                             */
138 title1 "Total salary for each division except PUBS and EDU";
139 title2 "and the Host Systems division";
140 proc print data=HR_Data."Employee Info"n
141      (keep="Annual Salary"n Division
142      aggregate=yes
143      filter=(NOT("Education and Publications"n
144                  OR "Host Systems Development"n)) );
145 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          0.42 seconds
      cpu time           0.06 seconds

```

Output 7.4 Output from the PRINT Procedure

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 PUBS and EDU
and the Host Systems division

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 is produced with ODS (Output Delivery System).

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

```

/* Use a filter to create a data set that contains only data */
/* for the Host Systems Development division.                */
data work.HRinfo;
set HR_Data."Employee Info"(filter="Host Systems Development");
keep jobcode "Annual Salary";

```

```

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 in Host Systems Development Division";
run;

/* Close ODS output. */
ods html close;

```

Log 7.8 Log for the DATA Step and the MEANS Procedure

```

147 /* Use a filter to create a data set that contains only data */
148 /* for the Host Systems Development division. */
149 data work.HRinfo;
150 set HR_Data."Employee Info"n(filter="Host Systems Development"n);
151 keep jobcode "Annual Salary"n;
152 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          0.39 seconds
      cpu time           0.07 seconds

153
154 /* Produce an ODS report. */
155 ods html body="example_body.htm";
NOTE: Writing HTML Body file: example_body.htm
156
157 /* Analyze the annual salary distribution data. */
158 proc means data=work.HRinfo maxdec=0;
159     var "Annual Salary"n;
160     class jobcode;
161     title "Annual Salary by Job Code in Host Systems Development
Division"
161! ;
162 run;

NOTE: There were 21 observations read from the data set WORK.HRINFO.
NOTE: PROCEDURE MEANS used (Total process time):
      real time          0.02 seconds
      cpu time           0.03 seconds

163
164 /* Close ODS output. */
165 ods html close;

```

Output 7.5 Output from the MEANS Procedure

```

Annual Salary by Job Code in Host Systems Development Division

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:

Output 7.6 Report Displayed in the Results Viewer

Annual Salary by Job Code in Host Systems Development Division						
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

Appendix 1

SQL Dictionary Tables for the Information Maps Engine

Using SQL DICTONARY Tables	123
DICTIONARY.INFOMAPS Table	123
DICTIONARY.DATAITEMS Table	124
DICTIONARY.FILTERS Table	125

Using SQL DICTONARY Tables

An SQL DICTONARY 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 DICTONARY 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 DICTONARY.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 DICTONARY.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 DICTONARY.INFOMAPS as i;
```

Output A1.1 Output from DICTONARY.INFOMAPS Table Query

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

DICTONARY.DATAITEMS Table

The SQL DICTONARY.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 identifier

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	Identification	CATEGORY
	Number	Number	
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 identifier

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

SAS System Options for Diagnostic Messages	127
Dictionary	127
SASTRACE	127
SASTRACELOC	128
NOSTSUFFIX	128
Example: Displaying Detailed and Summary Timing Information	128

SAS System Options for Diagnostic Messages

You can use the following SAS system options to control whether and how diagnostic messages are issued by the Information Maps Engine. For more information about SAS system options, see *SAS System Options: Reference*.

Dictionary

SASTRACE

Traces diagnostic messages issued by the Information Maps engine.

Syntax

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

Optional Arguments

'>>>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.

SASTRACELOC

Specifies the destination for diagnostic messages.

Syntax

SASTRACELOC=stdout | SASLOG | FILE *'path-name'*

NOSTSUFFIX

Simplifies diagnostic messages by suppressing some nonessential output.

Syntax

NOSTSUFFIX

Example: Displaying Detailed and Summary Timing Information

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

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

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.

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

an application that requests either resources or services from a server, possibly over a network.

column

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

cube

See OLAP cube.

data element

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

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 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, or the end of the job. The DATA step enables you to read raw data or other SAS data sets and to create SAS data sets.

data view

See SAS data view.

dimension

a data element that categorizes values in a data set into non-overlapping categories that can be used to group, filter, and label the data in meaningful ways. 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, and Date, and (2) Year, Week, and Day.

engine

a component of SAS software that reads from or writes to a file. Various engines enable SAS to access different types of file formats.

Extensible Markup Language

See XML.

filter

See package filter.

format

See SAS format.

global 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.

hierarchy

an arrangement of related objects into levels that are based on parent-child relationships. Members of a hierarchy are arranged from more general to more specific.

informat

See SAS informat.

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.

join

an operation that combines data from two or more tables. A join is typically created by means of SQL (Structured Query Language) code or a 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.

library reference

See libref.

libref

a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library.

literal

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

MDX language

See multidimensional expressions language.

measure

a member of a Measures dimension.

measure data item

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.

member

an element of a dimension. For example, for a dimension that contains time periods, each time period is a member of the dimension.

metadata

descriptive data about data that is stored and managed in a database, in order to facilitate access to captured and archived data for further use.

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 stores information about servers, users, and stored processes and that provides this information to one or more client applications.

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).

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 cube

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

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.

package filter

specified criteria that are applied to data in order to identify the subset of data for a subsequent operation, such as continued processing.

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.

PROC

See SAS procedure.

procedure

See SAS procedure.

prompted filter

a type of filter that is associated with a prompt, and that enables the user to specify filtering criteria when a query is executed.

query

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

register

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

relationship

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

result set

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

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.

SAS data set

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

SAS data set option

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

SAS data view

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

SAS format

a type of SAS language element that applies a pattern to or executes instructions for a data value to be displayed or written as output. Types of formats correspond to the data's type: numeric, character, date, time, or timestamp. The ability to create user-defined formats is also supported. Examples of SAS formats are BINARY and DATE.

SAS informat

a type of SAS language element that applies a pattern to or executes instructions for a data value to be read as input. Types of informats correspond to the data's type: numeric, character, date, time, or timestamp. The ability to create user-defined informats is also supported. Examples of SAS informats are BINARY and DATE.

SAS library

one or more files that are defined, recognized, and accessible by SAS and that are referenced and stored as a unit. Each file is a member of the library.

SAS Metadata Repository

a container for metadata that is managed by the SAS Metadata Server.

SAS Open Metadata Architecture

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

SAS procedure

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

SAS program

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

SAS system option

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

SAS 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 (rows).

schema

a map or model of the overall data structure of a database. A schema consists of schema records that are organized in a hierarchical tree structure. Schema records contain schema items.

server

software that provides either resources or services to requesting clients, possibly over a network.

statement option

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

variable

See SAS variable.

XML

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

Index

Special Characters

ALL argument
 DELETE IDENTITY_PROPERTY
 statement 11
 INSERT DATASOURCE statement 30
 LIST statement 42

A

access permissions
 changing 70
 denying for users or groups 71
 granting for users or groups 71
 ACTIONS= argument
 INSERT DATAITEM statement 20
 UPDATE DATAITEM statement 57
 advanced join strategy 48, 69
 aggregate functions
 assigning to data items 21
 InternalAggregation 21
 InternalAggregationAdditive 21
 removing from data item 24
 table of 21
 AGGREGATE= argument
 LIBNAME statement 91
 AGGREGATE= data set option
 Information Maps engine 95
 aggregated data
 returning with Information Maps engine
 91, 95
 AGGREGATION= argument
 INSERT DATAITEM statement 21
 UPDATE DATAITEM statement 58
 AGGREGATIONS_DROP_LIST=
 argument
 INSERT DATAITEM statement 24
 AGGREGATIONS_KEEP_LIST=
 argument
 INSERT DATAITEM statement 24
 AGGREGATIONS_LIST= argument
 UPDATE DATAITEM statement 58

ALLOW_MAJOR_VERSION_UPGRAD
 E= argument
 SAVE statement 49
 ALLOW_MINOR_VERSION_UPGRAD
 E= argument
 SAVE statement 50
 assigned filters
 authorization based 70
 for all users 51
 AUTO_REPLACE= argument
 NEW INFOMAP statement 47

B

best practices
 INFOMAPS procedure 101
 Information Maps engine 102
 buffer size
 for reading data 92, 98
 BY-group processing 86

C

calculated data items 103
 cardinality
 specifying for relationships 40
 CARDINALITY= argument
 INSERT RELATIONSHIP statement
 40
 UPDATE RELATIONSHIP statement
 73
 CLASSIFICATION= argument
 INSERT DATAITEM statement 24
 UPDATE DATAITEM statement 59
 CLOSE INFOMAP statement
 INFOMAPS procedure 10
 COLUMN= argument
 INSERT DATAITEM statement 18
 columns 96
 length of 91
 COLUMNS= argument

- INSERT DATASOURCE statement 30
 - CONDITION= argument
 - INSERT FILTER statement 33
 - INSERT RELATIONSHIP statement 40
 - UPDATE FILTER statement 63
 - UPDATE RELATIONSHIP statement 73
 - CONTENTS procedure
 - viewing aggregated data items 116
 - viewing data items and filters 82
 - viewing data items with 115
 - viewing filters with 115
 - CREATE_TARGET_FOLDER= argument
 - NEW INFOMAP statement 47
 - UPDATE CURRENT_INFOMAP statement 53
 - UPDATE INFOMAP statement 67
 - CUBE= argument
 - INSERT DATASOURCE statement 29
 - current information map
 - assigning custom properties 53
 - assigning description 54
 - capitalizing first letter of data item names 54
 - creating folder for 53
 - replacing underscores in data item names 55
 - selecting required data sources 55
 - using column labels as data item names 56
 - verifying objects in 56
 - CUSTOM_PROPERTIES argument
 - INSERT FOLDER statement 36
 - CUSTOM_PROPERTIES= argument
 - INSERT DATAITEM statement 25
 - INSERT FILTER statement 33
 - INSERT RELATIONSHIP statement 40
 - NEW INFOMAP statement 47
 - UPDATE CURRENT_INFOMAP statement 53
 - UPDATE DATAITEM statement 59
 - UPDATE FILTER statement 63
 - UPDATE FOLDER statement 65
 - UPDATE INFOMAP statement 68
 - UPDATE RELATIONSHIP statement 73
 - custom properties
 - assigning to current information map 53
 - assigning to data items 25, 59
 - assigning to filters 33, 63
 - assigning to folders 36, 65
 - assigning to information maps 47, 68
 - assigning to relationships 40, 73
- D**
- data item names
 - capitalizing first letter of 47
 - replacing underscores in 48
 - using column labels for 48
 - data items 1, 6
 - aggregate functions 21
 - assigning classification 24
 - assigning custom properties 25, 59
 - assigning descriptions 25, 60
 - assigning formats 26
 - assigning identifiers (IDs) 26
 - assigning names 27, 60
 - assigning types 27, 60
 - assigning value generation methods 27, 61
 - calculated 103
 - capitalizing first letter of names 47
 - inserting 17
 - moving 44
 - removing aggregate functions 24
 - replacing underscores in names 48
 - specifying folder for 25
 - updating in information maps 56
 - using column labels for names 48
 - validating in information maps 48
 - viewing with CONTENTS procedure 115
 - viewing with the CONTENTS procedure 82
 - data reads
 - buffering 92, 98
 - data set options
 - for Information Maps engine 95
 - data sources
 - assigning descriptions 30, 62
 - assigning identifiers (IDs) 31
 - assigning names 31, 62
 - defining OLAP cubes as 29
 - defining relational tables as 29
 - identifying as required 31
 - identifying SAS server for 29
 - identifying table columns 30
 - inserting in information maps 28
 - updating in information maps 62
 - DATAITEMS argument
 - LIST statement 42
 - DATASETS procedure
 - listing available information maps 82
 - DATASOURCES argument
 - LIST statement 42
 - DEFINITION= argument

- SET ASSIGNED_FILTERS statement 51
 - DELETE IDENTITY_PROPERTY statement
 - _ALL_ argument 11
 - ID= argument 11
 - INFOMAPS procedure 11
 - DELETE INFOMAP statement
 - INFOMAPS procedure 11
 - MAPPATH= argument 11
 - deleting SAS identity properties 11
 - DESCRIPTION= argument
 - INSERT DATAITEM statement 25
 - INSERT DATASOURCE statement 30
 - INSERT FILTER statement 33
 - INSERT FOLDER statement 36
 - INSERT RELATIONSHIP statement 40
 - NEW INFOMAP statement 47
 - UPDATE CURRENT_INFOMAP statement 54
 - UPDATE DATAITEM statement 60
 - UPDATE DATASOURCE statement 62
 - UPDATE FILTER statement 64
 - UPDATE FOLDER statement 66
 - UPDATE INFOMAP statement 68
 - UPDATE RELATIONSHIP statement 74
 - descriptions
 - assigning to current information map 54
 - assigning to data items 25, 60
 - assigning to data sources 30, 62
 - assigning to filters 33, 64
 - assigning to folders 36, 66
 - assigning to information maps 47, 68
 - assigning to relationships 40, 74
 - DOMAIN= argument
 - LIBNAME statement 89
 - PROC INFOMAPS statement 8
 - dynamic filters
 - creating 38
- E**
- engine 81
 - engine name
 - Information Maps engine 88
 - ERRORSTOP argument
 - PROC INFOMAPS statement 8
 - examples 105
 - creating information maps with
 - INFOMAPS procedure 108
 - LIBNAME statement 82
 - ODS reports 119
 - printing information maps 117
 - retrieving data 115
 - viewing aggregated data items 116
- EXPCOLUMNLEN= argument
 - LIBNAME statement 91
- EXPCOLUMNLEN= data set option
 - Information Maps engine 96
- EXPORT
 - LOCALIZABLE_PROPERTIES statement
 - FILE= argument 13
 - INFOMAP= argument 13
 - INFOMAPS procedure 13
- EXPORT statement
 - FILE= argument 12
 - INFOMAP argument 12
 - INFOMAPS procedure 12
 - MAPPATH= argument 12
- exporting information maps 12
- exporting localizable properties 13
- EXPRESSION= argument
 - INSERT DATAITEM statement 18
 - UPDATE DATAITEM statement 60
- EXTERNAL_IDENTITY property 37
- external identity
 - SAS identity properties and 39
- F**
- FILE= argument
 - EXPORT
 - LOCALIZABLE_PROPERTIES statement 13
 - EXPORT statement 12
 - IMPORT LOCALIZED_PROPERTIES statement 16
 - IMPORT statement 15
 - FILTER= data set option
 - Information Maps engine 97
 - filters 1, 6
 - assigning custom properties 33, 63
 - assigning descriptions 33, 64
 - assigning identifiers (IDs) 34
 - assigning names 34, 64
 - defining conditional expressions for 33
 - for OLAP data items 20
 - inserting in information maps 32
 - marking as hidden 34
 - moving 44
 - restricting returned data 82
 - specifying folder for 34
 - subsetting information maps with 97
 - updating in information maps 63
 - validating in information maps 48
 - viewing with CONTENTS procedure 115

- viewing with the CONTENTS
 - procedure 82
 - FILTERS argument
 - LIST statement 42
 - FOLDER= argument
 - INSERT DATAITEM statement 25
 - INSERT FILTER statement 34
 - folders 1
 - adding data items to 25
 - adding filters to 34
 - assigning custom properties 36, 65
 - assigning descriptions 36, 66
 - assigning name for 35
 - assigning names 66
 - inserting in information maps 35
 - moving 45
 - specifying parent folder 36
 - updating in information maps 65
 - FORMAT= argument
 - INSERT DATAITEM statement 26
 - UPDATE DATAITEM statement 60
 - formats
 - assigning to data items 26
- H**
- HIDDEN= argument
 - INSERT FILTER statement 34
 - UPDATE FILTER statement 64
 - HIERARCHY= argument
 - INSERT DATAITEM statement 19
- I**
- ID_LIST= argument
 - MOVE DATAITEM statement 44
 - MOVE FILTER statement 44
 - ID= argument
 - DELETE IDENTITY_PROPERTY statement 11
 - INSERT DATAITEM statement 26
 - INSERT DATASOURCE statement 31
 - INSERT FILTER statement 34
 - INSERT IDENTITY_PROPERTY statement 38
 - INSERT RELATIONSHIP statement 40
 - UPDATE DATAITEM statement 60
 - UPDATE DATASOURCE statement 62
 - UPDATE FILTER statement 64
 - UPDATE RELATIONSHIP statement 74
 - identifiers (IDs)
 - assigning to data items 26
 - assigning to data sources 31
 - assigning to filters 34
 - assigning to relationships 40
 - assigning to SAS identity properties 38
 - IDENTITY_GROUP_NAME property 37
 - IDENTITY_GROUPS property 38
 - IDENTITY_NAME property 38
 - IMPORT LOCALIZED_PROPERTIES statement
 - FILE= argument 16
 - INFOMAPS procedure 15
 - LOCALES= argument 16
 - IMPORT statement
 - FILE= argument 15
 - INFOMAPS procedure 14
 - importing information maps 14, 15
 - importing localizable properties files 15
 - INFOMAP argument
 - EXPORT statement 12
 - SAVE statement 50
 - INFOMAP= argument
 - EXPORT
 - LOCALIZABLE_PROPERTIES statement 13
 - INFOMAPS argument
 - LIBNAME statement 88
 - INFOMAPS procedure 6
 - best practices 101
 - CLOSE INFOMAP statement 10
 - creating information maps 108
 - DELETE IDENTITY_PROPERTY statement 11
 - DELETE INFOMAP statement 11
 - EXPORT
 - LOCALIZABLE_PROPERTIES statement 13
 - EXPORT statement 12
 - IMPORT LOCALIZED_PROPERTIES statement 15
 - IMPORT statement 14
 - improving performance 101
 - INSERT DATAITEM statement 17
 - INSERT DATASOURCE statement 28
 - INSERT FILTER statement 32
 - INSERT FOLDER statement 35
 - INSERT IDENTITY_PROPERTY statement 37
 - INSERT RELATIONSHIP statement 39
 - LIST statement 41
 - MOVE DATAITEM statement 44
 - MOVE FILTER statement 44
 - MOVE FOLDER statement 45
 - NEW INFOMAP statement 45
 - PROC INFOMAPS statement 7
 - SAVE statement 49

- SET ASSIGNED_FILTERS statement
 - 51
- SET STORED PROCESS statement 52
- syntax 7
- UPDATE CURRENT_INFOMAP
 - statement 52
- UPDATE DATAITEM statement 56
- UPDATE DATASOURCE statement
 - 62
- UPDATE FILTER statement 63
- UPDATE FOLDER statement 65
- UPDATE INFOMAP statement 66
- UPDATE MAP_PERMISSIONS
 - statement 70
- UPDATE RELATIONSHIP statement
 - 72
- information maps 1
 - allowing upgrade when saving 49
 - assigning custom properties 47, 68
 - assigning descriptions 47, 68
 - associating stores processes 52
 - benefits of 2
 - changing access permissions for 70
 - closing 10
 - creating folder for data items 47
 - creating new information maps 45
 - defining prefilters for 51
 - defining results filters for 51
 - deleting from metadata server 11
 - examples 105
 - examples of creating 75
 - exporting localizable properties 13
 - exporting to XML files 12
 - importing from XML files 14
 - inserting data items 17
 - inserting data sources 28
 - inserting filters 32
 - inserting folders 35
 - inserting relationships 39
 - inserting SAS identity properties 37
 - listing properties of 41
 - moving data items 44
 - moving folders 45
 - naming restrictions 102
 - printing data 117
 - reading with Information Maps engine
 - 115
 - removing access permission 72
 - removing SAS identity properties 11
 - replacing existing information maps 47
 - SAS Information Map Studio and 6
 - saving 49
 - specifying join model 48, 69
 - specifying location for 48
 - updating current information map 52
 - updating existing data items 56
 - updating existing data sources 62
 - updating existing filters 63
 - updating existing folders 65
 - updating existing information maps 66
 - updating existing relationships 72
 - validating objects in 48
- Information Maps engine 81
 - advantages of 85
 - AGGREGATE= data set option 95
 - best practices 102
 - data set options 95
 - engine name 88
 - examples 105
 - EXPCOLUMNLEN= data set option
 - 96
 - FILTER= data set option 97
 - how it works 81
 - improving performance 102
 - memory usage 102
 - printing data 82
 - READBUFF= data set option 98
 - requirements for using 85
 - restricting returned data 82
 - retrieving data 115
 - what is supported 85
- INIT_CAP= argument
 - NEW INFOMAP statement 47
 - UPDATE CURRENT_INFOMAP
 - statement 54
 - UPDATE INFOMAP statement 68
- INSERT DATAITEM statement
 - ACTIONS= argument 20
 - AGGREGATION= argument 21
 - AGGREGATIONS_DROP_LIST=
 - argument 24
 - AGGREGATIONS_KEEP_LIST=
 - argument 24
 - CLASSIFICATION= argument 24
 - COLUMN= argument 18
 - CUSTOM_PROPERTIES= argument
 - 25
 - DESCRIPTION= argument 25
 - EXPRESSION= argument 18
 - FOLDER= argument 25
 - FORMAT= argument 26
 - HIERARCHY= argument 19
 - ID= argument 26
 - INFOMAPS procedure 17
 - MEASURE= argument 20
 - NAME= argument 27
 - TYPE= argument 27
 - VALUE_GENERATION= argument
 - 27
- INSERT DATASOURCE statement
 - _ALL_ argument 30
 - COLUMNS= argument 30

- CUBE= argument 29
 - DESCRIPTION= argument 30
 - ID= argument 31
 - INFOMAPS procedure 28
 - NAME= argument 31
 - REQUIRED_DATASOURCE= argument 31
 - SASSERVER= argument 29
 - TABLE= argument 29
 - INSERT FILTER statement
 - CONDITION= argument 33
 - CUSTOM_PROPERTIES= argument 33
 - DESCRIPTION= argument 33
 - FOLDER= argument 34
 - HIDDEN= argument 34
 - ID= argument 34
 - INFOMAPS procedure 32
 - NAME= argument 34
 - INSERT FOLDER statement
 - CUSTOM_PROPERTIES argument 36
 - DESCRIPTION= argument 36
 - INFOMAPS procedure 35
 - LOCATION= argument 36
 - INSERT IDENTITY_PROPERTY statement
 - ID= argument 38
 - INFOMAPS procedure 37
 - PROPERTY= argument 37
 - INSERT RELATIONSHIP statement
 - CARDINALITY= argument 40
 - CONDITION= argument 40
 - CUSTOM_PROPERTIES= argument 40
 - DESCRIPTION= argument 40
 - ID= argument 40
 - INFOMAPS procedure 39
 - JOIN= argument 41
 - LEFT_TABLE= argument 40
 - RIGHT_TABLE= argument 40
 - InternalAggregation function 21
 - InternalAggregationAdditive function 21
 - IP address
 - metadata server 90
- J**
- JOIN_MODEL= argument
 - NEW INFOMAP statement 48
 - UPDATE CURRENT_INFOMAP statement 54
 - UPDATE INFOMAP statement 69
 - join models 48
 - specifying for current information map 54
 - specifying for information map 48, 69
 - join strategy
 - See [join models](#)
 - join type
 - specifying for relationships 41
 - JOIN= argument
 - INSERT RELATIONSHIP statement 41
 - UPDATE RELATIONSHIP statement 74
- L**
- LEFT_TABLE= argument
 - INSERT RELATIONSHIP statement 40
 - LIBNAME statement, Information Maps engine 87
 - AGGREGATE= argument 91
 - connection options 89
 - DOMAIN= argument 89
 - examples 82
 - EXPCOLUMNLEN= argument 91
 - INFOMAPS argument 88
 - MAPPATH= argument 88
 - METACREDENTIALS= argument 89
 - METAPASS= argument 89
 - METAPORT= argument 90
 - METASERVER= argument 90
 - METAUSER= argument 90
 - PRESERVE_MAP_NAMES= argument 91
 - READBUFF= argument 92
 - SPOOL= argument 92
 - SSPI= argument 90
 - syntax 87
 - library definitions
 - creating in metadata server 106
 - librefs 81, 88
 - LIST statement
 - _ALL_ argument 42
 - DATAITEMS argument 42
 - DATASOURCES argument 42
 - FILTERS argument 42
 - INFOMAPS procedure 41
 - RELATIONSHIPS argument 42
 - LOCALES= argument
 - IMPORT LOCALIZED_PROPERTIES statement 16
 - localizable properties
 - exporting 13
 - importing 15
 - LOCATION= argument
 - INSERT FOLDER statement 36
 - MOVE FOLDER statement 45
 - SET STORED PROCESS statement 52
 - UPDATE FOLDER statement 66

M

MAPPATH= argument
 DELETE INFOMAP statement 11
 EXPORT statement 12
 LIBNAME statement 88
 NEW INFOMAP statement 48
 PROC INFOMAPS statement 9
 SAVE statement 50
 UPDATE INFOMAP statement 69
 MEASURE= argument
 INSERT DATAITEM statement 20
 member-name normalization 92
 METACREDENTIALS= argument
 LIBNAME statement 89
 PROC INFOMAPS statement 9
 metadata 1
 metadata server
 connecting to 7, 89
 creating library definitions 106
 deleting information maps from 11
 IP address 90
 IP address of host 9
 LIBNAME statement connection options 89
 passwords for 9, 89
 TCP port 9, 90
 user ID for connecting 10, 90
 METALIB procedure
 registering data 107
 METAPASS= argument
 LIBNAME statement 89
 PROC INFOMAPS statement 9
 METAPORT= argument
 LIBNAME statement 90
 PROC INFOMAPS statement 9
 METASERVER= argument
 LIBNAME statement 90
 PROC INFOMAPS statement 9
 METAUUSER= argument
 LIBNAME statement 90
 PROC INFOMAPS statement 10
 MOVE DATAITEM statement
 ID_LIST= argument 44
 INFOMAPS procedure 44
 NEW_LOCATION= argument 44
 MOVE FILTER statement
 ID_LIST= argument 44
 INFOMAPS procedure 44
 NEW_LOCATION= argument 44
 MOVE FOLDER statement
 INFOMAPS procedure 45
 LOCATION= argument 45
 NEW_LOCATION= argument 45
 moving data items
 specifying new location 44
 moving filters

specifying new location 44
 moving folders
 specifying new location 45

N

NAME= argument
 INSERT DATAITEM statement 27
 INSERT DATASOURCE statement 31
 INSERT FILTER statement 34
 SET STORED PROCESS statement 52
 UPDATE DATAITEM statement 60
 UPDATE DATASOURCE statement 62
 UPDATE FILTER statement 64
 UPDATE FOLDER statement 66
 names
 assigning to data items 27, 60
 assigning to data sources 31, 62
 assigning to filters 34, 64
 assigning to folders 66
 assigning to stored processes 52
 natural language names 103
 restrictions on 102
 special characters in information map names 91
 tables 101
 using special characters in 103
 natural language names 103
 NEW_LOCATION= argument
 MOVE DATAITEM statement 44
 MOVE FILTER statement 44
 MOVE FOLDER statement 45
 NEW INFOMAP statement
 AUTO_REPLACE= argument 47
 CREATE_TARGET_FOLDER= argument 47
 CUSTOM_PROPERTIES= argument 47
 DESCRIPTION= argument 47
 INFOMAPS procedure 45
 INIT_CAP= argument 47
 JOIN_MODEL= argument 48
 MAPPATH= argument 48
 REPLACE_UNDERSCORES= argument 48
 USE_LABELS= argument 48
 VERIFY= argument 48
 NOERRORSTOP argument
 PROC INFOMAPS statement 8
O
 ODS reports 119
 OLAP cubes
 assigning as data sources 29

- OLAP data
 - conditional expressions for 33
- P**
- PASSWORD= argument
 - See [METAPASS= argument](#)
- passwords
 - metadata server 9, 89
- performance
 - INFOMAPS procedure 101
 - Information Maps engine 102
- PERSON_NAME property 38
- PORT= argument
 - See [METAPORT= argument](#)
- prefilters
 - authorization based 70
 - defining 51
 - for all users 51
- PRESERVE_MAP_NAMES= argument
 - LIBNAME statement 91
- PRINT procedure
 - printing information map data 82
 - printing information maps 117
- PROC INFOMAPS statement
 - DOMAIN= argument 8
 - ERRORSTOP argument 8
 - INFOMAPS procedure 7
 - MAPPATH= argument 9
 - METACREDENTIALS= argument 9
 - METAPASS= argument 9
 - METAPORT= argument 9
 - METASERVER= argument 9
 - METAUSER= argument 10
 - NOERRORSTOP argument 8
- properties, SAS identity 38
- PROPERTY= argument
 - INSERT IDENTITY_PROPERTY statement 37
- R**
- READBUFF= argument
 - LIBNAME statement 92
- READBUFF= data set option
 - Information Maps engine 98
- relational data
 - conditional expressions for 33
- relational tables
 - assigning as data sources 29
- relationships
 - assigning custom properties 40, 73
 - assigning descriptions 40, 74
 - assigning identifiers (IDs) 40
 - conditional expression for 40
 - data sources for 40
 - inserting in information maps 39
 - specifying cardinality of 40
 - specifying join type for 41
 - updating conditional expressions 73
 - updating in information maps 72
 - updating join types 74
 - validating in information maps 48
- RELATIONSHIPS argument
 - LIST statement 42
- REPLACE_UNDERSCORES= argument
 - NEW INFOMAP statement 48
 - UPDATE CURRENT_INFOMAP statement 55
 - UPDATE INFOMAP statement 69
- REQUIRED_DATASOURCE= argument
 - INSERT DATASOURCE statement 31
 - UPDATE DATASOURCE statement 62
- REQUIRED_DATASOURCES= argument
 - UPDATE CURRENT_INFOMAP statement 55
 - UPDATE INFOMAP statement 69
- required data sources 31, 69
- results filters
 - authorization based 70
 - defining 51
 - for all users 51
- RIGHT_TABLE= argument
 - INSERT RELATIONSHIP statement 40
- S**
- SAS identity properties 38
 - assigning identifiers (IDs) 38
 - deleting 11
 - EXTERNAL_IDENTITY 37
 - IDENTITY_GROUP_NAME 37
 - IDENTITY_GROUPS 38
 - IDENTITY_NAME 38
 - inserting in information maps 37
 - PERSON_NAME 38
 - removing from the information map 11
 - USERID 38
- SAS Information Map Studio 6
- SAS information maps
 - See [information maps](#)
- SAS name literals 91
- SAS.ExternalIdentity property 37
- SAS.IdentityGroupName property 37
- SAS.IdentityGroups property 38
- SAS.IdentityName property 38
- SAS.PersonName property 38
- SAS.Userid property 38
- SASSERVER= argument

- INSERT DATASOURCE statement 29
- SAVE statement
 - ALLOW_MAJOR_VERSION_UPGRADE= argument 49
 - ALLOW_MINOR_VERSION_UPGRADE= argument 50
 - INFOMAP argument 50
 - INFOMAPS procedure 49
 - MAPPATH= argument 50
- SERVER= argument
 - See METASERVER= argument
- SET ASSIGNED_FILTERS statement
 - DEFINITION= argument 51
 - INFOMAPS procedure 51
- SET STORED PROCESS statement
 - INFOMAPS procedure 52
 - LOCATION= argument 52
 - NAME= argument 52
- special characters
 - in information map names 91
 - using in names 103
- spool file 92
- SPOOL= argument
 - LIBNAME statement 92
- SSPI= argument
 - LIBNAME statement 90
- stored processes
 - assigning names 52
 - associating with information map 52

- T**
- TABLE= argument
 - INSERT DATASOURCE statement 29
- tables
 - naming 101
- TCP port 9
 - metadata server 90
- TYPE= argument
 - INSERT DATAITEM statement 27
 - UPDATE DATAITEM statement 60
- types
 - assigning to data items 27, 60

- U**
- UPDATE CURRENT_INFOMAP statement
 - CREATE_TARGET_FOLDER= argument 53
 - CUSTOM_PROPERTIES= argument 53
 - DESCRIPTION= argument 54
 - INFOMAPS procedure 52
 - INIT_CAP= argument 54
 - JOIN_MODEL= argument 54
- REPLACE_UNDERSCORES= argument 55
- REQUIRED_DATASOURCES= argument 55
- USE_LABELS= argument 56
- VERIFY= argument 56
- UPDATE DATAITEM statement
 - ACTIONS= argument 57
 - AGGREGATION= argument 58
 - AGGREGATIONS_LIST= argument 58
 - CLASSIFICATION= argument 59
 - CUSTOM_PROPERTIES= argument 59
 - DESCRIPTION= argument 60
 - EXPRESSION= argument 60
 - FORMAT= argument 60
 - ID= argument 60
 - INFOMAPS procedure 56
 - NAME= argument 60
 - TYPE= argument 60
 - VALUE_GENERATION= argument 61
- UPDATE DATASOURCE statement
 - DESCRIPTION= argument 62
 - ID= argument 62
 - INFOMAPS procedure 62
 - NAME= argument 62
 - REQUIRED_DATASOURCE= argument 62
- UPDATE FILTER statement
 - CONDITION= argument 63
 - CUSTOM_PROPERTIES= argument 63
 - DESCRIPTION= argument 64
 - HIDDEN= argument 64
 - ID= argument 64
 - INFOMAPS procedure 63
 - NAME= argument 64
- UPDATE FOLDER statement
 - CUSTOM_PROPERTIES= argument 65
 - DESCRIPTION= argument 66
 - INFOMAPS procedure 65
 - LOCATION= argument 66
 - NAME= argument 66
- UPDATE INFOMAP statement
 - CREATE_TARGET_FOLDER= argument 67
 - CUSTOM_PROPERTIES= argument 68
 - DESCRIPTION= argument 68
 - INFOMAPS procedure 66
 - INIT_CAP= argument 68
 - JOIN_MODEL= argument 69
 - MAPPATH= argument 69

- REPLACE_UNDERSCORES=
 - argument 69
 - REQUIRED_DATASOURCES=
 - argument 69
 - USE_LABELS= argument 70
 - VERIFY= argument 70
 - UPDATE MAP_PERMISSIONS
 - statement
 - INFOMAPS procedure 70
 - UPDATE RELATIONSHIP statement
 - CARDINALITY= argument 73
 - CONDITION= argument 73
 - CUSTOM_PROPERTIES= argument 73
 - DESCRIPTION= argument 74
 - ID= argument 74
 - INFOMAPS procedure 72
 - JOIN= argument 74
 - USE_LABELS= argument
 - NEW INFOMAP statement 48
 - UPDATE CURRENT_INFOMAP statement 56
 - UPDATE INFOMAP statement 70
 - user ID
 - authenticating 38
 - connecting to metadata server 10, 90
 - SAS identity properties and 39
 - USERID property 38
 - USERID= argument
 - See *METAUSER= argument*
- V**
- VALUE_GENERATION= argument
 - INSERT DATAITEM statement 27
 - UPDATE DATAITEM statement 61
 - value generation methods
 - assigning to data items 27, 61
 - VERIFY= argument
 - NEW INFOMAP statement 48
 - UPDATE CURRENT_INFOMAP statement 56
 - UPDATE INFOMAP statement 70
- W**
- WHERE clause
 - filter as 63
 - filters as 32
 - WHERE clauses 85
- X**
- XML files
 - exporting information maps in 12
 - importing information maps from 14