Base SAS® 9.4 Guide to Information Maps
# Contents

**What’s New in the INFOMAPS Procedure and the Information Maps LIBNAME Engine** for SAS 9.4 ................................................................. v

**Accessibility** ................................................................................. 0

## 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 ................................................. 6
- Examples: INFOMAPS Procedure ........................................... 109

## Chapter 3 / Using the SAS Information Maps LIBNAME Engine ............ 115
- What Does the Information Maps Engine Do? .......................... 115
- Understanding How the Information Maps Engine Works ........... 115
- Advantages of Using the Information Maps Engine .................. 120
- What Is Required to Use the Information Maps Engine? ............. 120
- What Is Supported? .................................................................. 120

## Chapter 4 / LIBNAME Statement for the Information Maps Engine ....... 123
- Using the LIBNAME Statement ................................................ 123
- Dictionary .............................................................................. 124
- Examples: ............................................................................. 130

## Chapter 5 / SAS Data Set Options for the Information Maps Engine .... 133
- Using Data Set Options ............................................................ 133
- Dictionary .............................................................................. 133

## Chapter 6 / Hints and Tips for Using the INFOMAPS Procedure or the Information Maps Engine .................................................. 139
- Hints and Tips for Using the INFOMAPS Procedure ................. 139
- Hints and Tips for Using the Information Maps Engine ............... 140

## Chapter 7 / Example: Using the INFOMAPS Procedure and the Information Maps Engine .................................................. 143
- About This Example .................................................................. 143
- Step 1: Create a Library Definition in the SAS Metadata Server .......... 144
- Step 2: Set the Metadata System Options and a Macro Variable ........ 144
- Step 3: Register Data Using the METALIB Procedure ................. 145
- Step 4: Create an Information Map Using the INFOMAPS Procedure ... 147
- Step 5: Retrieve the Data Associated with the Information Map Using the Information Maps Engine ................................................. 153
- Step 6: View the Data Items and Filters Using the CONTENTS Procedure 154
- Step 7: Print the Data from the Information Map .......................... 156
- Step 8: Analyze the Data in SAS and Produce an ODS Report ........ 158
### Appendix 1 / SQL Dictionary Tables for the Information Maps Engine

- Using SQL DICTIONARY Tables ........................................... 161
- DICTIONARY.INFOMAPS Table ........................................... 162
- DICTIONARY.DATAITEMS Table ......................................... 162
- DICTIONARY.FILTERS Table .............................................. 164

### Appendix 2 / SAS Tracing and the Information Maps Engine

- SAS System Options for Diagnostic Messages ....................... 165
What’s New in the INFOMAPS Procedure and the Information Maps LIBNAME Engine for SAS 9.4

Overview

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

- Support for creating date, numeric, OLAP member, text, time, and timestamp prompts.
- Support for deleting data items, data sources, filters, and relationships from an information map.
- Support for controlling whether users can drill down to detail data when the data source for an information map is an OLAP cube.
- Support for a locale for data in an information map that is different from the current SAS session locale.
- Support for clustered metadata servers.

The Information Maps LIBNAME engine supports clustered metadata servers.

INFOMAPS Procedure Features

The following statements are new:

DELETE DATAITEM
    deletes one or more data items from an information map.

DELETE DATASOURCE
    deletes one or more data sources from an information map.

DELETE FILTER
    deletes one or more filters from an information map.

DELETE RELATIONSHIP
    deletes one or more relationships from an information map.
INSERT DATE_PROMPT
inserts a date prompt into an information map.

INSERT NUMERIC_PROMPT
inserts a numeric prompt into an information map.

INSERT OLAP_MEMBER_PROMPT
inserts an OLAP member prompt into an information map.

INSERT TEXT_PROMPT
inserts a text prompt into an information map.

INSERT TIME_PROMPT
inserts a time prompt into an information map.

INSERT TIMESTAMP_PROMPT
inserts a timestamp prompt into an information map.

The NEW INFOMAP, UPDATE CURRENT_INFOMAP, and UPDATE INFOMAP statements support the following new options:

ALLOW_DRILL_THROUGH=
    specifies whether users can drill down to detail data when the data source for an information map is an OLAP cube.

DATA_LOCALE=
    specifies a supported locale value that is used as the data locale of an information map.

The INFOMAP procedure supports a clustered metadata server. If you do not specify an explicit server name and port number in the PROC INFOMAPS statement, then the procedure retrieves the list of server names and port numbers specified in the metadata profile or the METASERVER and METAPORT options.

---

Information Maps LIBNAME Engine Features

The Information Maps LIBNAME engine supports a clustered metadata server. When no server or port is specified in the LIBNAME statement, the INFOMAP engine retrieves the list of servers and corresponding ports from the metadata profile.
Overview of SAS Information Maps

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.
Overview: INFOMAPS Procedure

What Does the INFOMAPS Procedure Do?

The INFOMAPS procedure enables you to create information maps programmatically. You can also use the procedure to modify an existing information map by adding new data sources, data items, filters, folders, or relationships. Or you can change the definitions of any existing data item, filter, data source, folder, or relationship within an information map.

A SAS Information Map is a business metadata layer that is applied on top of the data sources in your data warehouse. (Metadata is information about the structure and content of data. An information map does not contain any physical data.) Information maps provide business users with a user-friendly way to query data and get results for themselves. For example, you can create data items with names such as "Age Group" or "Sales Revenue from Internet Orders."

Information maps can contain data items and filters, which are used to build queries. A data item can refer to a physical data source such as one or more columns from a table, to an OLAP hierarchy, or to an OLAP measure. It can also refer to one or more other data items in the same information map. A data item is classified as either a measure item or a category item. Measure items can be used for calculations. Category items are used to group measure items. Filters contain criteria for subsetting the data that is returned for a query. You can organize data items and filters into folders and subfolders to help users find the information 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 Chapter 3, “Using the SAS Information Maps LIBNAME Engine,” on page 115.

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 DATAITEM _ALL_ | ID="data-item-ID" <option>;
DELETE DATASOURCE "ALL_" | ID="data-source-ID" <option>;
DELETE FILTER "ALL_" | ID="filter-ID"<option>;
DELETE IDENTITYPROPERTY "ALL_" | ID="identity-property-ID";
DELETE INFORMAP "information-map-name" <options>;
DELETE RELATIONSHIP "ALL_" | ID="relationship-ID" <option>;
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 DATE_PROMPT DISPLAYED_TEXT="displayed-text" <options>;
INSERT FILTER CONDITION="conditional-expression" <options>;
INSERT FOLDER "folder-name" <options>;
INSERT IDENTITYPROPERTY PROPERTY=property-keyword
   <ID="identity_property-ID">;
INSERT NUMERIC_PROMPT DISPLAYED_TEXT="displayed-text" <options>;
INSERT RELATIONSHIP <options>;
INSERT TEXT_PROMPT DISPLAYED_TEXT="displayed-text" <options>;
INSERT TIME_PROMPT DISPLAYED_TEXT="displayed-text" <options>;
INSERT TIMESTAMP_PROMPT DISPLAYED_TEXT="displayed-text"
   <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 INFORMAP "information-map-name" <options>;
SAVE <options>;
SET ASSIGNEDFILTERS DEFINITION=(<data-source-filters-1
   <… data-source-filters-n>));
SET STORED PROCESS NAME="stored-process-name" <option>;
UPDATE CURRENT_INFORMAP <options>;
UPDATE DATAITEM "data-item-ID" <options>;
UPDATE DATASOURCE "data-source-ID" <options>;
UPDATE FILTER "filter-ID" <options>;
UPDATE FOLDER "folder-name" <options>;
UPDATE INFORMAP "information-map-name" <options>;
UPDATE MAP_PERMISSIONS GROUP="identity" | USER="identity"
   permission-specification-1 <… permission-specification-n>;
UPDATE RELATIONSHIP "relationship-ID" <options>;

Syntax: INFOMAPS Procedure
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 runtime 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 XEXITERROR when an error is encountered.

<table>
<thead>
<tr>
<th>Default</th>
<th>NOERRORSTOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>The ERRORSTOP option has no effect when the INFOMAPS procedure is used in interactive mode.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Aliases</th>
<th>PASSWORD=</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PW=</td>
</tr>
<tr>
<td>Examples</td>
<td>metapass=&quot;My Password*</td>
</tr>
<tr>
<td></td>
<td>metapass=&quot;MyPassword&quot;</td>
</tr>
</tbody>
</table>
**METAPORT=port-number**
specifies the TCP port that the metadata server is listening to for connections.

<table>
<thead>
<tr>
<th>Alias</th>
<th>PORT=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>If this option is not specified, the value is obtained from the METAPORT= system option. See the METAPORT= system option in <em>SAS Language Interfaces to Metadata</em>.</td>
</tr>
<tr>
<td>Example</td>
<td>metaport=8561</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Aliases</th>
<th>SERVER=</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST=</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>If this option is not specified, the value is obtained from the METASERVER= system option. See the METASERVER= system option in <em>SAS Language Interfaces to Metadata</em>.</td>
</tr>
<tr>
<td>Example</td>
<td>metaserver=&quot;myip.us.mycompany.com&quot;</td>
</tr>
</tbody>
</table>

**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*.

<table>
<thead>
<tr>
<th>Aliases</th>
<th>USER=</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID=</td>
<td></td>
</tr>
<tr>
<td>ID=</td>
<td></td>
</tr>
<tr>
<td>Restrictions</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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. Use the following format:</td>
</tr>
<tr>
<td></td>
<td>metauser=&quot;Windows-domain-name\user-ID&quot;</td>
</tr>
<tr>
<td>Note</td>
<td>The user ID is not case sensitive.</td>
</tr>
<tr>
<td>Example</td>
<td>metauser=&quot;myUserID&quot;</td>
</tr>
</tbody>
</table>
Example

```plaintext
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 DATAITEM Statement

Deletes one or more data items from the current information map.

Syntax

```
DELETE DATAITEM _ALL_ | ID="data-item-ID" <option>;
```

Required Arguments

_**ALL**_  
removes all data items from the current information map.

**ID="data-item-ID"**  
removes the specified data item from the current information map.

Optional Argument

**CASCADE_DELETE=YES | NO**  
specifies whether the deletion of the data item triggers the deletion of all the business model objects or business rules that reference the data item. If the CASCADE_DELETE= option is set to NO and a data item is currently being
referenced by a business object, then the DELETE DATAITEM statement is ignored and a warning message is written to the log.

Default: NO

Details
For more information about data items, see “INSERT DATAITEM Statement” on page 20.

DELETE DATASOURCE Statement
Deletes one or more data sources from the current information map.

Syntax
DELETE DATASOURCE _ALL_ | ID="data-source-ID" <option>;

Required Arguments
_ALL_
removes all data sources from the current information map.

ID="data-source-ID"
removes the specified data source from the current information map.

Optional Argument
CASCADE_DELETE=YES | NO
specifies whether the deletion of the data source triggers the deletion of all the business model objects or business rules that reference the data source. If the CASCADE_DELETE= option is set to NO and a data source is currently being referenced by a business object, then the DELETE DATASOURCE statement is ignored and a warning message is written to the log.

Default: NO

Details
For more information about data sources, see “INSERT DATASOURCE Statement” on page 33.

DELETE FILTER Statement
Deletes one or more filters from the current information map.
DELETE FILTER Statement

Syntax

DELETE FILTER _ALL_ | ID="filter-ID"<option>;

Required Arguments

_ALL_  
removes all filters from the current information map.

ID="filter-ID"  
removes the specified filter from the current information map.

Optional Argument

CASCADE_DELETE=YES | NO  
specifies whether the deletion of the filter triggers the deletion of all the business model objects or business rules that reference the filter. If the CASCADE_DELETE= option is set to NO and a filter is currently being referenced by a business object, then the DELETE FILTER statement is ignored and a warning message is written to the log.

Default  NO

Details

For more information about filters, see “INSERT FILTER Statement” on page 43.

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 48.
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";

DELETE RELATIONSHIP Statement
Deletes one or more relationships from the current information map.

Syntax
DELETE RELATIONSHIP _ALL_ | ID="relationship-ID" <option>;

Required Arguments
_ALL_
removes all relationships from the current information map.
ID="relationship-ID"
removes the specified relationship from the current information map.
Optional Argument

CASCADE_DELETE=YES | NO

specifies whether the deletion of the relationship triggers the deletion of all the business model objects or business rules that reference the relationship. If the CASCADE_DELETE= option is set to NO and a relationship is currently being referenced by a business object, then the DELETE RELATIONSHIP statement is ignored and a warning message is written to the log.

Default NO

Details

For more information about relationships, see "INSERT RELATIONSHIP Statement" on page 57.

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.
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:

  \( (\text{information-map-name-1} <... \text{information-map-name-n}>)/\text{KEEP} \mid/\text{DROP} \)  

  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 19.
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_PROPS

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.
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, Information Maps Engine Statements" on page 124. For more information about the EXPCOLUMNLEN= data set option, see "EXPCOLUMNLEN= Data Set Option" on page 134.

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:

```sql
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). The data item Geographical2 in the following example returns all members of the level:

```sql
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. The following example shows the use of the EXPRESSION option:

```sql
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:

```sql
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 filters can be applied to members of OLAP data items that represent hierarchies. 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. The setting also 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 25. 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, \(\text{SUM(COL1)}/\text{COUNT(COL1)}\) 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, \(\text{SUM(COL1*COL2)}\) 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 for Information Maps

<table>
<thead>
<tr>
<th>Function</th>
<th>Definition</th>
<th>Available to nonnumeric item that is a measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG</td>
<td>average (mean) of values</td>
<td></td>
</tr>
<tr>
<td>AvgDistinct</td>
<td>average (mean) of distinct values</td>
<td></td>
</tr>
<tr>
<td>COUNT</td>
<td>number of nonmissing values</td>
<td></td>
</tr>
<tr>
<td>CountDistinct</td>
<td>number of distinct nonmissing values</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Definition</td>
<td>Available to nonnumeric item that is a measure</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>CountPlusNMISS</td>
<td>number of values (including the number of missing values)</td>
<td>✓</td>
</tr>
<tr>
<td>CountPlusNMISSDistinct</td>
<td>number of distinct values (including the number of distinct missing values)</td>
<td>✓</td>
</tr>
<tr>
<td>CSS</td>
<td>corrected sum of squares</td>
<td></td>
</tr>
<tr>
<td>CSSDistinct</td>
<td>corrected sum of squares of distinct values</td>
<td></td>
</tr>
<tr>
<td>CV</td>
<td>coefficient of variation (percent)</td>
<td></td>
</tr>
<tr>
<td>CVDistinct</td>
<td>coefficient of variation (percent) of distinct values</td>
<td></td>
</tr>
<tr>
<td>FREQ</td>
<td>number of nonmissing values</td>
<td>✓</td>
</tr>
<tr>
<td>FreqDistinct</td>
<td>number of distinct nonmissing values</td>
<td>✓</td>
</tr>
<tr>
<td>InternalAggregation</td>
<td>defined in an expression (nonadditive)</td>
<td>✓</td>
</tr>
<tr>
<td>InternalAggregationAdditive</td>
<td>defined in an expression (additive)</td>
<td>✓</td>
</tr>
<tr>
<td>MAX</td>
<td>largest value</td>
<td>✓</td>
</tr>
<tr>
<td>MEAN</td>
<td>mean (average) of values</td>
<td></td>
</tr>
<tr>
<td>MeanDistinct</td>
<td>mean (average) of distinct values</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>smallest value</td>
<td>✓</td>
</tr>
<tr>
<td>N</td>
<td>number of nonmissing values</td>
<td>✓</td>
</tr>
<tr>
<td>NDistinct</td>
<td>number of distinct nonmissing values</td>
<td>✓</td>
</tr>
<tr>
<td>NMISS</td>
<td>number of missing values</td>
<td>✓</td>
</tr>
<tr>
<td>Function</td>
<td>Definition</td>
<td>Available to nonnumeric item that is a measure</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>NMISSDistinct</td>
<td>number of distinct missing values</td>
<td>☑</td>
</tr>
<tr>
<td>PRT</td>
<td>probability of a greater absolute value of Student's $t$</td>
<td></td>
</tr>
<tr>
<td>PRTDistinct</td>
<td>probability of a greater absolute value of Student's $t$ of distinct values</td>
<td></td>
</tr>
<tr>
<td>RANGE</td>
<td>range of values</td>
<td></td>
</tr>
<tr>
<td>RANGEDistinct</td>
<td>range of distinct values</td>
<td></td>
</tr>
<tr>
<td>STD</td>
<td>standard deviation</td>
<td></td>
</tr>
<tr>
<td>STDDistinct</td>
<td>standard deviation of distinct values</td>
<td></td>
</tr>
<tr>
<td>STDERR</td>
<td>standard error of the mean</td>
<td></td>
</tr>
<tr>
<td>STDERRDistinct</td>
<td>standard error of the mean of distinct values</td>
<td></td>
</tr>
<tr>
<td>SUM</td>
<td>sum of values</td>
<td></td>
</tr>
<tr>
<td>SumDistinct</td>
<td>sum of distinct values</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Student's $t$ value for testing the hypothesis that the population mean is zero</td>
<td></td>
</tr>
<tr>
<td>TDistinct</td>
<td>Student's $t$ value for testing the hypothesis that the population mean of distinct values is zero</td>
<td></td>
</tr>
<tr>
<td>USS</td>
<td>uncorrected sum of squares</td>
<td></td>
</tr>
<tr>
<td>USSDistinct</td>
<td>uncorrected sum of squares for distinct values</td>
<td></td>
</tr>
<tr>
<td>VAR</td>
<td>variance</td>
<td></td>
</tr>
<tr>
<td>VarDistinct</td>
<td>variance of distinct values</td>
<td></td>
</tr>
</tbody>
</table>
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 25 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 25 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.
This option applies only to relational data items that are measures.

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 fails. Therefore, you should 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**

```plaintext```
custom_properties={
  ("MA_Level" "Nominal" "Descriptive text goes here.")
```plaintext```
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. An example is folder="CUSTOMERS".
- If the folder is not in the root directory, then you must specify the location of the folder beginning with a slash. An example is 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, then 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 DataItem number, where number is an internally maintained counter for ID generation. This counter is also used for generating IDs for other business data, including data sources, filters, and relationships.
  - If the data item is inserted with the HIERARCHY= or MEASURE= option, then the ID is generated from the caption of the hierarchy or measure.

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

**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. If a data item in the information map has a name that is more than 32 bytes long, then the name is truncated to a unique name when it is used in a SAS program.

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 requires users to manually enter 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

```plaintext
value_generation=
    ["CA" "California"]
    ["NC"]
    ["NY" "New York"]
```

Example

```plaintext
/* 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ALL</em></td>
<td>specifies to insert a data item for each physical column or hierarchy as defined in the specified table or cube.</td>
</tr>
<tr>
<td>COLUMNS=(column-1 &lt;… column-n&gt;)</td>
<td>specifies one or more physical column names as defined in the specified table.</td>
</tr>
<tr>
<td>DESCRIPTION=&quot;descriptive-text&quot;</td>
<td>specifies the description of the data source.</td>
</tr>
<tr>
<td>ID=&quot;data-source-ID&quot;</td>
<td>specifies the ID assigned to the data source.</td>
</tr>
<tr>
<td>NAME=&quot;data-source-name&quot;</td>
<td>enables you to specify a descriptive name for each data source inserted in an information map.</td>
</tr>
<tr>
<td>REQUIRED_DATASOURCE= YES</td>
<td>NO</td>
</tr>
</tbody>
</table>
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.

**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 in its 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.

**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 DATE_PROMPT Statement

Inserts a date prompt into the current information map.

Syntax

```plaintext
INSERT DATE_PROMPT DISPLAYED_TEXT="displayed-text"
    COLUMN="relational-table-path"."column-name" <options>;

INSERT DATE_PROMPT DISPLAYED_TEXT="displayed-text"
    DATAITEM="data-item-ID" <options>;

INSERT DATE_PROMPT DISPLAYED_TEXT="displayed-text"
    VALUES=USER | custom-values-list <options>;
```

Summary of Optional Arguments

- **ALLOW_MISSING_VALUES=** YES | NO
  specifies whether SAS missing values are supported in the prompt.
- **ALLOW_USER_VALUES=** YES | NO
  specifies whether the user can specify additional unformatted values at run time.
- **DATE_TYPE=** DAY | WEEK | MONTH | QUARTER | YEAR
  specifies the type of date prompt to create.
- **DEFAULT_VALUES=** "SAS-date-value" | "formatted-date-string" | "relative-date"
  specifies the default date prompt value.
- **DESCRIPTION=** "descriptive-text"
specifies the description of the date prompt.

**HIDDEN=** "YES | NO"
- specifies whether the prompt is hidden at run time.

**ID=** "prompt-ID"
- specifies the ID of the prompt to insert.

**INTEGER_ONLY=** "YES | NO"
- specifies whether the prompt values have to be of integer type.

**MAX_DECIMAL=** "maximum-decimal_places"
- specifies the maximum number of decimal places that are allowed in an unformatted prompt value.

**MAX_VALUE_COUNT=** "maximum-value-count"
- specifies the maximum number of selections that the user can specify at a time.

**MAX_VALUE=** "maximum-value"
- specifies the maximum value allowed for an unformatted prompt value.

**MIN_DECIMAL=** "minimum-decimal_places"
- specifies the minimum number of decimal places that are required in an unformatted prompt value.

**MIN_VALUE_COUNT=** "minimum-value-count"
- specifies the minimum number of selections that the user needs to make at the prompt.

**MIN_VALUE=** "minimum-value"
- specifies the minimum value allowed for an unformatted prompt value.

**NAME=** "prompt-name"
- specifies the name of a prompt to insert into the current information map.

**READ_ONLY=** "YES | NO"
- specifies whether the prompt is read-only.

**SELECTION_TYPE=** "SINGLE | MULTIPLE | MULTIPLE_ORDERED"
- specifies whether the user can make multiple selections from the prompt.

**SORT=** "NONE | ASCENDING_LABEL | DESCENDING_LABEL | ASCENDING_VALUE | DESCENDING_VALUE"
- specifies how the list of available prompt values is sorted for display.

**VALUE DISPLAYED=** "YES | NO"
- specifies whether formatted values are appended to unformatted values in the prompt list.

### Required Arguments

The **INSERT DATE_PROMPT** statement must include one of the **COLUMN=**, **DATAITEM=**, or **VALUES=** arguments:

**COLUMN=** "relational-table-path"."column-name-1"
- specifies that the prompt use columns from an external table source to dynamically generate values.

- **relational-table-path** specifies the physical path of a relational table where the specified columns are defined.

- **column-name-1** specifies the table column to be used for generating unformatted values. The value must be the SAS name of a column defined in the specified relational table.
The DISTINCT= option indicates whether the prompt values derived from the specified column are distinct.

column-name-2
specifies the table column to be used for generating formatted values. If column-name-2 is not specified, then the column specified in column-name-1 is used to generate the formatted values.

You can specify a format to apply to values from the column specified in column-name-2.

Interaction
You can specify only one of the COLUMN=, DATAITEM=, and VALUES= options in an INSERT DATE_PROMPT statement.

Example
insert date_prompt
displayed_text="Select a date:"
id=datePromptWithColumn*
value_displayed=yes
allow_user_values=yes
minimum_value="01Jan1980"
sort=ascending_label
column="/Shared Data/Transaction.*Cust_ID"
    label_column="Sales_Amount"
default_value="01May1985"d;

DATAITEM="data-item-ID-1" <LABEL_DATAITEM="data-item-ID-2" /></FORMAT="format-name">
specifies that the prompt uses data items from the current information map to dynamically generate values.

data-item-ID-1
specifies the data item used for generating unformatted values.

data-item-ID-2
specifies the data item to be used for generating formatted values. If data-item-ID-2 is not specified, then the data item specified in data-item-ID-1 is used to generate the formatted values.

You can specify a format to apply to values from the data item specified in data-item-ID-2.

Interaction
You can specify only one of the COLUMN=, DATAITEM=, and VALUES= options in an INSERT DATE_PROMPT statement.

Example
insert date_prompt
displayed_text="Select a date:"
id=numPromptWithDataItem*
value_displayed=yes
allow_user_values=yes
allow_missing_values=yes
dataitem="di_t1dateID" label_dataItem="di_t1date"
default_value="01May1985"d;

DISPLAYED_TEXT="displayed-text"
specifies the text that end users see when they are prompted for a value.

VALUES=USER | custom-values-list

USER
specifies that the user must enter a value in response to this prompt.
custom-values-list

defines a custom list of values for the prompt to display. The form of the
custom-values-list is

("unformatted-value-1" "formatted-value-1")
...
("unformatted-value-n" "formatted-value-n")

where

unformatted-value

specifies the unformatted value for a report. This must be an integer value
unless the INTEGER_ONLY= option is disabled.

formatted-value

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

Example

insert date_prompt
displayed_text="Select a date:"
id="datePromptWithStaticValues"
value_displayed=yes
allow_user_values=yes
allow_missing_values=yes
min_value="01May1985"
max_value="01May2015"
values=("01May1988" "ONE")
("01Jul1990" "JAPAN")
("01May2012")
("D-5D2Y" "2 years from 5 days before today")
default_value="01Jul1990";

Default

USER

Interaction

You can specify only one of the COLUMN=, DATAITEM=, and
VALUES= options in an INSERT DATE_PROMPT statement.

Note

To refer to a custom value in the DEFAULT_VALUES= option, you
must specify the unformatted value rather than the formatted value.

Optional Arguments

ALLOW_MISSING_VALUES=YES | NO

specifies whether SAS missing values are supported in the prompt.

Default

NO

ALLOW_USER_VALUES=YES | NO

specifies whether the user can specify additional unformatted values at run
time that are not included when the prompt is created.

Default

NO

DATE_TYPE=DAY | WEEK | MONTH | QUARTER | YEAR

specifies the type of date prompt to create.

Default

DAY
Restriction  The DATE_TYPE= option applies only if the prompt being created uses a data item as the value provider and its data type is DAY.

**DEFAULT_VALUES=**"SAS-date-value"d | "formatted-date-string" | "relative-date"

specifies the default date prompt value. The value can be missing if ALLOW_MISSING_VALUES= option is enabled.

Requirement  At least one default value is required if either the HIDDEN= option or the READ_ONLY= option is enabled.

Interaction  You cannot create a prompted filter if the referenced prompt does not have a default value unless the VERIFY= option associated with the current information map is disabled.

**DESCRIPTION="descriptive-text"**

specifies the description of the date prompt, which can be viewed by the information map consumer.

**Alias  DESC=**

**HIDDEN=YES | NO**

specifies whether the prompt is hidden at run time.

Default   NO

**ID="prompt-ID"**

specifies the ID of the prompt to insert. If you do not specify an ID, the INFOMAPS procedure generates a unique ID from the prompt name.

Restrictions  The ID value cannot contain nulls and non-blank nonprintable characters. 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.

**INTEGER_ONLY=YES | NO**

specifies whether the prompt values have to be of integer type.

Default   YES

**MAX_DECIMAL=**maximum-decimal-places

specifies the maximum number of decimal places that are allowed in an unformatted prompt value.

Restriction  This option does not apply if the prompt allows integer values only.

**MAX_VALUE=**maximum-value

specifies the maximum value allowed for an unformatted prompt value. This must be an integer value unless the INTEGER_ONLY= option is disabled.
**MAX_VALUE_COUNT=**\textit{maximum-value-count}

specifies the maximum number of selections that the user can specify at a time.

**MIN_DECIMAL=**\textit{minimum-decimal-places}

specifies the minimum number of decimal places that are required in an unformatted prompt value.

\textbf{Restriction} This option does not apply if the prompt allows integer values only.

**MIN_VALUE=**\textit{minimum-value}

specifies the minimum value allowed for an unformatted prompt value. This must be an integer value unless the INTEGER_ONLY= option is disabled.

**MIN_VALUE_COUNT=**\textit{minimum-value-count}

specifies the minimum number of selections that the user needs to make at the prompt.

**NAME=**"\textit{prompt-name}"

specifies the name of a prompt to insert into the current information map. If the NAME= option is not specified, the INFOMAPS procedure generates a default name.

\textbf{Restriction} The prompt name cannot contain nulls or non-blank nonprintable characters.

**READ_ONLY=**\textit{YES | NO}

specifies whether the prompt is read-only.

\textbf{Default} NO

**SELECTION_TYPE=**\textit{SINGLE | MULTIPLE | MULTIPLE_ORDERED}

specifies whether the user can make multiple selections from the prompt.

\textit{SINGLE}

specifies that the user can select only one value from the prompt.

\textit{MULTIPLE}

specifies that the user can select multiple value from the prompt.

\textit{MULTIPLE_ORDERED}

specifies that the user can select multiple value from the prompt and that the selected values are ordered.

\textbf{Default} MULTIPLE if either the MAXIMUM_VALUE_COUNT= or MINIMUM_VALUE_COUNT= options specify a value greater than 1 or if more than one value is specified for the DEFAULT_VALUES= option; otherwise, SINGLE.

**SORT=**\textit{NONE | ASCENDING_LABEL | DESCENDING_LABEL | ASCENDING_VALUE | DESCENDING_VALUE}

specifies how the list of available prompt values is sorted for display.

\textit{NONE}

specifies that the prompt values are not sorted.

\textit{ASCENDING_LABEL}

sorts the prompt list in ascending order based on the formatted label values.

\textit{DESCENDING_LABEL}

sorts the prompt list in descending order based on the formatted label values.
ASCENDING_VALUE
sorts the prompt list in ascending order based on the unformatted values.

DESCENDING_VALUE
sorts the prompt list in descending order based on the unformatted values.

Default: NONE

Restriction: This option applies only when the COLUMN= or DATAITEM= option is used. In that case, the prompt list is dynamic.

VALUE_DISPLAYED=YES | NO
specifies whether formatted values are appended to unformatted values in the prompt list.

Default: NO

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. An example is «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 fails. Therefore, you should 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, which can be viewed by the information map consumer.

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. An example is `folder="CUSTOMERS"`.

If the folder is not in the root directory, then you must specify the location of the folder beginning with a slash. An example is `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 generates a default name.

Restrictions

Nulls and non-blank nonprintable characters are not valid characters for a filter name.
If a filter in the information map has a name that is more than 32 bytes long, then the name is truncated to a unique name when it is used in a SAS program.

Examples:

Example 1

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

Example 2

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

Example 3

```sas
/* 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>` 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 fails. Therefore, you should 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, which can be viewed by the information map consumer.

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. An example is 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. An example is folders="/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.

**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 does not 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**

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

insert filter "myFilter"
    condition="<<SECURITY.USER_ID>>=<<SasPersonName>>";
```
INSERT NUMERIC_PROMPT Statement
Inserts a numeric prompt into the current information map.

Syntax

```
INSERT NUMERIC_PROMPT DISPLAYED_TEXT="displayed-text"
COLUMN="relational-table-path"."column-name" <options>;
```

```
INSERT NUMERIC_PROMPT DISPLAYED_TEXT="displayed-text"
DATAITEM="data-item-ID" <options>;
```

```
INSERT NUMERIC_PROMPT DISPLAYED_TEXT="displayed-text"
VALUES=USER | custom-values-list <options>;
```

Summary of Optional Arguments

- **ALLOW_ALL_VALUES=**YES | NO
  specifies whether the “all possible values” special values are supported in the prompt.

- **ALLOW_MISSING_VALUES=**YES | NO
  specifies whether SAS numeric missing values are supported in the prompt.

- **ALLOW_USER_VALUES=**YES | NO
  specifies whether the user can specify additional unformatted values at run time.

- **DEFAULT_VALUES=** ALL | (numeric-value-1<... numeric-value-n>)
  specifies the default prompt values.

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

- **HIDDEN=**YES | NO
  specifies whether the prompt is hidden at run time.

- **ID=**"prompt-ID"
  specifies the ID of the prompt to insert.

- **INTEGER_ONLY=**YES | NO
  specifies whether the prompt values have to be of integer type.

- **MAX_DECIMAL=**maximum-decimal-places
  specifies the maximum number of decimal places that are allowed in an unformatted prompt value.

- **MAX_VALUE_COUNT=**maximum-value-count
  specifies the maximum number of selections that the user can specify at a time.

- **MAX_VALUE=**maximum-value
  specifies the maximum value allowed for an unformatted prompt value.

- **MIN_DECIMAL=**minimum-decimal-places
  specifies the minimum number of decimal places that are required in an unformatted prompt value.

- **MIN_VALUE_COUNT=**minimum-value-count
specifies the minimum number of selections that the user needs to make at the prompt.

MIN_VALUE=minimum-value
specifies the minimum value allowed for an unformatted prompt value.

NAME="prompt-name"
specifies the name of a prompt to insert into the current information map.

READ_ONLY=YES | NO
specifies whether the prompt is read-only.

SELECTION_TYPE=SINGLE | MULTIPLE | MULTIPLE_ORDERED
specifies whether the user can make multiple selections from the prompt.

SORT=NONE | ASCENDING_LABEL | DESCENDING_LABEL | ASCENDING_VALUE | DESCENDING_VALUE
specifies how the list of available prompt values is sorted for display.

VALUE_DISPLAYED=YES | NO
specifies whether formatted values are appended to unformatted values in the prompt list.

Required Arguments
The INSERT NUMERIC_PROMPT statement must include one of the COLUMN=, DATAITEM=, or VALUES= arguments:

COLUMN="relational-table-path":"column-name-1"
<LABEL_COLUMN="column-name-2" <FORMAT="format-name">>
specifies that the prompt uses columns from an external table source to dynamically generate values.

relational-table-path
specifies the physical path of a relational table where the specified columns are defined.

column-name-1
specifies the table column to be used for generating unformatted values. The value must be the SAS name of a column defined in the specified relational table.

The DISTINCT= option indicates whether the prompt values derived from the specified column are distinct.

column-name-2
specifies the table column to be used for generating formatted values. If column-name-2 is not specified, then the column specified in column-name-1 is used to generate the formatted values.

You can specify a format to apply to values from the column specified in column-name-2.

Interaction
You can specify only one of the COLUMN=, DATAITEM=, and VALUES= options in an INSERT NUMERIC_PROMPT statement.

Example
insert numeric_prompt
displayed_text="Make one or two selections:"
id="numPromptWithColumn"
value_displayed=yes
sort=ascending_label
selection_type=multiple
max_value_count=2
allow_user_values=yes
DATAITEM="data-item-ID-1" <LABEL_DATAITEM="data-item-ID-2" /> FORMATTING="format-name">

specifies that the prompt uses data items from the current information map to dynamically generate values.

- data-item-ID-1
  specifies the data item used for generating unformatted values.

- data-item-ID-2
  specifies the data item to be used for generating formatted values. If data-item-ID-2 is not specified, then the data item specified in data-item-ID-1 is used to generate the formatted values.

You can specify a format to apply to values from the data item specified in data-item-ID-2.

Interaction

You can specify only one of the COLUMN=, DATAITEM=, and VALUES= options in an INSERT NUMERIC_PROMPT statement.

Example

```
insert numeric_prompt
  displayed_text="Select Customers:"
  id="numPromptWithDataItem"
  value_displayed=yes
  selection_type=multiple
  allow_user_values=yes
  allow_missing_values=yes
  dataitem="di_t1custID" label_dataItem="di_t1amount"
  default_values=(50 73.0 .a .m);
```

DISPLAYED_TEXT="displayed-text"

specifies the text that end users see when they are prompted for a value.

VALUES=USER | custom-values-list

- USER
  specifies that the user must enter a value in response to this prompt.

- custom-values-list
  defines a custom list of values for the prompt to display. The form of the custom-values-list is

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

  where

  - unformatted-value
    specifies the unformatted value for a report. This must be an integer value unless the INTEGER_ONLY= option is disabled.

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

```sql
insert numeric_prompt
displayed_text="Select Customer:" id="numPromptWithStaticValues"
value_displayed=yes
allow_user_values=yes
allow_missing_values=yes
values=((886 "TAIWAN" ) {a "missing A"} { "missing ."} 
(755 "JAPAN" ) { 666 "KOREA"} (400))
default_values=(886);
```

Default USER

Interaction

You can specify only one of the COLUMN=, DATAITEM=, and VALUES= options in an INSERT NUMERIC_PROMPT statement.

Note

To refer to a custom value in the DEFAULT_VALUES= option, you must specify the unformatted value rather than the formatted value.

Optional Arguments

**ALLOW_ALL_VALUES=YES | NO**

specifies whether the "all possible values" special values are supported in the prompt. If enabled, the user is allowed to set the default values of a prompt to _ALL_. This adds all possible values to the list of prompt values that the end user sees at run time. This special value is typically used when you have other prompts that are dependent on this prompt. At run time, if the end user selects this special value for the prompt, then the prompt processes all of the defined values.

Default NO

Restriction

This option does not apply to a single-value prompt or to a multiple-value prompt when the method for populating prompt is “Users enter values”.

**ALLOW_MISSING_VALUES=YES | NO**

specifies whether SAS numeric missing values are supported in the prompt.

Default NO

**ALLOW_USER_VALUES=YES | NO**

specifies whether the user can specify additional unformatted values at run time that are not included when the prompt is created.

Default NO

**DEFAULT_VALUES= _ALL_ | (numeric-value-1<... numeric-value-n>)**

specifies the default prompt values. The values must be integer unless the INTEGER_ONLY option is disabled. The values can be missing if ALLOW_MISSING_VALUES= option is enabled. The number of specified default values must not be greater than the value specified in the MAX_VALUE_COUNT= option nor less than the value specified in the MIN_VALUE_COUNT= option.

The _ALL_ value specifies the all possible values. This argument does not apply unless the ALLOW_ALL_VALUES= option is enabled.
<table>
<thead>
<tr>
<th>Alias</th>
<th>DEFAULT_VALUE=</th>
</tr>
</thead>
</table>

**Interaction** You cannot create a prompted filter if the referenced prompt does not have a default value unless the VERIFY= option associated with the current information map is disabled.

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

<table>
<thead>
<tr>
<th>Alias</th>
<th>DESC=</th>
</tr>
</thead>
</table>

**HIDDEN=YES | NO**
specifies whether the prompt is hidden at run time.

**Default** NO

**ID="prompt-ID"**
specifies the ID of the prompt to insert. If you do not specify an ID, the INFOMAPS procedure generates a unique ID from the prompt name.

**Restrictions**
The ID value cannot contain nulls and non-blank nonprintable characters. 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.

**INTEGER_ONLY=YES | NO**
specifies whether the prompt values have to be of integer type.

**Default** YES

**MAX_DECIMAL=maximal-decimal-places**
specifies the maximum number of decimal places that are allowed in an unformatted prompt value.

**Restriction** This option does not apply if the prompt allows integer values only.

**MAX_VALUE=maximal-value**
specifies the maximum value allowed for an unformatted prompt value. This must be an integer value unless the INTEGER_ONLY= option is disabled.

**MAX_VALUE_COUNT=maximal-value-count**
specifies the maximum number of selections that the user can specify at a time.

**MIN_DECIMAL=minimum-decimal-places**
specifies the minimum number of decimal places that are required in an unformatted prompt value.

**Restriction** This option does not apply if the prompt allows integer values only.
MIN_VALUE=minimum-value
specifies the minimum value allowed for an unformatted prompt value. This must be an integer value unless the INTEGER_ONLY= option is disabled.

MIN_VALUE_COUNT=minimum-value-count
specifies the minimum number of selections that the user needs to make at the prompt.

NAME="prompt-name"
specifies the name of a prompt to insert into the current information map. If the NAME= option is not specified, the INFOMAPS procedure generates a default name.

Restriction  The prompt name cannot contain nulls or non-blank nonprintable characters.

READ_ONLY=YES | NO
specifies whether the prompt is read-only.

Default  NO

SELECTION_TYPE=SINGLE | MULTIPLE | MULTIPLE_ORDERED
specifies whether the user can make multiple selections from the prompt.

SINGLE
specifies that the user can select only one value from the prompt.

MULTIPLE
specifies that the user can select multiple values from the prompt.

MULTIPLE_ORDERED
specifies that the user can select multiple values from the prompt and that the selected values are ordered.

Default  MULTIPLE if either the MAXIMUM_VALUE_COUNT= or MINIMUM_VALUE_COUNT= options specify a value greater than 1 or if more than one value is specified for the DEFAULT_VALUES= option; otherwise, SINGLE.

SORT=NONE | ASCENDING_LABEL | DESCENDING_LABEL | ASCENDING_VALUE | DESCENDING_VALUE
specifies how the list of available prompt values is sorted for display.

NONE
specifies that the prompt values are not sorted.

ASCENDING_LABEL
sorts the prompt list in ascending order based on the formatted label values.

DESCENDING_LABEL
sorts the prompt list in descending order based on the formatted label values.

ASCENDING_VALUE
sorts the prompt list in ascending order based on the unformatted values.

DESCENDING_VALUE
sorts the prompt list in descending order based on the unformatted values.

Default  NONE

Restriction  This option applies only when the COLUMN= or DATAITEM= option is used. In that case, the prompt list is dynamic.
VALUE_DISPLAYED=YES | NO
specifies whether formatted values are appended to unformatted values in the
prompt list.

Default NO

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**

This 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=**

This 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 fails. Therefore, you should 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=**

This specifies the description of the relationship, which can be viewed by the information map consumer.

**Alias**

**DESC=**

**ID=**

This 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

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

---

**INSERT TEXT_PROMPT Statement**

Inserts a text prompt into the current information map.

**Syntax**

```plaintext
INSERT TEXT_PROMPT DISPLAYED_TEXT="displayed-text"
COLUMN="relational-table-path".column-name <options>;

INSERT TEXT_PROMPT DISPLAYED_TEXT="displayed-text"
DATAITEM="data-item-ID" <options>;

INSERT TEXT_PROMPT DISPLAYED_TEXT="displayed-text"
VALUES=USER | custom-values-list <options>;
```
Summary of Optional Arguments

ALLOW_ALL_VALUES=YES | NO
specifies whether the "all possible values" special values are supported in the prompt.

ALLOW_MISSING_VALUES=YES | NO
specifies whether SAS numeric missing values are supported in the prompt.

ALLOW_USER_VALUES=YES | NO
specifies whether the user can specify additional unformatted values at run time.

DEFAULT_VALUES=_ALL_ | (value-1<... value-n>)
specifies the default prompt values.

DESCRIPTION="descriptive-text"
specifies the description of the text prompt.

HIDDEN=YES | NO
specifies whether the prompt is hidden at run time.

ID="prompt-ID"
specifies the ID of the prompt to insert.

MAX_LENGTH=maximum-string-length
specifies the maximum number of characters that are allowed in an unformatted prompt value.

MAX_VALUE_COUNT=maximum-value-count
specifies the maximum number of selections that the user can specify at a time.

MIN_LENGTH=minimum-string-length
specifies the minimum number of characters that are required in an unformatted prompt value.

MIN_VALUE_COUNT=minimum-value-count
specifies the minimum number of selections that the user needs to make at the prompt.

NAME="prompt-name"
specifies the name of a prompt to insert into the current information map.

READ_ONLY=YES | NO
specifies whether the prompt is read-only.

SELECTION_TYPE=SINGLE | MULTIPLE | MULTIPLE_ORDERED
specifies whether the user can make multiple selections from the prompt.

SORT=NONE | ASCENDING_LABEL | DESCENDING_LABEL | ASCENDING_VALUE | DESCENDING_VALUE
specifies how the list of available prompt values is sorted for display.

VALUE_DISPLAYED=YES | NO
specifies whether formatted values are appended to unformatted values in the prompt list.

Required Arguments

The INSERT TEXT_PROMPT statement must include one of the COLUMN=, DATAITEM=, or VALUES= arguments:

COLUMN="relational-table-path":"column-name-1"
<LABEL_COLUMN="column-name-2" <FORMAT="format-name">>
specifies that the prompt uses columns from an external table source to dynamically generate values.
relational-table-path
specifies the physical path of a relational table where the specified columns are defined.

column-name-1
specifies the table column to be used for generating unformatted values. The value must be the SAS name of a column defined in the specified relational table.

The DISTINCT= option indicates whether the prompt values derived from the specified column are distinct.

column-name-2
specifies the table column to be used for generating formatted values. If column-name-2 is not specified, then the column specified in column-name-1 is used to generate the formatted values.

You can specify a format to apply to values from the column specified in column-name-2.

Interaction
You can specify only one of the COLUMN=, DATAITEM=, and VALUES= options in an INSERT TEXT_PROMPT statement.

Example
insert text_prompt
displayed_text="Select countries that apply:"
id="countryPrompt"
value_displayed=yes
allow_user_values=yes
column="/Shared Data/CUSTOMER"."Country"
label_column="Postal_Code" /format="custom-format"
default_values=('CANADA' 'GERMANY');

DATAITEM="data-item-ID-1" <LABEL_DATAITEM="data-item-ID-2" />
FORMAT="format-name"

specifies that the prompt uses data items from the current information map to dynamically generate values.

data-item-ID-1
specifies the data item used for generating unformatted values.

data-item-ID-2
specifies the data item to be used for generating formatted values. If data-item-ID-2 is not specified, then the data item specified in data-item-ID-1 is used to generate the formatted values.

You can specify a format to apply to values from the data item specified in data-item-ID-2.

Interaction
You can specify only one of the COLUMN=, DATAITEM=, and VALUES= options in an INSERT TEXT_PROMPT statement.

Example
insert text_prompt
displayed_text="Select one country:"
id="countryPrompt"
value_displayed=yes
allow_user_values=yes
dataitem="Country" label_dataitem="Postal_Code"
/format="custom-format"
default_values=('GERMANY');
DISPLAYED_TEXT="displayed-text"
specifies the text that end users see when they are prompted for a value.

VALUES=USER | custom-values-list

USER
specifies that the user must enter a value in response to this prompt.

custom-values-list
defines a custom list of values for the prompt to display. The form of the
custom-values-list is

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

where

unformatted-value
specifies the unformatted value for a report.

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

Example

```
insert text_prompt
displayed_text="Select at least one country:"
id="promptWithStaticValues"
value_displayed=yes
allow_user_values=yes
values=(("TAIWAN" "TW800") ("JAPAN" "JA100")
 ("KOREA") ("CANADA") ("CHINA" "ZH200"))
default_values=(("TAIWAN" "CANADA"));
```

Default

USER

Interaction

You can specify only one of the COLUMN=, DATAITEM=, and
VALUES= options in an INSERT TEXT_PROMPT statement.

Note

To refer to a custom value in the DEFAULT_VALUES= option, you
must specify the unformatted value rather than the formatted value.

Optional Arguments

ALLOW_ALL_VALUES=YES | NO

specifies whether the “all possible values” special values are supported in the
prompt. If enabled, the user is allowed to set the default values of a prompt to
_ALL_. This adds (all possible values) to the list of prompt values that the
end user sees at run time. This special value is typically used when you have
other prompts that are dependent on this prompt. At run time, if the end user
selects this special value for the prompt, then the prompt processes all of the
defined values.

Default

NO
Restriction  This option does not apply to a single-value prompt or to a multiple-value prompt when the method for populating prompt is “Users enter values”.

**ALLOW_MISSING_VALUES=**YES | NO  
specifies whether SAS numeric missing values are supported in the prompt.

If the ALLOW_MISSING_VALUES= option is set to YES, then _BLANK_ or "_BLANK_" is treated as a missing value in the DEFAULT_VALUES= or VALUES= arguments. If the ALLOW_MISSING_VALUES= option is set to NO, then _BLANK_ or "_BLANK_" is treated as a string value.

Default  NO

**ALLOW_USER_VALUES=**YES | NO  
specifies whether the user can specify additional unformatted values at run time that are not included when the prompt is created.

Default  NO

**DEFAULT_VALUES=**_ALL_ | (value-1<… value-n>)  
specifies the default prompt values. The values can be missing if ALLOW_MISSING_VALUES= option is enabled. The number of specified default values must not be greater than the value specified in the MAX_VALUE_COUNT= option nor less than the value specified in the MIN_VALUE_COUNT= option.

The _ALL_ value specifies the all possible values. This argument does not apply unless the ALLOW_ALL_VALUES option is enabled.

If the ALLOW_MISSING_VALUES= option is enabled, then _BLANK_ or "_BLANK_" is treated as a missing value. Otherwise, _BLANK_ or "_BLANK_" is treated as a string value.

Alias  DEFAULT_VALUE=

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

Alias  DESC=

**HIDDEN=**YES | NO  
specifies whether the prompt is hidden at run time.

Default  NO

**ID=**"prompt-ID"  
specifies the ID of the prompt to insert. If you do not specify an ID, the INFOMAPS procedure generates a unique ID from the prompt name.

Restrictions  The ID value cannot contain nulls and non-blank nonprintable characters. 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.

**MAX_LENGTH=** *maximum-string-length*
specifies the maximum number of characters that are allowed in an unformatted prompt value.

Trailing blanks are included when the length of a prompt value is calculated and validated.

**Default**  2,147,483,647

**MAX_VALUE_COUNT=** *maximum-value-count*
specifies the maximum number of selections that the user can specify at a time.

**MIN_LENGTH=** *minimum-string-length*
specifies the minimum number of characters that are required in an unformatted prompt value.

Trailing blanks are included when the length of a prompt value is calculated and validated.

**Default**  1

**MIN_VALUE_COUNT=** *minimum-value-count*
specifies the minimum number of selections that the user needs to make at the prompt.

**NAME=** "*prompt-name*"
specifies the name of a prompt to insert into the current information map. If the NAME= option is not specified, the INFOMAPS procedure generates a default name.

**Restriction** The prompt name cannot contain nulls or non-blank nonprintable characters.

**READ_ONLY=** *YES* | *NO*
specifies whether the prompt is read-only.

**Default**  *NO*

**SELECTION_TYPE=** *SINGLE* | *MULTIPLE* | *MULTIPLE_ORDERED*
specifies whether the user can make multiple selections from the prompt.

**SINGLE**
specifies that the user can select only one value from the prompt.

**MULTIPLE**
specifies that the user can select multiple value from the prompt.

**MULTIPLE_ORDERED**
specifies that the user can select multiple value from the prompt and that the selected values are ordered.
SORT=NONE | ASCENDING_LABEL | DESCENDING_LABEL | ASCENDING_VALUE | DESCENDING_VALUE
specifies how the list of available prompt values is sorted for display.

NONE
specifies that the prompt values are not sorted.

ASCENDING_LABEL
sorts the prompt list in ascending order based on the formatted label values.

DESCENDING_LABEL
sorts the prompt list in descending order based on the formatted label values.

ASCENDING_VALUE
sorts the prompt list in ascending order based on the unformatted values.

DESCENDING_VALUE
sorts the prompt list in descending order based on the unformatted values.

Default
NONE
Restriction
This option applies only when the COLUMN= or DATAITEM= option is used. In that case, the prompt list is dynamic.

VALUE_DISPLAYED=YES | NO
specifies whether formatted values are appended to unformatted values in the prompt list.

Default
NO

INSERT_TIME_PROMPT Statement
Inserts a time prompt into the current information map.

Syntax

INSERT TIME_PROMPT DISPLAYED_TEXT="displayed-text" <options>;

Summary of Optional Arguments

ALLOW_MISSING_VALUES=YES | NO
specifies whether SAS numeric missing values are supported in the prompt.

DEFAULT_VALUE="SAS-time-value"t | "time-string" | "relative-time"
specifies the default time prompt value.

DESCRIPTION="descriptive-text"
specifies the description of the time prompt.
HIDDEN=YES | NO
  specifies whether the prompt is hidden at run time.

ID="prompt-ID"
  specifies the ID of the prompt to insert.

MAX_VALUE="SAS-time-value" | "time-string" | "relative-time"
  specifies the maximum time value allowed for an unformatted prompt value.

MIN_VALUE="SAS-time-value" | "time-string" | "relative-time"
  specifies the minimum time value allowed for an unformatted prompt value.

NAME="prompt-name"
  specifies the name of a prompt to insert into the current information map.

READ_ONLY=YES | NO
  specifies whether the prompt is read-only.

Required Argument

DISPLAYED_TEXT="displayed-text"
  specifies the text that end users see when they are prompted for a value.

Optional Arguments

ALLOW_MISSING_VALUES=YES | NO
  specifies whether SAS numeric missing values are supported in the prompt.

  Default NO

DEFAULT_VALUE="SAS-time-value" | "time-string" | "relative-time"
  specifies the default time prompt value. The value can be missing if the ALLOW_MISSING_VALUES= option is enabled.

You can specify a time value in the following ways:

- Enter a SAS time value (for example, "03:20:20").
- Enter a quoted string with a time value (for example, "1:01 AM").
- Enter a quoted string indicating a relative time. For example, "t0m" represents the current time, "m0m" represents the current minute, "H-1H" represents the previous hour, and "t-1HBH" represents the beginning of previous hour.

  Requirement At least one default value is required if either the HIDDEN= option or the READ_ONLY= option is enabled.

  Interaction You cannot create a prompted filter if the referenced prompt does not have a default value unless the VERIFY= option associated with the current information map is disabled.

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

  Alias DESC=

HIDDEN=YES | NO
  specifies whether the prompt is hidden at run time.
**ID=“prompt-ID”**
specifies the ID of the prompt to insert. If you do not specify an ID, the
INFOMAPS procedure generates a unique ID from the prompt name.

**Restrictions**
The ID value cannot contain nulls and non-blank nonprintable
characters. 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.

**MAX_VALUE=“SAS-time-value” | “time-string” | “relative-time”**
specifies the maximum time value allowed for an unformatted prompt value. You
can specify a time value in the following ways:

- Enter a SAS time value (for example, "03:20:20").
- Enter a quoted string with a time value (for example, "1:01 AM").
- Enter a quoted string indicating a relative time. For example, "t0m" represents the current time, "m0m" represents the current minute, "H-1H" represents the previous hour, and "t-1HBH" represents the beginning of previous hour.

When you specify a relative value as the maximum value, it is resolved to the
real-time value to help you determine whether you correctly interpreted the
meaning of the relative value. For example, if you are editing the prompt at
12:00:00 and you specify the relative value Next minute as the maximum
value, then the value is mapped to 12:01.

**MIN_VALUE=“SAS-time-value” | “time-string” | “relative-time”**
specifies the minimum time value allowed for an unformatted prompt value. You
can specify a time value in the following ways:

- Enter a SAS time value (for example, "03:20:20").
- Enter a quoted string with a time value (for example, "1:01 AM").
- Enter a quoted string indicating a relative time. For example, "t0m" represents the current time, "m0m" represents the current minute, "H-1H" represents the previous hour, and "t-1HBH" represents the beginning of previous hour.

**NAME=“prompt-name”**
specifies the name of a prompt to insert into the current information map. If the
NAME= option is not specified, the INFOMAPS procedure generates a default
name.

**Restriction**
The prompt name cannot contain nulls or non-blank nonprintable
characters.

**READ_ONLY=YES | NO**
specifies whether the prompt is read-only.
INSERT_TIMESTAMP_PROMPT Statement

Inserts a time prompt into the current information map.

Syntax

```
INSERT_TIMESTAMP_PROMPT DISPLAYED_TEXT="displayed-text" <options>;
```

Summary of Optional Arguments

- **ALLOW_MISSING_VALUES=** YES | NO
  - Specifies whether SAS numeric missing values are supported in the prompt.

- **DEFAULT_VALUE=** "SAS-timestamp-value"dt | "timestamp-string" | "relative-time"
  - Specifies the default time prompt value.

- **DESCRIPTION=** "descriptive-text"
  - Specifies the description of the timestamp prompt.

- **HIDDEN=** YES | NO
  - Specifies whether the prompt is hidden at run time.

- **ID=** "prompt-ID"
  - Specifies the ID of the prompt to insert.

- **MAX_VALUE=** "SAS-timestamp-value"dt | "timestamp-string" | "relative-time"
  - Specifies the maximum time value allowed for an unformatted prompt value.

- **MIN_VALUE=** "SAS-timestamp-value"dt | "timestamp-string" | "relative-time"
  - Specifies the minimum time value allowed for an unformatted prompt value.

- **NAME=** "prompt-name"
  - Specifies the name of a prompt to insert into the current information map.

- **READ_ONLY=** YES | NO
  - Specifies whether the prompt is read-only.

Required Argument

- **DISPLAYED_TEXT=** "displayed-text"
  - Specifies the text that end users see when they are prompted for a value.

Optional Arguments

- **ALLOW_MISSING_VALUES=** YES | NO
  - Specifies whether SAS numeric missing values are supported in the prompt.

  Default: NO
DEFAULT_VALUE="SAS-timestamp-value"dt | "timestamp-string" | "relative-time"
specifies the default time prompt value. The value can be missing if the
ALLOW_MISSING_VALUES= option is enabled.

You can specify a timestamp value in the following ways:
- Enter a SAS timestamp value (for example, "14FEB2020:11:00:00"dt).
- Enter a quoted string with a time value (for example, "Dec 25, 2020 12:00:00 AM").
- Enter a quoted string indicating a relative time. For example, "t0m" represents the current time, "m0m" represents the current minute, "H-1H" represent the previous hour, and "t-1HBH" represents the beginning of previous hour.

Requirement At least one default value is required if either the HIDDEN= option or the READ_ONLY= option is enabled.

Interaction You cannot create a prompted filter if the referenced prompt does not have a default value unless the VERIFY= option associated with the current information map is disabled.

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

Alias DESC=

HIDDEN=YES | NO
specifies whether the prompt is hidden at run time.

Default NO

ID="prompt-ID"
specifies the ID of the prompt to insert. If you do not specify an ID, the INFOMAPS procedure generates a unique ID from the prompt name.

Restrictions The ID value cannot contain nulls and non-blank nonprintable characters. 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.

MAX_VALUE="SAS-timestamp-value"dt | "timestamp-string" | "relative-time"
specifies the maximum time value allowed for an unformatted prompt value. You can specify a timestamp value in the following ways:
- Enter a SAS timestamp value (for example, "14FEB2020:11:00:00"dt).
- Enter a quoted string with a time value (for example, "Dec 25, 2020 12:00:00 AM").
Enter a quoted string indicating a relative time. For example, "t0m" represents the current time, "m0m" represents the current minute, "H-1H" represents the previous hour, and "t-1HBH" represents the beginning of previous hour.

When you specify a relative value as the maximum value, it is resolved to the real-time value to help you determine whether you correctly interpreted the meaning of the relative value. For example, if you are editing the prompt at 12:00:00 and you specify the relative value `Next minute` as the maximum value, then the value is mapped to 12:01.

**MIN_VALUE=** "SAS-timestamp-value"dt | "timestamp-string" | "relative-time"

specifies the minimum time value allowed for an unformatted prompt value. You can specify a timestamp value in the following ways:

- Enter a SAS timestamp value (for example, "$14FEB2020:11:00:00"dt).
- Enter a quoted string with a time value (for example, "Dec  25,  2020 12:00:00 AM").
- Enter a quoted string indicating a relative time. For example, "t0m" represents the current time, "m0m" represents the current minute, "H-1H" represents the previous hour, and "t-1HBH" represents the beginning of previous hour.

**NAME=** "prompt-name"

specifies the name of a prompt to insert into the current information map. If the NAME= option is not specified, the INFOMAPS procedure generates a default name.

Restriction  The prompt name cannot contain nulls or non-blank nonprintable characters.

**READ_ONLY=** YES | NO

specifies whether the prompt is read-only.

Default  NO

---

## 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

```sas
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 Chapter 7, “Example: Using the INFOMAPS Procedure and the
Information Maps Engine,” on page 143.

Example Code 2.1  LIST Statement Information for Data Sources

<table>
<thead>
<tr>
<th>Total datasources: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data source: SAS Sample Data.EMPINFO</td>
</tr>
<tr>
<td>ID: Empinfo</td>
</tr>
<tr>
<td>Name: EMPINFO</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>Required data source: NO</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Data source: SAS Sample Data.JOBCODES</td>
</tr>
<tr>
<td>ID: Jobcodes</td>
</tr>
<tr>
<td>Name: JOBCODES</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>Required data source: NO</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Example Code 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: ...

Example Code 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")
  ...

Example Code 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

```bash
/* 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

- **ALLOW_DRILL_THROUGH=** *YES | NO*
  specifies whether users can drill down to detail data.

- **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.

- **DATA_LOCALE="** *locale-name*
  specifies a supported locale value that is used as the data locale of the information map being created.

- **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 bytes 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 bytes, which is the maximum length for SAS names.

Optional Arguments

ALLOW_DRILL_THROUGH=YES | NO
specifies whether users can drill down to detail data when the data source for the information map is an OLAP cube.

Default YES

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 fails. Therefore, you should 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.

DATA_LOCALE="locale-name"
   specifies a supported locale value that is used as the data locale of the information map being created. For more information about locale values, see SAS National Language Support (NLS): Reference Guide.

Alias       COMPUTATIONAL_LOCALE=
Default     Current SAS session locale

Note        The data locale of an information map does not have to be the same as the current SAS session locale. However, a warning is issued if the specified locale is not compatible with the current SAS session encoding. A warning about possible data integrity violation is also generated if you change the data locale of an existing information map.

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, multiple consecutive blank spaces within a data item name are replaced with a single blank space, and trailing blank spaces are removed.

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 be 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

```plaintext
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.
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 overwrites 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.

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

1 The MAPPATH specified in the NEW INFOMAP or UPDATE INFOMAP statement

2 The MAPPATH specified in the PROC INFOMAPS statement

Alias LOCATION=

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 104.

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

If 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

ALLOW_DRILL_THROUGH=YES | NO

specifies whether users can drill down to detail data.

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.

DATA_LOCALE="locale-name"

specifies a supported locale value that is used as the data locale of the information map being 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

ALLOW_DRILL_THROUGH=YES | NO
  specifies whether users can drill down to detail data when the data source for the information map is an OLAP cube.
  
  Default  YES

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 fails. Therefore, you should add a prefix or suffix to the property name to ensure uniqueness.
**UPDATE CURRENT_INFOMAP Statement 85**

- **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.

**DATA_LOCALE="locale-name"**
- specifies a supported locale value that is used as the data locale of the information map being updated. For more information about locale values, see *SAS National Language Support (NLS): Reference Guide*.

  **Alias**
  - COMPUTATIONAL_LOCALE=
  
  **Default**
  - Current SAS session locale

  **Note**
  - The data locale of an information map does not have to be the same as the current SAS session locale. However, a warning is issued if the specified locale is not compatible with the current SAS session encoding. A warning about possible data integrity violation is also generated if you change the data locale of an existing information map.

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

  **Alias**
  - DESC=

**INIT_CAP=**
- 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 DATAIMPORT 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 be 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"
    requireddatasources=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 filters can be applied to members of OLAP data items that represent hierarchies. 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 25 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, \(\text{SUM}(\text{COL1})/\text{COUNT}(\text{COL1})\) 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, \(\text{SUM}(\text{COL1} \cdot \text{COL2})\) 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_` 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` 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 fails. Therefore, you should 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, Information Maps Engine Statements" on page 124. For more information about the EXPCOLUMNLEN= data set option, see "EXPCOLUMNLEN= Data Set Option" on page 134.

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

Specify a blank value for the format-name argument to remove the current format that is applied to 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 requires its user to manually enter 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 list 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" "Virgina")
  ("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, which can be viewed by the information map consumer.

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"
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">"

    (Optional)
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 fails. Therefore, you should 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, which can be viewed by the information map
  consumer.

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 fails. Therefore, you should 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, which can be viewed by the information map consumer.

**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. An example is 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. An example is 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

ALLOW_DRILL THROUGH=YES | NO
specifies whether users can drill down to detail data.

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.

DATA_LOCALE="locale-name"
specifies a supported locale value that is used as the data locale of the information map being updated.

DESCRIPTION="descriptive-text"
specifies the description of an information map.

INIT_CAP=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=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, then an error occurs.

**Optional Arguments**

**ALLOW_DRILL_THROUGH**=**YES** | **NO**
specifies whether users can drill down to detail data when the data source for the information map is an OLAP cube.

**Default**
**YES**

**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.
Property names must be unique. If a specified property name already exists in an information map, then the UPDATE INFOMAP statement fails. Therefore, you should 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)

### DATA_LOCALE="locale-name"
- specifies a supported locale value that is used as the data locale of the information map being updated. For more information about locale values, see SAS National Language Support (NLS): Reference Guide.

**Alias**
COMPUTATIONAL_LOCALE=

**Default**
Current SAS session locale

**Note**
The data locale of an information map does not have to be the same as the current SAS session locale. However, a warning is issued if the specified locale is not compatible with the current SAS session encoding. A warning about possible data integrity violation is also generated if you change the data locale of an existing information map.

### DESCRIPTION="descriptive-text"
- specifies the description of an 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.
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 be 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

```plaintext
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. Authorization-based prefilters for a specific user or group can also be assigned.

**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. However, 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 81.
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 fails. Therefore, you should 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, which can be viewed by the information map consumer.

**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**

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

**Example 2**

```sql
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

```plaintext
proc infomaps 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
```

/* Create a filter that can be used in queries to subset a data item */
insert filter"
   set=EmployeeInfo"
   logic="Name eq 'John Doe'"
   id="John Doe Filter";
/* Insert data items into the folders */
insert dataitem"
   id="EmployeeInfo"
   folder="Employee Info"
   mappath="/Users/sasdemo/My Folder"
   auto_replace=yes;
/* Insert data items into the Salary Information table */
insert dataitem"
   id="SalaryInfo"
   folder="Salary Info"
   mappath="/Users/sasdemo/My Folder"
   auto_replace=yes;
```
Example 2: Creating an Information Map with Relationships and Filters

```plaintext
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. */
```
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:

```sas
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";
```

The following window shows the resulting information map opened in SAS Information Map Studio.
Example 3: Aggregating a Data Item

/* 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.
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=XXXXXXXXXX
               metaserver="myserver.mycompany.com"
               metaport=8561
               mappath="/Users/myUserID/My Folder";
```

**Example Code 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 includes 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 are not 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;
/* Run the Information Maps engine to retrieve the data. */
libname mymaps infomaps metauser=myUserID
metapass=XXXXXXXXXX
metaserver="myserver.mycompany.com"
metaport=8561
mappath="/Users/myUserID/My Folder";

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

/* Display a list of available information maps. */
proc datasets lib=mymaps;
  Directory
    Libref  MYMAPS
    Engine  INFOMAPS
    Physical Name /Users/myUserID/My Folder
  Member
    #  Name                        Type
      1  Employee Statistics Sample  DATA
run;
quit;

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

/* 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;

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

/* 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;

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 INFORMAPS Indexes 0
Created . Observation Length 0
Last Modified . Deleted Observations 0
Protection Compressed NO
Data Set Type Sorted NO
Label Filters 1
Data Representation Default
Encoding Default

Alphabetic List of Variables and Attributes

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

Information Maps

FilterName FilterType FilterDesc
Cary Employees Unp Employees who work in Cary, North Carolina.

Number of Years

<table>
<thead>
<tr>
<th>Obs</th>
<th>Name</th>
<th>Salary_2</th>
<th>Salary2</th>
<th>Salary3</th>
<th>Salary4</th>
<th>Hire Date</th>
<th>Employed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryan, Lynne C.</td>
<td>$185,000.00</td>
<td>$185,000.00</td>
<td>$185,000.00</td>
<td>$185,000.00</td>
<td>08APR1984</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>Fissel, Ronald T.</td>
<td>$85,000.00</td>
<td>$85,000.00</td>
<td>$85,000.00</td>
<td>$85,000.00</td>
<td>02FEB1985</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>White, Frank P.</td>
<td>$69,000.00</td>
<td>$69,000.00</td>
<td>$69,000.00</td>
<td>$69,000.00</td>
<td>01JUN1984</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>Winfree, Ambrose Y.</td>
<td>$100,000.00</td>
<td>$100,000.00</td>
<td>$100,000.00</td>
<td>$100,000.00</td>
<td>14JUN1989</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Blue, Kenneth N.</td>
<td>$18,000.00</td>
<td>$18,000.00</td>
<td>$18,000.00</td>
<td>$18,000.00</td>
<td>12NOV1991</td>
<td>17</td>
</tr>
</tbody>
</table>

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 135. 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 139.

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 must 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.
LIBNAME Statement for the Information Maps Engine

Using the LIBNAME Statement .................................................. 123
Dictionary ................................................................. 124
   LIBNAME Information Maps Engine, Statement .................. 124
Examples: ........................................................................ 130
   Submitting a LIBNAME Statement Using the Defaults ............ 130
   Submitting a LIBNAME Statement Using Connection Options ...... 130

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 is a multi-user server that stores metadata from one or more metadata repositories. The metadata server 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

Information Maps Engine 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. An example is 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*.

**Aliases**  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*.

**Aliases**  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.

Aliases

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. An example is 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 133.

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 134.

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 136.

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.
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.

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

**Example Code 4.1  Log for the LIBNAME Statement**

```plaintext
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:

```plaintext
libname mylib infomaps metauser='myUserID'
  metapass=myPassword
  metaserver="myserver.mycompany.com"
  metaport=8561
  mappath="/Users/myUserID/My Folder";
```
Example Code 4.2  Log for the LIBNAME Statement

```plaintext
libname mylib infomaps metauser='myUserID'
     metapass=XXXXXXXXXX
     metaserver=myserver.mycompany.com
     metaport=8561
     mappath="/Users/myUserID/My Folder"

NOTE: Libref MYLIB was successfully assigned as follows:
   Engine:          INFOMAPS
   Physical Name: /Users/myUserID/My Folder
```
SAS Data Set Options for the Information Maps Engine

Using Data Set Options

Data set options specify actions that apply only to the SAS data set with which they appear. 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
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 satisfies 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.

```sas
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.
### READBUFSIZE= Data Set Option

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA Step, LIBNAME Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Data Set Control</td>
</tr>
<tr>
<td>Alias:</td>
<td>BUFSIZE=</td>
</tr>
<tr>
<td>Default:</td>
<td>1000</td>
</tr>
</tbody>
</table>

#### Syntax

```plaintext
READBUFSIZE=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 READBUFSIZE= 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 READBUFSIZE= option use more memory. In addition, setting a high value for the READBUFSIZE= option could yield stale data if the data source is updated frequently.

#### See Also

*READBUFSIZE= option in the LIBNAME statement for the Information Maps engine*
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 20.

- 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 does 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 can 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 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 is not 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 that the relevant memory settings are 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. However, 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 truncated to a unique 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 127.

- 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)"
(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.
Example: Using the INFOMAPS Procedure and the Information Maps Engine

About This Example

The example in this chapter shows you how to use the INFOMAPS procedure to create a new information map and then use the Information Maps engine to retrieve the data associated with the new information map. Once you have the data, you can use 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\9.4\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:

```sas
/* 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"
```
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:

```sas
/* 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;
```

Example Code 7.1  Setting the Metadata System Options and the Macro Variable

```sas
/* 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="myUserID"
    metapass=XXXXXXXX
    metaserver="myserver.mycompany.com"
    metaport=8561;
```
Example Code 7.2  Log for the METALIB Procedure

```sas
/* 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;

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

<table>
<thead>
<tr>
<th>Metadata Summary Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total tables analyzed</td>
</tr>
<tr>
<td>Tables Updated</td>
</tr>
<tr>
<td>Tables Added</td>
</tr>
<tr>
<td>Tables matching data source</td>
</tr>
<tr>
<td>Tables not found</td>
</tr>
<tr>
<td>Tables not processed</td>
</tr>
</tbody>
</table>

-----------------------------------------------

Tables Added

<table>
<thead>
<tr>
<th>Metadata Name</th>
<th>Metadata ID</th>
<th>SAS Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPINFO</td>
<td>ASY110V0.BH000028</td>
<td>empinfo</td>
</tr>
<tr>
<td>JOBCODES</td>
<td>ASY110V0.BH000029</td>
<td>jobcodes</td>
</tr>
<tr>
<td>SALARY</td>
<td>ASY110V0.BH00002A</td>
<td>salary</td>
</tr>
</tbody>
</table>
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:

```sas
/* 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";
```

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, then your output will be different from what is shown.
Example Code 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";
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;
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";
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>>)";
71
72  NOTE: A relationship between the data sources "Empinfo" and "Jobcodes" has been
73    inserted. The relationship's ID is "JOIN_10".
insert relationship
  
  left_table="Empinfo"
  right_table="Salary"
  join=inner
  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".

/* 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";

NOTE: A data item was inserted for the physical column Salary.ENDDATE. The data item's
ID is "Enddate".

/* 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;

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>>)
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:

```sas
/* 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.

Example Code 7.4 Log for the Information Maps Engine LIBNAME Statement

```sas
/* 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: 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:

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

Example Code 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
Enciding             Default

Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Annual Salary</td>
<td>Num</td>
<td>8</td>
<td>DOLLAR12.</td>
<td>Physical column SALARY</td>
</tr>
<tr>
<td>5</td>
<td>Dept_code</td>
<td>Char</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Division</td>
<td>Char</td>
<td>40</td>
<td></td>
<td>Physical column DIVISION</td>
</tr>
<tr>
<td>9</td>
<td>Enddate</td>
<td>Num</td>
<td>8</td>
<td>DATE9.</td>
<td>Physical column ENDDATE</td>
</tr>
<tr>
<td>4</td>
<td>Identification Number</td>
<td>Num</td>
<td>8</td>
<td>SSN11.</td>
<td>Physical column IDNUM</td>
</tr>
<tr>
<td>1</td>
<td>Jobcode</td>
<td>Char</td>
<td>8</td>
<td></td>
<td>Physical column JOBCODE</td>
</tr>
<tr>
<td>2</td>
<td>Location</td>
<td>Char</td>
<td>8</td>
<td></td>
<td>Physical column LOCATION</td>
</tr>
<tr>
<td>8</td>
<td>Monthly Salary</td>
<td>Num</td>
<td>8</td>
<td>DOLLAR12.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Title</td>
<td>Char</td>
<td>20</td>
<td>$F20.</td>
<td>Physical column TITLE</td>
</tr>
</tbody>
</table>

Information Maps

<table>
<thead>
<tr>
<th>FilterName</th>
<th>FilterType</th>
<th>FilterDesc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status is Current</td>
<td>Unp</td>
<td>Employees in Education and Publications</td>
</tr>
<tr>
<td>Education and Publications</td>
<td>Unp</td>
<td>Employees in Education and Publications</td>
</tr>
<tr>
<td>Host Systems Development</td>
<td>Unp</td>
<td>Employees in Host Systems Development</td>
</tr>
<tr>
<td>Cary HQ</td>
<td>Unp</td>
<td>Located in Cary, North Carolina HQ</td>
</tr>
</tbody>
</table>

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:

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

Example Code 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
```
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"
   (keep="Annual Salary" n Division
    aggregate=yes
    filter=(NOT("Education and Publications" n
            OR "Host Systems Development" n))
   );
run;

Example Code 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" (obs=5);
133 run;

NOTE: There were 5 observations read from the data set HR_DATA.'Employee Info'

NOTE: PROCEDURE PRINT used (Total process time):
      real time   0.44 seconds
      cpu time   0.07 seconds

135 /* Use aggregation and a filter to produce a report of total */
136 /* salary for selected divisions. */
137 title1 "Total salary for each division except PUBS and EDU"
title2 "and the Host Systems division"
proc print data=HR_Data."Employee Info"
   (keep="Annual Salary" n Division
    aggregate=yes
    filter=(NOT("Education and Publications" n
            OR "Host Systems Development" n))
   );
run;

NOTE: There were 14 observations read from the data set HR_DATA.'Employee Info'

NOTE: PROCEDURE PRINT used (Total process time):
      real time   0.42 seconds
      cpu time   0.06 seconds
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 use a DATA step to apply a filter to view only the data for the employees in the Host Systems Development Division. You 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. Finally, a report is produced with ODS (Output Delivery System).

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

```sas
/* 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"n(filter="Host Systems Development"n);
keep jobcode "Annual Salary"n;
run;

/* Produce an ODS report. */
ods html body="example_body.htm";

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

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

Example Code 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";
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

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

#### Output 7.6  Report Displayed in the Results Viewer

```plaintext
Annual Salary by Job Code in Host Systems Development Division

The MEANS Procedure

Analysis Variable : Annual Salary Physical column SALARY

```

<table>
<thead>
<tr>
<th>JOBCODE</th>
<th>N Obs</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSD001</td>
<td>1</td>
<td>1</td>
<td>30000</td>
<td>.</td>
<td>30000</td>
<td>30000</td>
</tr>
<tr>
<td>HSD002</td>
<td>4</td>
<td>4</td>
<td>39625</td>
<td>11940</td>
<td>27000</td>
<td>55000</td>
</tr>
<tr>
<td>HSD003</td>
<td>1</td>
<td>1</td>
<td>29000</td>
<td>.</td>
<td>29000</td>
<td>29000</td>
</tr>
<tr>
<td>HSD004</td>
<td>3</td>
<td>3</td>
<td>47667</td>
<td>20108</td>
<td>31000</td>
<td>70000</td>
</tr>
<tr>
<td>HSD005</td>
<td>2</td>
<td>2</td>
<td>57500</td>
<td>3536</td>
<td>55000</td>
<td>60000</td>
</tr>
<tr>
<td>HSD006</td>
<td>1</td>
<td>1</td>
<td>120000</td>
<td>.</td>
<td>120000</td>
<td>120000</td>
</tr>
<tr>
<td>HSD007</td>
<td>4</td>
<td>4</td>
<td>65750</td>
<td>9777</td>
<td>57000</td>
<td>79000</td>
</tr>
<tr>
<td>HSD008</td>
<td>5</td>
<td>5</td>
<td>61000</td>
<td>18990</td>
<td>45000</td>
<td>93500</td>
</tr>
</tbody>
</table>
```
Using SQL DICTIONARY Tables

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

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

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

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

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

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

LIBNAME
Information Maps engine libref for the information map

MEMNAME
SAS name for the information map

MAPNAME
Information map name

PATH
Location of the information map within the metadata server

DESCRIPTION
Description of the information map

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

```sql
libname mymapps infomaps mappath="/Users/myUserID/My Folder*";

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

Output A4.1  Output from DICTIONARY.INFOMAPS Table Query

<table>
<thead>
<tr>
<th>Information Map Name</th>
<th>Information Map Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee Statistics Sample</td>
<td>/Users/myUserID/My Folder</td>
</tr>
</tbody>
</table>

DICTIONARY.DATAITEMS Table

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

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

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

DATAITEMNAME
   Data item name

ID
   Data item 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:

```sas
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 A4.2  Output from DICTIONARY.DATAITEMS Table Query**

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Data Item Name</th>
<th>Data Item ID</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee Statistics Sample</td>
<td>Name</td>
<td>Name</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Identification</td>
<td>Identification</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Number</td>
<td>Number</td>
<td></td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Job Code</td>
<td>Jobcode</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Department</td>
<td>Deptcode</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Location</td>
<td>Location</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Average Salary</td>
<td>Salary_2</td>
<td>MEASURE</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Minimum Salary</td>
<td>Salary2</td>
<td>MEASURE</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Maximum Salary</td>
<td>Salary3</td>
<td>MEASURE</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Sum of Salaries</td>
<td>Salary4</td>
<td>MEASURE</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Hire Date</td>
<td>Hire Date</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Number of Years</td>
<td>Number of Years</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>Employee Statistics Sample</td>
<td>Employed</td>
<td>Employed</td>
<td></td>
</tr>
</tbody>
</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:

```sql
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 A4.3  Output from DICTIONARY.FILTERS Table Query

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Filter Name</th>
<th>Filter ID</th>
<th>Filter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee Statistics Sample</td>
<td>Cary Employees</td>
<td>Cary Employees</td>
<td>Employees who work in Cary, North Carolina. Employee filters based on selected departments.</td>
</tr>
<tr>
<td>Statistics Sample</td>
<td>Which department?</td>
<td>Which department?</td>
<td></td>
</tr>
</tbody>
</table>
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= System Option

Traces diagnostic messages issued by the Information Maps engine.

Valid in: configuration file, SAS invocation, OPTIONS statement
Default: none
See: SASTRACELOC= on page 166 system option

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= System Option

Specifies the destination for diagnostic messages.
Valid in: configuration file, SAS invocation, OPTIONS statement
Default: stdout
See: SASTRACE= on page 165 system option

Syntax

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

NOSTSUFFIX System Option

Simplifies diagnostic messages by suppressing some nonessential output.

Syntax

NOSTSUFFIX

Example: Displaying Detailed and Summary Timing Information

Example 1: 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;