SAS/ETS® 14.3 User's Guide
The SASEFAME Interface Engine

This document is an individual chapter from *SAS/ETS® 14.3 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2017. *SAS/ETS® 14.3 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/ETS® 14.3 User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

September 2017

Chapter 48
The SASEFAME Interface Engine

Contents

Overview: SASEFAME Interface Engine

The SASEFAME interface engine provides a seamless interface between Fame and SAS data to enable SAS users to access and process time series, case series, and formulas that reside in a Fame database.

Fame is an integrated, front-to-back market data and historical database solution for storing and managing real-time and high-volume time series data that are used by leading institutions in the financial, energy, and public sectors, as well as by third-party content aggregators, software vendors, and individual investors. Fame provides real-time market data feeds and end-of-day data, a web-based desktop solution, application hosting, data delivery components, and tools for performing analytic modeling.

The SASEFAME engine uses the LIBNAME statement to enable you to specify the time series that you want to read from the Fame database and how you want to convert the selected time series to the same time scale. You can then use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set. You can perform more analysis (if desired) either in the same SAS session or in a later session.

The SASEFAME interface engine supports Windows and Linux Opteron hosts that use Fame 11.5. Although SASEFAME is no longer available on the AIX and Solaris hosts, you can still get remote access to Fame data on those hosts by using SASEFAME from a Windows or Linux Opteron host to connect to the MCADBS or master server on the AIX and Solaris hosts. For more information about MarketMap (formerly Fame) servers, see *Guide to MarketMap Database Servers*, formerly known as *Guide to Fame Database Servers*.

Getting Started: SASEFAME Interface Engine

Structure of a SAS Data Set That Contains Time Series Data

The SAS System represents time series data in a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASEFAME engine provides a time ID variable named DATE. The DATE variable can be represented by any of the time intervals shown in the section "Mapping Fame Frequencies to SAS Time Intervals" on page 3476.

Reading and Converting Fame Database Time Series

The SASEFAME engine supports reading and converting time series that reside in Fame databases. The SASEFAME engine uses Fame's Work database to temporarily store the converted time series. All series that are specified by the Fame wildcard are written to the Fame Work database. For conversion of very large databases, you might want to define the FAME_TEMP environment variable to point to a location where there is ample space for the Fame Work database.

The SASEFAME engine provides seamless access to Fame databases via Fame's C host language interface (CHLI). Fame expressions that contain formulas and Fame functions can be input to the engine via the INSET= option.

The SASEFAME engine finishes the CHLI whenever a fatal error occurs. To restart the engine after a fatal error, terminate the current SAS session and open a new SAS session.

Using the SAS DATA Step

You can store the converted series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. After your data are stored in a SAS data set, you can use this data set as you would any other SAS data set.

Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure and report information about the contents of your data set by using the CONTENTS procedure, as in Example 48.1. You can create a view of the Fame database by using the SQL procedure's USING clause to reference the SASEFAME engine in your libref. See Example 48.5.

Using the SAS Windowing Environment

You can see the available data sets in the SAS LIBNAME window of the SAS windowing environment. To do so, select the SASEFAME engine libref in the LIBNAME window that you have previously defined in your LIBNAME statement. You can view your SAS output observations by double-clicking the desired output data set libref in the LIBNAME window of the SAS windowing environment. Type **Viewtable** on the SAS command line to view any of your SASEFAME engine tables, views, or librefs both for input and output data sets. Before you use the **Viewtable** command, it is recommended that you store your output data sets in a physical folder or library that is separate from the folder or library used for your input databases. (The default location for output data sets is the SAS Work library.)

Remote Fame Data Access

The remote access feature of the SASEFAME engine uses the MarketMap (Fame) CHLI to communicate with your remote server (master or MCADBS). It is available to licensed MarketMap customers who have the CHLI on both their remote and client machines.

For an example that uses the master server, see Example 48.7, where you simply provide the frdb_m port number and node name of your Fame master server in your SASEFAME engine libref. For more information, see the section "Start the Master Server" in *Guide to MarketMap Database Servers*.

For an example that uses the MCADBS remote server, see Example 48.18, where you specify an explicit connection with the CONNECT=YES option, and you specify the service, host, and name of the connection by using the TO_SERVICE= option, ON_HOST= option, and AS_NAME= options, respectively. In addition, you specify the USER= and PASS= options for the connection. For more information, see the section "Start the MCADBS Server" in *Guide to MarketMap Database Servers*.

Creating Views of Time Series by Using SASEFAME LIBNAME Options

You can perform selection based on names of your time series simply by using Fame wildcard specifications in the SASEFAME engine WILDCARD= option.

You can limit the time span of time series data by specifying a beginning and ending date range in the SASEFAME engine RANGE= option.

It is also easy to use the SAS input data set INSET= option to create a specific view of your Fame data. You can create multiple views by using multiple LIBNAME statements with customized options that are tailored to the unique views that you want to create.

You can list the INSET variables that you want to keep in your SAS data set by using the KEEP= clause. When you use INSET variables in conjunction with the input data set that you specify in the INSET= option, the SASEFAME engine can show any or all of your expression groups in the same view or in multiple views. The INSET= option defines the valid set of expression groups that you can reference in the KEEP= clause, as shown in Example 48.16.

The INSET variables define the BY variables that enable you to view cross sections (slices) of your data. When you use INSET variables in conjunction with the WHERE clause and the CROSSLIST= option, the SASEFAME engine can show any or all of your BY groups in the same view or in multiple views. When you use the INSET= option along with a WHERE clause that specifies the BY variables that you want to use in your view, you must also use the CROSSLIST= option, as shown in Example 48.10. You can use the CROSSLIST= option without using the INSET= option, as shown in Example 48.8 and Example 48.9.

Syntax: SASEFAME Interface Engine

The SASEFAME interface engine uses standard engine syntax. Table 48.1 summarizes the options used by the SASEFAME engine.

Table 48.1 Summary of LIBNAME *libref* SASEFAME Statement Options

Option	Description
AS_DB=	Specifies the channel name to use for a local database; used when Fame expressions or formulas need to resolve in a Fame child process.
AS_NAME=	Specifies the name to use for an explicit connection for remote access; used with the CONNECT=YES option.
CONNECT=	Specifies whether or not you want to use an explicit named connection for remote access, which you name in the AS_NAME= option
CONVERT=	Specifies the Fame frequency and the Fame technique
CROSSLIST=	Specifies a Fame crosslist *fame_namelist* to perform selection based on the crossproduct of two Fame namelists
DBVERSION=	Echoes the present version number of the Fame Work database in the SAS log
DEBUG=	Specifies whether or not you need diagnostic message logging in the SAS log window
FAMEOUT=	Specifies the Fame data object class/type that you want output to the SAS data set
INSET=	Uses a SAS data set named setname and KEEP= *fame_expression_group* as selection input variables or WHERE= *fame_bygroup* as selection input for BY variables
ON_HOST=	Specifies the remote Fame MCADBS server node name to connect to; used with the CONNECT=YES option.
PASS=	Specifies the password for the connection, which should match the password that you use as the **adduser** parameter in your **Fame FRDB facsq** command; used with the USER= option (for remote access).
RANGE=	Specifies the range of data to keep in `'ddmmmyyy'` – `'ddmmmyyyy'` format
TO_SERVICE=	Specifies the port number that you started the MCADBS service on, which is the same port that you specified in the **-p** argument in the **mcadbs.exe** command on your MCADBS server; used with the CONNECT=YES option (for remote access).
TUNEFAME=	Tunes the Fame database engine's use of memory to reduce I/O in favor of a bigger virtual memory for caching database objects
TUNECHLI=	Tunes the CHLI database engine's use of memory to reduce I/O in favor of a bigger virtual memory for caching database objects
USER=	Specifies the user name for the connection, which should match the user name you use as the **adduser** parameter in your **Fame FRDB facsq** command; used with the PASS= option (for remote access).
WILDCARD=	Specifies a Fame wildcard to match data object series names within the Fame database

LIBNAME *libref* SASEFAME Statement

LIBNAME *libref* **SASEFAME** '*physical name*' *options* **;**

Because '*physical name*' specifies the location of the folder where your Fame database resides, it should end in a backslash if you are in a Windows environment or a forward slash if you are in a UNIX environment.

If you are accessing a remote Fame database using an implicit connection in the Fame CHLI, you can use the following syntax for '*physical name*':

'`#port_number @hostname physical_path_name`'

You can specify the following *options*.

AS_DB=*fame_db_name*

OPEN_AS= *fame_db_name*

specifies the Fame database ID to use in the **Fame OPEN** command, which is often the same as the name of the database (without the *.db* extension). In Fame, you can retrieve a list of open database ID names by using the Fame command **TYPE @OPEN.DB**. Use this option when you get this error:

```
ERROR: From cfmfame: Error from a FAME-like server, error from
       cfmferr is: \Variable{XXXX} not found
```

For a more complete discussion of opening a local Fame database, see the section "Opening a Local Fame Database" on page 3472.

AS_NAME="*fame_channel_name*"

NAME= "*fame_channel_name*"

specifies the Fame channel name to use in the **Fame CONNECT** command for remote database access. In Fame, you can retrieve a list of open channel names by using the Fame command **TYPE @OPEN.CONNECTIONS**. For a more complete discussion of opening a remote Fame database on an MCADBS server, see Example 48.18.

CONNECT=YES | IMPLICIT | NO

CONNECTION=YES | IMPLICIT | NO

specifies whether or not the connection is an explicit connection.

YES	specifies that the connection is explicit.
IMPLICIT	specifies that the connection is implicit.
NO	specifies that the connection is implicit.

An explicit connection also requires the TO_SERVICE= , ON_HOST=, AS_NAME=, USER=, and PASS= options. When an implicit connection is specified, these additional options are not used; instead the details of the connection are given inside the physical pathname in the SASEFAME LIBNAME statement with the special syntax described in the section "LIBNAME *libref* SASEFAME Statement" on page 3468. For more information about implicit Fame connections, see the section "Opening Databases on Implicit Connections" in *MarketMap Fame 11.5 Online Help* at the following URL:

```
https://fame.sungard.com/support_secure/fame/online_help_115/commands_and_options/
          opening_databases_implicit_connections.htm
```

For more information about explicit Fame connections, see the section "Opening Databases on Explicit Connections" in the *MarketMap Fame 11.5 Online Help* at the following URL:

```
https://fame.sungard.com/support_secure/fame/online_help_115/commands_and_options/
          opening_databases_explicit_connections.htm
```

CONVERT=(FREQ=*fame_frequency* **TECH=***fame_technique***)**

CONV=(FREQ=*fame_frequency* **TECH=***fame_technique***)**

specifies the Fame frequency and the Fame technique, just as you would in the Fame CONVERT function. There are four possible values for *fame_technique*: *Constant* (default), *Cubic*, *Discrete*, and *Linear*. Table 48.2 shows the Fame frequencies that are supported by the SASEFAME engine.

For a more complete discussion of Fame frequencies and SAS time intervals, see the section "Mapping Fame Frequencies to SAS Time Intervals" on page 3476. For all possible *fame_frequency* values, see the section "Understanding Frequencies" in the *User's Guide to Fame*. For example:

```
LIBNAME libref sasefame 'physical-name'
        CONVERT=(TECH=CONSTANT FREQ=TWICEMONTHLY);
```

CROSSLIST=(< *fame_namelist1***, >** *fame_namelist2* **)**

CROSS=(< *fame_namelist1***, >** *fame_namelist2* **)**

performs a crossproduct of the members of the first namelist with the members of the second namelist, using a glue symbol "." to join the two. If one of the time series that are listed in *fame_namelist2* does not exist, the SASEFAME engine stops processing the remainder of the namelist. For more information, see the section "Performing the Crosslist Selection Function" on page 3479.

DBVERSION=ON | OFF

specifies whether or not to display the version number of the Fame Work database. DBVERSION=ON specifies that the SAS log show the version number (3 or 4) of the Fame Work database. By default, DBVERSION=OFF.

DEBUG= ON | OFF (default)

specifies that additional diagnostic information in the SAS log be reported. When you specify DEBUG=ON, the information about Fame commands that are outlined in the SAS log by debug tracing can be valuable for diagnosing and identifying the issues that cause errors when you are using the SASEFAME engine. By default, DEBUG=OFF. See Example 48.18 for a detailed SAS log that is created when you specify DEBUG=ON.

FAMEOUT=*fame_data_object_class_type*

specifies the class and type of the Fame data series objects to include in your SAS output data set. The possible values for *fame_data_object_class_type* are FORMULA, TIME, BOOLEAN, CASE, DATE, and STRING. Case series can be numeric, boolean, string, and date, or they can be generated using formulas that resolve to series. The SASEFAME engine resolves all formulas that belong to the type of series data object that you specify in the FAMEOUT= option. If the FAMEOUT= option is not specified, numeric time series are output to the SAS data set. FAMEOUT=CASE defaults to case series of numeric type. If you want another type of case series in your output, then you must specify it. Scalar data objects are not supported.

INSET=(*setname* **KEEP=***fame_expression_group* **)**

INSET=(*setname* **KEEPLIST=***fame_expression_group* **)**

specifies the name of a SAS data set (setname) and selects series that are generated by the expressions defined in *fame_expression_group*. You can define *fame_expression_group* by using Fame functions and Fame expressions. It is important to specify the length of the longest expression, or expressions might be truncated because the default length is the first defined variable in the DATA step. The INSET (input data set) must output each expression statement as a character string ending with a semicolon, enclosed in single quotation marks, and followed by another semicolon and an output statement. For more about using the INSET= option to define a group of selected series that are generated by Fame expressions, see the section "Performing the Keeplist Expression Function" on page 3477.

INSET=(*setname* **WHERE=***fame_bygroup* **)**

specifies a SAS data set (setname) as input for a BY group such as a ticker, and uses the *fame_bygroup* to select time series that are named using the following convention. Selected variable names are glued together by the BY-group name (such as a ticker symbol) concatenated with the glue character (such as DOT) to the series name that is specified in the CROSSLIST= option or in the *fame_bygroup*.

For more information, see the section "Performing the Crosslist Selection Function" on page 3479.

ON_HOST="*fame_hostname***"**

HOST= "*fame_hostname***"**

specifies the Fame host name to use in the **Fame CONNECT** command, which is the name of the host or node that is running as the MCADBS server. You can see the host name when you use the **MCADBS** command with the **show** option. For a more complete discussion of using the **MCADBS** command with the **show** option, see the section "Using the MCADBS Show Function" on page 3474.

PASS="*fame_password***"**

PASSWORD= "*fame_password***"**

specifies the Fame password to use in the **Fame CONNECT** command, which is the password for the user name designated in the **adduser** function in the **facsq access control** command. For a more complete discussion about managing and monitoring your Fame server processes, see the section "Managing Fame Server Processes for Remote Access" on page 3473.

RANGE='*fame_begdt***'d-'***fame_enddt***'d**

DATERANGE='*fame_begdt***'d-'***fame_enddt***'d**

DATE='*fame_begdt***'d-'***fame_enddt***'d**

DATECASE='*fame_begdt***'d-'***fame_enddt***'d**

limits the time range of data that are read from your Fame database. The string *fame_begdt* is the beginning date in `'ddmmmyyyy'` format, and the string *fame_enddt* is the ending date of the range in `'ddmmmyyyy'` format; both strings must be enclosed in single quotation marks and followed by the letter `'d'`.

For example, to read a series with a date range that spans the first quarter of 1999, you could use the following statement:

```
LIBNAME test sasefame 'physical name of test database'
        RANGE='01jan1999'd - '31mar1999'd;
```

TO_SERVICE="*fame_service_portnumber*"

SERVICE= "*fame_service_portnumber*"

specifies the Fame service port number to use in the **Fame CONNECT** command, which is the same port number that you use in your **MCADBS** command for the name port (**-n** option). For a more complete discussion about managing and monitoring your Fame server processes, see the section "Managing Fame Server Processes for Remote Access" on page 3473.

TUNEFAME=NODES *fameengine_size_virtual_memory_MB*

specifies the number of megabytes to use for the cache size for the Fame API (CHLI). The *fameengine_size_virtual_memory_MB* can range from a minimum of 0.1 MB (100 KB) to a maximum of 17,592,186,000,000 MB. For more information, see Example 48.17.

TUNECHLI=NODES *famechliengine_size_virtual_memory_MB*

specifies the number of megabytes to use for the cache size for the Fame API (CHLI). The *famechliengine_size_virtual_memory_MB* can range from a minimum of 0.1 MB (100 KB) to a maximum of 17,592,186,000,000 MB. For more information, see Example 48.17.

USER="*fame_username*"

USERNAME= "*fame_username*"

specifies the Fame user name to use in the **Fame CONNECT** command, which corresponds to the password and user name designated in the **adduser** function in the **facsq access control** command. For a more complete discussion about managing and monitoring your Fame server processes, see the section "Managing Fame Server Processes for Remote Access" on page 3473.

WILDCARD="*fame_wildcard***"**

WILD="*fame_wildcard***"**

limits the time series read from the Fame database. By default, the SASEFAME engine reads all time series in the Fame database that you name in your SASEFAME libref. The *fame_wildcard* is a quoted string that contains the Fame wildcard you want to use. The wildcard is used for matching against the data object names of series that you want to select from the Fame database that resides in the library you are assigning.

For more information about using wildcards, see the section "Specifying Wildcards" in the *User's Guide to Fame*.

For example, to read all time series in the TEST library that is being accessed by the SASEFAME engine, you would specify the following statement:

```
LIBNAME test sasefame 'physical name of test database'
        WILDCARD="?";
```

To read series that have names such as A_DATA, B_DATA, and C_DATA, you could specify the following statement:

```
LIBNAME test sasefame 'physical name of test database'
        WILDCARD="^_DATA";
```

When you use the WILDCARD= option, you limit the number of series that are read and converted to the desired frequency. This option can help you save resources when processing large databases or when processing a large number of observations, such as daily or hourly frequencies. Because the SASEFAME engine uses the Fame Work database to store the converted time series, using wildcards is recommended to prevent your workspace from getting too large. When the FAMEOUT= option is also specified, the wildcard is applied to the type of data object series that you specify in the FAMEOUT= option.

Details: SASEFAME Interface Engine

Opening a Local Fame Database

Fame databases often contain expressions and formulas that resolve to a series. In order for Fame to resolve the expressions and formulas a channel is opened to the local database to a Fame-like server that is invoked by the SASEFAME interface engine so that the selected series are complete.

For example, the following SAS code generates the SAS log after it, which shows the **OPEN** command that is used to open the local training database on the Fame channel named TR, enabling the Fame **Crosslist** to resolve all the time series values for all the tickers included in the inset's BY group (TICK) :

```
libname lib5 sasefame "\\tappan\crsp1\fame10"
   as_db="TR"
   debug=ON
   convert=(frequency=business technique=constant)
   inset=( inseta where=tick )
   crosslist=
     ({adjust, close, high, low, open, volume, uclose, uhigh, ulow, uopen, uvolume})

data trout;
   set lib5.training;
run;
```

Here is an excerpt of the information shown in the SAS log (on Windows), which is created by using the DEBUG=ON option:

```
NOTE: The SASEFAME engine is using Version 11.43000 of the HLI.

len4=2
SIMPLE FAMECMD for local open is: \\tappan\crsp1\fame10/training
len4= 2
FAME COMMAND line 1004 is:
OPEN <ACCESS READ> """\\tappan\crsp1\fame10/training""" AS TR
```

It is important to note that the SAS **SET** command for local access uses the database name, training (without the *.db* extension), in the DATA step. This is in contrast to the SET statement for remote MCADBS server

access, which uses the channel name in the SAS SET statement, as shown in Example 48.18. For more information about opening and closing local Fame databases, see the section "Opening and Closing Local Databases" in *Online Help for MarketMap Analytic Studio* at the following URL:

```
https://fame.sungard.com/support_secure/fame/online_help/commands_and_options/
          opening_local_databases.htm
```

Managing Fame Server Processes for Remote Access

Whether you use a master server or an MCADBS server, the appropriate configuration file is necessary. For the master server, on UNIX, your configuration file might look like this:

```
cat master1.config
security access all
dbback $FAME/frdb/dbback
```

Your **master** command could look like this:

```
$FAME/frdb/master -p \#5555 -s master1.config > master1.log &
```

For more information about the **master** server command, visit the following URL:

```
https://fame.sungard.com/support_secure/fame/online_help_115/
   servers/master_server_command.htm
```

To manage your MCADBS Fame server processes, you can start the FACS daemon on your Fame server. On Windows, enter the **facsd** command in the command window (if that is your Fame server):

```
%FAME%\frdb\64\facsd -d U:\fame940\doc\ -p 2990 -o U:\fame940\test\facs
    -s U:\fame940\doc\facsd.config
```

After you start the FACS daemon this way, you can use it for user authentication, access control, and accounting and logging facilities of Fame access control and accounting. To set up authentication, you can use the **facsq** command as follows:

```
%FAME%\frdb\64\facsq -p 2990 adduser <fame_username> <fame_password>
```

The user name and password in the **adduser** function are the same as those that are specified in the SASEFAME LIBNAME statement's USER= and PASS= options.

For a more complete discussion of the FACS daemon and configuration file, visit the following URLs:

```
https://fame.sungard.com/support_secure/fame/online_help_115/
    servers/facs_server_command.htm

https://fame.sungard.com/support_secure/fame/online_help_115/
    facs/facsd_access_control_command.htm

https://fame.sungard.com/support_secure/fame/online_help_115/
    facs/facsq_access_control_command.htm
```

Next you start your Fame server. The **MCADBS** server command, on Windows, looks like this:

```
C:\PROGRA~2\FAME\frdb\64\mcadbs.exe -n 2960 -p 2961 -s U:\fame940\doc\mcadbs.config
    -o U:\fame940\doc\mcadbs.log
```

For a more complete discussion, see the section "Start the MCADBS Server" in *Guide to MarketMap Database Servers*.

After starting the server, you can ask for information about the MACDBS server, as shown in the following section, "Using the MCADBS Show Function" on page 3474.

Using the MCADBS Show Function

When you have the MCADBS server running, you can get detailed information about the server by using the MCADBS **show** function as follows:

```
C:\Users\saskff>%FAME%\frdb\64\mcadbs -n 2960 show
```

This results in the following report:

```
MCADBS Release 11.4 64-bit  Copyright (C) 2014 by SunGard.  All rights reserved.


Operating System:          Windows 6.1 Service Pack 1
Hostname:                  d79286
Server pid:                7404
Listen Port:               2961
Name Port:                 2960
Client Limit:              25
Idle client timer:         3600
Inactive client timer:     600
Next expiration:           Fri Oct 17 12:11:39 2014
Request timeout:           none
Preserve search:           OFF
Configuration file:        U:\fame940\doc\mcadbs.config

Server Logging:
 Main log file:            U:\fame940\doc\mcadbs.log
 Logging levels:
  Default:                 detail
```

```
Security Rules:             Enforced by FACS
 Primary Daemon:            2990@localhost in use
  when unavailable:         Retry
 Secondary Daemon:          none configured
 Request Count:             1
 Last access Time:          Fri Oct 17 11:11:39 2014

Frdb Secure Settings:       Specified in configuration file
 Handshake:                 BEST
 Transport:                 NEVER

Procedure code files:
LOADED  FILE
YES     C:\PROGRA~2\FAME\sutil\adjdiv.pc
YES     C:\PROGRA~2\FAME\fdsutil\splfunc.pc

Databases:
Channel name  Status                Clients/Limit  File
TR            Open                        0/20     C:\PROGRA~2\FAME\util\trainin
g.db
SAMPLEV4      Open                        0/20     C:\PROGRA~2\FAME\util\samplev
4.db
DRIECON       Open                        0/20     C:\PROGRA~2\FAME\util\driecon
.db
OPT           Open                        0/0      C:\PROGRA~2\FAME\util\opt.db

No Active Clients
```

The host name, d79286, is listed in the first few lines of the report, after the operating system details.

SAS Output Data Set

You can use the SAS DATA step to write the selected time series from your Fame database to a SAS data set. This enables you to easily analyze the data by using the SAS System. You can specify the name of the output data set in the DATA statement. This causes the engine supervisor to create a SAS data set by using the specified name in either the SAS Work library or, if specified, the Sasuser library. For more information about naming your SAS data set, see the section “SAS Data Sets: Data Set Names” in *SAS Language Reference: Concepts*.

The contents of the SAS data set that contains time series include the date of each observation, the name of each series read from the Fame database as specified by the WILDCARD= option, and the label or Fame description of each series. Missing values are represented as ‘.’ in the SAS data set. You can see the available data sets in the SAS LIBNAME window of the SAS windowing environment by selecting the SASEFAME libref in the LIBNAME window that you have previously used in your LIBNAME statement. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can use PROC SQL and the SASEFAME engine to create a view of your SAS data set. You can view your SAS output observations by double-clicking the desired output data set libref in the LIBNAME window of the SAS windowing environment.

The DATE variable in the SAS data set contains the date of the observation. For Fame weekly intervals that end on a Friday, Fame reports the date on the Friday that ends the week, whereas the SAS System reports the date on the Saturday that begins the week.

A more detailed discussion of how to map Fame frequencies to SAS time intervals follows. For other types of data, such as Boolean case series, numeric case series, date case series, string case series, and extracting source for formulas, see Example 48.11, Example 48.12, Example 48.13, Example 48.14, and Example 48.15, respectively.

Mapping Fame Frequencies to SAS Time Intervals

Table 48.2 summarizes the mapping of Fame frequencies to SAS time intervals. Fame frequencies often have a sample unit in parentheses after the keyword frequency. This sample unit is an end-of-interval unit. SAS dates are represented by beginning-of-interval notation.

For more information about SAS time intervals, see Chapter 4, "Date Intervals, Formats, and Functions."

For more information about Fame frequencies, see the section "Understanding Frequencies" in the *User's Guide to Fame*.

Table 48.2 Mapping Fame Frequencies

Fame Frequency	SAS Time Interval
WEEKLY (SUNDAY)	WEEK.2
WEEKLY (MONDAY)	WEEK.3
WEEKLY (TUESDAY)	WEEK.4
WEEKLY (WEDNESDAY)	WEEK.5
WEEKLY (THURSDAY)	WEEK.6
WEEKLY (FRIDAY)	WEEK.7
WEEKLY (SATURDAY)	WEEK.1
BIWEEKLY (ASUNDAY)	WEEK2.2
BIWEEKLY (AMONDAY)	WEEK2.3
BIWEEKLY (ATUESDAY)	WEEK2.4
BIWEEKLY (AWEDNESDAY)	WEEK2.5
BIWEEKLY (ATHURSDAY)	WEEK2.6
BIWEEKLY (AFRIDAY)	WEEK2.7
BIWEEKLY (ASATURDAY)	WEEK2.1
BIWEEKLY (BSUNDAY)	WEEK2.9
BIWEEKLY (BMONDAY)	WEEK2.10
BIWEEKLY (BTUESDAY)	WEEK2.11
BIWEEKLY (BWEDNESDAY)	WEEK2.12
BIWEEKLY (BTHURSDAY)	WEEK2.13
BIWEEKLY (BFRIDAY)	WEEK2.14
BIWEEKLY (BSATURDAY)	WEEK2.8
BIMONTHLY (NOVEMBER)	MONTH2.2
BIMONTHLY	MONTH2.1

Table 48.2 *continued*

Fame Frequency	SAS Time Interval
QUARTERLY (OCTOBER)	QTR.2
QUARTERLY (NOVEMBER)	QTR.3
QUARTERLY	QTR.1
ANNUAL (JANUARY)	YEAR.2
ANNUAL (FEBRUARY)	YEAR.3
ANNUAL (MARCH)	YEAR.4
ANNUAL (APRIL)	YEAR.5
ANNUAL (MAY)	YEAR.6
ANNUAL (JUNE)	YEAR.7
ANNUAL (JULY)	YEAR.8
ANNUAL (AUGUST)	YEAR.9
ANNUAL (SEPTEMBER)	YEAR.10
ANNUAL (OCTOBER)	YEAR.11
ANNUAL (NOVEMBER)	YEAR.12
ANNUAL	YEAR.1
SEMIANNUAL (JULY)	SEMIYEAR.2
SEMIANNUAL (AUGUST)	SEMIYEAR.3
SEMIANNUAL (SEPTEMBER)	SEMIYEAR.4
SEMIANNUAL (OCTOBER)	SEMIYEAR.5
SEMIANNUAL (NOVEMBER)	SEMIYEAR.6
SEMIANNUAL	SEMIYEAR.1
YPP	Not supported
PPY	Not supported
SECONDLY	SECOND
MINUTELY	MINUTE
HOURLY	HOUR
DAILY	DAY
BUSINESS	WEEKDAY
TENDAY	TENDAY
TWICEMONTHLY	SEMIMONTH
MONTHLY	MONTH

Performing the Keeplist Expression Function

This section shows how to use the INSET= option to define a group of selected series that are generated by Fame expressions. It is important to use the LENGTH statement to avoid truncating the longest expression in

the group defined by the BY variable EXPRESS. **NOTE:** The EXPRESS variable is assigned the character string expression and is shown in Table 48.3. The following statements create an input data set, INSETA, and print it:

```
data inseta; /* Use this for training database */
  length express $52;
  express='{ibm.high,ibm.low,ibm.close};'; output;
  express='crosslist({gm,f,c},{volume});'; output;
  express='cvx.close;'; output;
  express='mave(ibm.close,30);'; output;
  express='cvx.close+ibm.close;'; output;
  express='ibm.close;'; output;
  express='close * shares/sum(close * shares);'; output;
  express='sum(pep.volume);'; output;
  express='mave(pep.close,20);'; output;
run;

proc print
   data=inseta;
run;
```

Next you can name the input data set that you want to use in the INSET= option, followed by the KEEP= variable that specifies the expression group you want to keep. Only series variables that are defined in the selected expression group are output to the output data set. You can define up to eight different expression groups in an INSET= option.

```
libname lib5 sasefame "C:\PROGRA~1\FAME10\util"
   wildcard="?"
   convert=(frequency=business technique=constant)
   range='23jul1997'd - '25jul1997'd
   inset=( inseta KEEP=express)
   ;

data trout;
   set lib5.trainten;
run;

title1 'TRAINING DB, Pricing Time Series for Expressions in INSET=';
title2 'OUT=TROUT from the PRINT Procedure';
proc print data=trout;
run;
```

Table 48.3 shows the eight expressions that are defined in INSETA.

Table 48.3 SAS Input Data Set, INSETA, Defined for Use in the INSET= Option

Observation	EXPRESS
1	cvx.close;
2	ibm.high,ibm.low,ibm.close;
3	mave(ibm.close,30);
4	crosslist(gm,f,c,volume);

Table 48.3 *continued*

Observation	EXPRESS
5	cvx.close+ibm.close;
6	ibm.close;
7	sum(pep.volume);
8	mave(pep.close,20);

Table 48.4 shows the output data set, TROUT. The output data set names each derived variable SASTEMPn by appending the number, n, to match the observation number of the input data set's expression for that variable. For example, SASTEMP1 names the series derived by 'cvx.close' in observation 1, and SASTEMP3 names the series derived by the expression 'mave(ibm.close,30);' in observation 3. Because SASTEMP2 is a simple name list of three series, the original series names are used.

Table 48.4 TRAINING DB, Pricing Timeseries for Expressions in INSETA for OUT=TROUT from the PRINT Procedure

DATE	C.VOLUME	VOLUME	GM.VOLUME	IBM.CLOSE	IBM.HIGH
23JUL1997	33791.88	45864.05	37392	52.5625	53.5000
24JUL1997	41828.85	29651.34	27771	53.9063	54.2188
25JUL1997	46979.83	36716.77	24969	53.5000	54.2188

IBM.LOW	SASTEMP1	SASTEMP3	SASTEMP5	SASTEMP6	SASTEMP8
51.5938	38.4063	.	90.9688	52.5625	.
52.2500	38.4375	.	92.3438	53.9063	.
52.8125	39.0000	.	92.5000	53.5000	.

Note that SASTEMP3 and SASTEMP8 have no observations in the date range July 23, 1997, to July 25, 1997, so the missing value symbol '.' appears for those observations.

Performing the Crosslist Selection Function

There are two methods of performing the crosslist selection function. The first method uses two Fame namelists, and the second method uses one namelist and one BY group specified in the WHERE= clause of the INSET= option.

For example, suppose that your Fame database has a string case series named TICKER, so that when the Fame NL function is used on TICKER, it returns the following namelist:

```
Ticker = {AOL, C, CVX, F, GM, HPQ, IBM, INDUA, INTC, SPX, SUNW, XOM}
```

Also suppose your time series are named in *fame_namelist2* as

```
{adjust, close, high, low, open, volume, uclose, uhigh, ulow, uopen, uvolume}
```

When you specify the following statements, the 132 variables shown in Table 48.5 are selected by the CROSSLIST= option:

```
LIBNAME test sasefame 'physical name of test database'
        RANGE='01jan1999'd - '31mar1999'd
        CROSSLIST=(nl(ticker),
                    {adjust, close, high, low, open, volume,
                     uclose, uhigh, ulow, uopen, uvolume})
        ;
```

Table 48.5 SAS Variables Selected by CROSSLIST= Option

AOL.ADJUST	C.ADJUST	CVX.ADJUST	F.ADJUST
AOL.CLOSE	C.CLOSE	CVX.CLOSE	F.CLOSE
AOL.HIGH	C.HIGH	CVX.HIGH	F.HIGH
AOL.LOW	C.LOW	CVX.LOW	F.LOW
AOL.OPEN	C.OPEN	CVX.OPEN	F.OPEN
AOL.UCLOSE	C.UCLOSE	CVX.UCLOSE	F.UCLOSE
AOL.UHIGH	C.UHIGH	CVX.UHIGH	F.UHIGH
AOL.ULOW	C.ULOW	CVX.ULOW	F.ULOW
AOL.UOPEN	C.UOPEN	CVX.UOPEN	F.UOPEN
AOL.UVOLUME	C.UVOLUME	CVX.UVOLUME	F.UVOLUME
AOL.VOLUME	C.VOLUME	CVX.VOLUME	F.VOLUME
GM.ADJUST	HPQ.ADJUST	IBM.ADJUST	INDUA.ADJUST
GM.CLOSE	HPQ.CLOSE	IBM.CLOSE	INDUA.CLOSE
GM.HIGH	HPQ.HIGH	IBM.HIGH	INDUA.HIGH
GM.LOW	HPQ.LOW	IBM.LOW	INDUA.LOW
GM.OPEN	HPQ.OPEN	IBM.OPEN	INDUA.OPEN
GM.UCLOSE	HPQ.UCLOSE	IBM.UCLOSE	INDUA.UCLOSE
GM.UHIGH	HPQ.UHIGH	IBM.UHIGH	INDUA.UHIGH
GM.ULOW	HPQ.ULOW	IBM.ULOW	INDUA.ULOW
GM.UOPEN	HPQ.UOPEN	IBM.UOPEN	INDUA.UOPEN
GM.UVOLUME	HPQ.UVOLUME	IBM.UVOLUME	INDUA.UVOLUME
GM.VOLUME	HPQ.VOLUME	IBM.VOLUME	INDUA.VOLUME
INTC.ADJUST	SPX.ADJUST	SUNW.ADJUST	XOM.ADJUST
INTC.CLOSE	SPX.CLOSE	SUNW.CLOSE	XOM.CLOSE
INTC.HIGH	SPX.HIGH	SUNW.HIGH	XOM.HIGH
INTC.LOW	SPX.LOW	SUNW.LOW	XOM.LOW
INTC.OPEN	SPX.OPEN	SUNW.OPEN	XOM.OPEN
INTC.UCLOSE	SPX.UCLOSE	SUNW.UCLOSE	XOM.UCLOSE
INTC.UHIGH	SPX.UHIGH	SUNW.UHIGH	XOM.UHIGH
INTC.ULOW	SPX.ULOW	SUNW.ULOW	XOM.ULOW
INTC.UOPEN	SPX.UOPEN	SUNW.UOPEN	XOM.UOPEN
INTC.UVOLUME	SPX.UVOLUME	SUNW.UVOLUME	XOM.UVOLUME
INTC.VOLUME	SPX.VOLUME	SUNW.VOLUME	XOM.VOLUME

Instead of using two namelists, you can use the WHERE= clause in an INSET= option to perform the crossproduct of the BY variables specified in your input data set via the WHERE= clause and the members named in your namelist. The following statements define a SAS input data set named INSETA to use as input for the CROSSLIST= option instead of using the Fame namelist:

```
DATA INSETA;
   LENGTH tick $5;
/* AOL, C, CVX, F, GM, HPQ, IBM, INDUA, INTC, SPX, SUNW, XOM */
   tick='AOL'; output;
   tick='C'; output;
   tick='CVX'; output;
   tick='F'; output;
   tick='GM'; output;
   tick='HPQ'; output;
   tick='IBM'; output;
   tick='INDUA'; output;
   tick='INTC'; output;
   tick='SPX'; output;
   tick='SUNW'; output;
   tick='XOM'; output;
RUN;

LIBNAME test sasefame 'physical name of test database'
        RANGE='01jan1999'd - '31mar1999'd
        INSET=(inseta, where=tick)
        CROSSLIST=(
              {adjust, close, high, low, open, volume,
               uclose, uhigh, ulow, uopen, uvolume})
        ;
```

Using a SAS INSET statement with a WHERE clause and using a Fame namelist in the CROSSLIST= statement are equivalent ways of performing the same selection function. In the preceding example, the Fame ticker namelist corresponds to the SAS input data set's BY variable named TICK.

Note that the *fame_bygroup* that you specify in the WHERE= clause must match the BY-variable name used in your input data set in order for the CROSSLIST= option to perform the desired selection. If one of the time series listed in *fame_namelist2* does not exist, the SASEFAME engine stops processing the remainder of the namelist. For complete results, make sure that your *fame_namelist2* is accurate and does not name unknown variables. The same holds true for *fame_namelist1* and the BY-variable values named in the input data set and used in the WHERE= clause.

Examples: SASEFAME Interface Engine

In this section, the examples were run on Windows, so the physical names used in the LIBNAME *libref* SASEFAME statement reflect the syntax necessary for that platform. In general, Windows environments use backslashes in their pathname, and the UNIX environments use forward slashes.

Example 48.1: Converting an Entire Fame Database

To enable conversion of all time series, no wildcard is specified, so the default "?" wildcard is used. Always consider both the number of time series and the number of observations generated by the conversion process. The converted series reside in the Fame Work database during the SAS DATA step. You can further limit your resulting SAS data set by using KEEP, DROP, or WHERE statements inside your DATA step.

The following statements convert a Fame database and print out its contents:

```
options pagesize=60 linesize=80 validvarname=any ;
%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir sasefame "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* Read in oecd1.db data from the Organization */
   /*  For Economic Cooperation and Development    */
   where date between '01jan88'd and '31dec93'd;
run;

proc print data=mydir.a;
run;
```

In the preceding example, the Fame database is called OECD1.DB, and it resides in the **famedir** directory. The DATA statement names the SAS output data set a that will reside in **mydir**. All time series in the Fame OECD1.DB database will be converted to an annual frequency and reside in the mydir.a SAS data set. Because the time series variable names contain the special glue symbol '.', the SAS option statement specifies VALIDVARNAME=ANY. For more information about this option, see *SAS System Options: Reference*. The Fame environment variable is the location of the Fame installation. In the Windows environment, the log would look like this:

```
1             options validvarname=any;

2             %let FAME=%sysget(FAME);
3             %put(&FAME);
(C:\PROGRA~1\FAME)
4             %let FAMETEMP=%sysget(FAME_TEMP);
5             %put(&FAMETEMP);
(\\ge\U11\saskff\fametemp\)
6
7             libname famedir sasefame "&FAME\util"
8                      convert=(freq=annual technique=constant);
NOTE: Libref FAMEDIR was successfully assigned as follows:
      Engine:        FAMECHLI
      Physical Name: C:\PROGRA~1\FAME\util
9
10            libname mydir '\\dntsrc\usrtmp\saskff';
NOTE: Libref MYDIR was successfully assigned as follows:
      Engine:        V9
      Physical Name: \\dntsrc\usrtmp\saskff
11
12            data mydir.a;  /* add data set to mydir */
13               set famedir.oecd1;
AUS.DIRDES -- SERIES (NUMERIC by ANNUAL)
AUS.DIRDES copied to work data base as AUS.DIRDES.
```

For more about the glue DOT character, see the section "Gluing Names Together" in the *User's Guide to Fame*. In the preceding log, the variable name AUS.DIRDES uses the glue DOT between AUS and DIRDES.

The PROC PRINT statement produces the results shown in Output 48.1.1, which displays all observations in the mydir.a SAS data set.

Output 48.1.1 Listing of OUT=MYDIR.A of the OECD1 Fame Data

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347
4	1991	.	.	.	.	.	.	1959.60	2488

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0
4	.	.	.	.	.	.	.	89908.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.
4	.	.	.	.	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	ITA.DIRDES	ITA.HERD	JPN.DIRDES	JPN.HERD	NLD.DIRDES
1	49.6000	37.0730	.	.	1861.50	2699927	9657.20	2014073	883
2	50.2000	39.0130	10.3000	786.762	1968.00	2923504	10405.90	2129372	945
3	51.7000	.	11.0000	902.498	2075.00	3183071	.	2296992	.
4	.	.	11.8000	990.865	2137.80	3374000	.	.	.

Obs	NLD.HERD	NOR.DIRDES	NOR.HERD	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES
1	2105	.	.	.	.	111.5	10158.20	.
2	2202	308.900	2771.40	78.7000	143.800	.	.	1076
3	.	.	.	.	.	.	.	.
4	.	352.000	3100.00	.	.	.	.	.

Obs	SWE.HERD	TUR.DIRDES	TUR.HERD	USA.DIRDES	USA.HERD	YUG.DIRDES	YUG.HERD
1	.	174.400	74474	20246.20	20246.20	233.000	29.81
2	11104	212.300	143951	22159.50	22159.50	205.100	375.22
3	.	.	.	23556.10	23556.10	.	2588.50
4	.	.	.	24953.80	24953.80	.	.

Example 48.2: Reading Time Series from the Fame Database

This example uses the Fame WILDCARD= option to limit the number of series converted. The following statements show how to read only series whose names begin with WSPCA:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname lib1 sasefame "%sysget(FAME_DATA)"
        wildcard="wspca?"
        convert=(technique=constant freq=twicemonthly );

libname lib2 "%sysget(FAME_TEMP)";

data lib2.twild(label='Annual Series from the FAMEECON.db');
   set lib1.subecon;
   where date between '01jan93'd and '31dec93'd;
   /* keep only  */
   keep date wspca;
run;

proc contents data=lib2.twild;
run;

proc print data=lib2.twild;
run;
```

Output 48.2.1 and Output 48.2.2 show the results of using WILDCARD="WSPCA?".

Output 48.2.1 Contents of OUT=LIB2.TWILD of the SUBECON Fame Data

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
1	DATE	Num	8	DATE9.	9.	Date of Observation
2	WSPCA	Num	8			STANDARD & POOR'S WEEKLY BOND YIELD: COMPOSITE, A

The WILDCARD="WSPCA?" option limits reading to only those series whose names begin with WSPCA. The KEEP statement further restricts the SAS data set to include only the series named WSPCA and the DATE variable. The time interval that is used for the conversion is TWICEMONTHLY.

Output 48.2.2 Listing of OUT=LIB2.TWILD of the SUBECON Fame Data

Obs	DATE	WSPCA
1	01JAN1993	8.59400
2	16JAN1993	8.50562
3	01FEB1993	8.47000
4	16FEB1993	8.31000
5	01MAR1993	8.27000
6	16MAR1993	8.29250
7	01APR1993	8.32400
8	16APR1993	8.56333
9	01MAY1993	8.37867
10	16MAY1993	8.26313
11	01JUN1993	8.21333
12	16JUN1993	8.14400
13	01JUL1993	8.09067
14	16JUL1993	8.09937
15	01AUG1993	7.98533
16	16AUG1993	7.91600

Example 48.3: Writing Time Series to the SAS Data Set

The following statements use the DROP statement to exclude certain time series from the SAS data set. (You can also use the KEEP statement to include certain series in the SAS data set.)

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir sasefame "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   drop 'ita.dirdes'n--'jpn.herd'n 'tur.dirdes'n--'usa.herd'n;
   where date between '01jan88'd and '31dec93'd;
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
title2 "Drop Using N-literals";

proc print data=mydir.a;
run;
```

Output 48.3.1 shows the results.

Output 48.3.1 Listing of OUT=MYDIR.A of the OECD1 Fame Data

OECD1: TECH=Constant, FREQ=Annual
Drop Using N-literals

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347
4	1991	.	.	.	.	.	.	1959.60	2488

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0
4	.	.	.	.	.	.	.	89908.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.
4	.	.	.	.	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	NLD.DIRDES	NLD.HERD	NOR.DIRDES	NOR.HERD
1	49.6000	37.0730	.	.	883	2105	.	.
2	50.2000	39.0130	10.3000	786.762	945	2202	308.900	2771.40
3	51.7000	.	11.0000	902.498	.	.	.	.
4	.	.	11.8000	990.865	.	.	352.000	3100.00

Obs	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES	SWE.HERD	YUG.DIRDES	YUG.HERD
1	.	.	111.5	10158.20	.	.	233.000	29.81
2	78.7000	143.800	.	.	1076	11104	205.100	375.22
3	.	.	.	.	.	.	.	2588.50
4	.	.	.	.	.	.	.	.

Note that the SAS option VALIDVARNAME=ANY was used at the beginning of this example because special characters are present in the time series names. SAS variables that contain certain special characters are called *n*-literals and are referenced in SAS code, as shown in this example.

You can rename your SAS variables by using the RENAME statement. The following statements show how to use *n*-literals when selecting variables that you want to keep and how to rename some of your kept variables:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir sasefame "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);
```

```
libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* keep and rename  */
   keep date 'ita.dirdes'n--'jpn.herd'n 'tur.dirdes'n--'usa.herd'n;
   rename 'ita.dirdes'n='italy.dirdes'n
          'jpn.dirdes'n='japan.dirdes'n
          'tur.dirdes'n='turkey.dirdes'n
          'usa.dirdes'n='united.states.of.america.dirdes'n ;
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
title2 "keep statement using n-literals";
title3 "rename statement using n-literals";

proc print data=mydir.a;
run;
```

Output 48.3.2 shows the results.

Output 48.3.2 Listing of OUT=MYDIR.A of the OECD1 Fame Data

OECD1: TECH=Constant, FREQ=Annual
keep statement using n-literals
rename statement using n-literals

Obs	DATE	italy.dirdes	ITA.HERD	japan.dirdes	JPN.HERD	turkey.dirdes	TUR.HERD
1	1985	1344.90	1751008	8065.70	1789780	144.800	22196
2	1986	1460.60	2004453	8290.10	1832575	136.400	26957
3	1987	1674.40	2362102	9120.80	1957921	121.900	32309
4	1988	1861.50	2699927	9657.20	2014073	174.400	74474
5	1989	1968.00	2923504	10405.90	2129372	212.300	143951
6	1990	2075.00	3183071	.	2296992	.	.
7	1991	2137.80	3374000	.	.	.	.

Obs	united.states.of.america.dirdes	USA.HERD
1	14786.00	14786.00
2	16566.90	16566.90
3	18326.10	18326.10
4	20246.20	20246.20
5	22159.50	22159.50
6	23556.10	23556.10
7	24953.80	24953.80

Example 48.4: Limiting the Time Range of Data

You can also limit the time range of the data in the SAS data set by using the RANGE= option in the LIBNAME statement or the WHERE statement in the DATA step to process the time ID variable DATE only when it falls in the range you are interested in.

All data for 1988, 1989, and 1990 are included in the SAS data set that is generated by using the RANGE='01JAN1988'D - '31DEC1990'D option or the WHERE DATE BETWEEN '01JAN88'D AND '31DEC90'D statement. The difference is that the RANGE= option uses less space in the Fame Work database. If you have a very large database and you want to use less space in your Fame Work database while you are processing the OECD1 database, you should use the RANGE= option as shown in the following statements:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir SASEFAME "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant)
        range='01jan1988'd - '31dec1990'd;

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* range on the libref restricts the dates *
    * read from famedir's oecd1 database      */
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
proc print data=mydir.a;
run;
```

Output 48.4.1 shows the results.

Output 48.4.1 OECD1 Fame Data Using the RANGE= Option

OECD1: TECH=Constant, FREQ=Annual

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	ITA.DIRDES	ITA.HERD	JPN.DIRDES	JPN.HERD	NLD.DIRDES
1	49.6000	37.0730	.	.	1861.5	2699927	9657.20	2014073	883
2	50.2000	39.0130	10.3000	786.762	1968.0	2923504	10405.90	2129372	945
3	51.7000	.	11.0000	902.498	2075.0	3183071	.	2296992	.

Obs	NLD.HERD	NOR.DIRDES	NOR.HERD	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES
1	2105	.	.	.	.	111.5	10158.20	.
2	2202	308.900	2771.40	78.7000	143.800	.	.	1076
3	.	.	.	.	.	.	.	.

Obs	SWE.HERD	TUR.DIRDES	TUR.HERD	USA.DIRDES	USA.HERD	YUG.DIRDES	YUG.HERD
1	.	174.400	74474	20246.20	20246.20	233.000	29.81
2	11104	212.300	143951	22159.50	22159.50	205.100	375.22
3	.	.	.	23556.10	23556.10	.	2588.50

The following statements show how you can use the WHERE statement in the DATA step to process the time ID variable DATE only when it falls in the range you are interested in:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir SASEFAME "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* where only  */
   where date between '01jan88'd and '31dec90'd;
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
proc print data=mydir.a;
run;
```

In Output 48.4.2, you can see that the result from the WHERE statement is the same as the result in Output 48.4.1 from using the RANGE= option.

Output 48.4.2 OECD1 Fame Data Using the WHERE Statement

OECD1: TECH=Constant, FREQ=Annual

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	ITA.DIRDES	ITA.HERD	JPN.DIRDES	JPN.HERD	NLD.DIRDES
1	49.6000	37.0730	.	.	1861.5	2699927	9657.20	2014073	883
2	50.2000	39.0130	10.3000	786.762	1968.0	2923504	10405.90	2129372	945
3	51.7000	.	11.0000	902.498	2075.0	3183071	.	2296992	.

Obs	NLD.HERD	NOR.DIRDES	NOR.HERD	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES
1	2105	.	.	.	.	111.5	10158.20	.
2	2202	308.900	2771.40	78.7000	143.800	.	.	1076
3	.	.	.	.	.	.	.	.

Obs	SWE.HERD	TUR.DIRDES	TUR.HERD	USA.DIRDES	USA.HERD	YUG.DIRDES	YUG.HERD
1	.	174.400	74474	20246.20	20246.20	233.000	29.81
2	11104	212.300	143951	22159.50	22159.50	205.100	375.22
3	.	.	.	23556.10	23556.10	.	2588.50

For more information about the KEEP, DROP, RENAME, and WHERE statements, see *SAS Language Reference: Concepts*.

Example 48.5: Creating a View Using the SQL Procedure and the SASEFAME Engine

The following statements create a view of OECD data by using the SQL procedure's FROM and USING clauses. For more information about SQL views, see the *SAS Visual Data Management and Utility Procedures Guide*.

```
title1 'famesql5: PROC SQL Dual Embedded Libraries w/ FAME option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

title2 'OECD1: Dual Embedded Library Allocations with FAME Option';
proc sql;
   create view fameview as
      select date, 'fin.herd'n
         from lib1.oecd1
      using libname lib1 sasefame "%sysget(FAME_DATA)"
               convert=(tech=constant freq=annual),
            libname temp "%sysget(FAME_TEMP)";
quit;

title2 'OECD1: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

Output 48.5.1 shows the results.

Output 48.5.1 Printout of the Fame View of OECD Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
OECD1: Print of View from Embedded Library with FAME Option

Obs	DATE	FIN.HERD
1	1985	1097.00
2	1986	1234.00
3	1987	1401.30
4	1988	1602.00
5	1989	1725.50
6	1990	1839.00
7	1991	.

The following statements create a view of the DRI Basic Economic data by using the SQL procedure's FROM and USING clauses:

```
title2 'SUBECON: Dual Embedded Library Allocations with FAME Option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

proc sql;
   create view fameview as
   select date, gaa
      from lib1.subecon
      using libname lib1 sasefame "%sysget(FAME_DATA)"
                         convert=(tech=constant freq=annual),
            libname temp "%sysget(FAME_TEMP)";
quit;

title2 'SUBECON: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

Output 48.5.2 shows the results.

Output 48.5.2 Printout of the Fame View of DRI Basic Economic Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
SUBECON: Print of View from Embedded Library with FAME Option

Obs	DATE	GAA
1	1946	.
2	1947	.
3	1948	23174
4	1949	19003
5	1950	24960
6	1951	21906
7	1952	20246
8	1953	20912
9	1954	21056
10	1955	27168
11	1956	27638
12	1957	26723
13	1958	22929
14	1959	29729
15	1960	28444
16	1961	28226
17	1962	32396
18	1963	34932
19	1964	40024
20	1965	47941
21	1966	51429
22	1967	49164
23	1968	51208
24	1969	49371
25	1970	44034
26	1971	52352
27	1972	62644
28	1973	81645
29	1974	91028
30	1975	89494
31	1976	109492
32	1977	130260
33	1978	154357
34	1979	173428
35	1980	156096
36	1981	147765
37	1982	113216
38	1983	133495
39	1984	146448
40	1985	128522
41	1986	111338
42	1987	160785
43	1988	210532
44	1989	201637
45	1990	218702
46	1991	210666

Output 48.5.2 *continued*

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
SUBECON: Print of View from Embedded Library with FAME Option

Obs	DATE	GAA
47	1992	.
48	1993	.

The following statements create a view of the DB77 database by using the SQL procedure's FROM and USING clauses:

```
title2 'DB77: Dual Embedded Library Allocations with FAME Option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

proc sql;
   create view fameview as
      select date, ann, 'qandom.x'n
      from lib1.db77
      using libname lib1 sasefame "%sysget(FAME_DATA)"
                    convert=(tech=constant freq=annual),
            libname temp "%sysget(FAME_TEMP)";
 quit;

title2 'DB77: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

Output 48.5.3 shows the results.

Output 48.5.3 Printout of the Fame View of DB77 Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
DB77: Print of View from Embedded Library with FAME Option

Obs	DATE	ANN	QANDOM.X
1	1959	.	0.56147
2	1960	.	0.51031
3	1961	.	.
4	1962	.	.
5	1963	.	.
6	1964	.	.
7	1965	.	.
8	1966	.	.
9	1967	.	.
10	1968	.	.
11	1969	.	.
12	1970	.	.
13	1971	.	.
14	1972	.	.
15	1973	.	.
16	1974	.	.
17	1975	.	.
18	1976	.	.
19	1977	.	.
20	1978	.	.
21	1979	.	.
22	1980	100	.
23	1981	101	.
24	1982	102	.
25	1983	103	.
26	1984	104	.
27	1985	105	.
28	1986	106	.
29	1987	107	.
30	1988	109	.
31	1989	111	.

The following statements create a view of the Data Resources Incorporated (DRI) Basic Economic data by using the SQL procedure's FROM and USING clauses:

```
title2 'DRIECON: Dual Embedded Library Allocations with FAME Option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

proc sql;
   create view fameview as
```

```
      select date, husts
      from lib1.driecon
      using libname lib1 sasefame "%sysget(FAME_DATA)"
                         convert=(tech=constant freq=annual)
                         range='01jan1980'd - '01jan2006'd ,
            libname temp "%sysget(FAME_TEMP)";
quit;

title2 'DRIECON: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

The SAS option VALIDVARNAME=ANY is used at the beginning of this example because special characters are present in the time series names. The output from this example shows how each Fame view is the output of the SASEFAME engine's processing. Different engine options could have been used in the USING LIBNAME clause if desired. Output 48.5.4 shows the results.

Output 48.5.4 Printout of the Fame View of DRI Basic Economic Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
DRIECON: Print of View from Embedded Library with FAME Option

Obs	DATE	HUSTS
1	1980	1292.2
2	1981	1084.2
3	1982	1062.2
4	1983	1703.0
5	1984	1749.5
6	1985	1741.8
7	1986	1805.4
8	1987	1620.5
9	1988	1488.1
10	1989	1376.1
11	1990	1192.7
12	1991	1013.9
13	1992	1199.7
14	1993	1287.6
15	1994	1457.0
16	1995	1354.1
17	1996	1476.8
18	1997	1474.0
19	1998	1616.9
20	1999	1666.5
21	2000	1568.7
22	2001	1602.7
23	2002	1704.9
24	2003	.

Example 48.6: Reading Other Fame Data Objects with the FAMEOUT= Option

This example shows how you can designate the data objects that are output to your SAS data set by using the FAMEOUT= option. In this example, the FAMEOUT=FORMULA option selects the formulas and their source definitions to be output. The RANGE= option is ignored because no time series are selected when FAMEOUT=FORMULA is specified.

```
options validvarname=any ls=90;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname lib6 sasefame "%sysget(FAME_DATA)"
   fameout=formula
   convert=(frequency=business technique=constant)
   range='02jan1995'd - '25jul1997'd
   wildcard="?YIELD?" ;

data crout;
   set lib6.training;
   keep 'S.GM.YIELD.A'n -- 'S.XON.YIELD.A'n ;
run;

title1 'Formulas from the TRAINING DB, FAMEOUT=FORMULA Option';
title2 'Using WILDCARD="?YIELD?"';
proc contents
   data=crout;
run;
```

Output 48.6.1shows the results.

Output 48.6.1 Contents of OUT=CROUT from the FAMEOUT=FORMULA Option of the Fame TRAINING Data

Formulas from the TRAINING DB, FAMEOUT=FORMULA Option Using WILDCARD="?YIELD?"

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	S.GM.YIELD.A	Char	82
2	S.GM__PP.YIELD.A	Char	82
3	S.HWP.YIELD.A	Char	82
4	S.IBM.YIELD.A	Char	82
5	S.INDUT.YIELD.A	Char	82
6	S.SPAL.YIELD.A	Char	82
7	S.SPALN.YIELD.A	Char	82
8	S.SUNW.YIELD.A	Char	82
9	S.XOM.YIELD.A	Char	82
10	S.XON.YIELD.A	Char	82

The FAMEOUT=FORMULA option restricts the SAS data set to include only formulas. The WILDCARD="?YIELD?" option further limits the selection of formulas to those whose names contain "YIELD".

```
options validvarname=any linesize=79;

title1 'Formulas from the TRAINING DB, FAMEOUT=FORMULA Option';
title2 'Using WILDCARD="?YIELD?"';
proc print
   data=crout noobs;
run;
```

Output 48.6.2 shows the results.

Output 48.6.2 Listing of OUT=CROUT from the FAMEOUT=FORMULA Option of the Fame TRAINING Data

Formulas from the TRAINING DB, FAMEOUT=FORMULA Option Using WILDCARD="?YIELD?"

S.GM.YIELD.A	S.GM__PP.YIELD.A
(%SPLC2TF(C37044210X01, IAD_DATE.H, IAD.H)/C37044210X01.CLOSE)*C37044210X01.ADJUST	(%SPLC2TF(C37044210X01, IAD_DATE.H, IAD.H)/C37044210X01.CLOSE)*C37044210X01.ADJUST

S.HWP.YIELD.A	S.IBM.YIELD.A
(%SPLC2TF(C42823610X01, IAD_DATE.H, IAD.H)/C42823610X01.CLOSE)*C42823610X01.ADJUST	(%SPLC2TF(C45920010X01, IAD_DATE.H, IAD.H)/C45920010X01.CLOSE)*C45920010X01.ADJUST

S.INDUT.YIELD.A	S.SPAL.YIELD.A
(%SPLC2TF(C00000110X00, IAD_DATE.H, IAD.H)/C00000110X00.CLOSE)*C00000110X00.ADJUST	(%SPLC2TF(C00000117X00, IAD_DATE.H, IAD.H)/C00000117X00.CLOSE)*C00000117X00.ADJUST

S.SPALN.YIELD.A	S.SUNW.YIELD.A
(%SPLC2TF(C00000117X00, IAD_DATE.H, IAD.H)/C00000117X00.CLOSE)*C00000117X00.ADJUST	(%SPLC2TF(C86681010X60, IAD_DATE.H, IAD.H)/C86681010X60.CLOSE)*C86681010X60.ADJUST

S.XOM.YIELD.A	S.XON.YIELD.A
(%SPLC2TF(C30231G10X01, IAD_DATE.H, IAD.H)/C30231G10X01.CLOSE)*C30231G10X01.ADJUST	(%SPLC2TF(C30231G10X01, IAD_DATE.H, IAD.H)/C30231G10X01.CLOSE)*C30231G10X01.ADJUST

Additional examples of the FAMEOUT= option are shown in Example 48.11, Example 48.12, Example 48.13, Example 48.14, and Example 48.15.

Example 48.7: Remote Fame Access by Using Fame CHLI

When you run Fame in a client/server environment and also have Fame CHLI capability to enable access to the server, you can access Fame remote data. Access the remote data by specifying the port number of the TCP/IP service that is defined for the frdb_m and the node name of the Fame master server in the physical path. In this example, the Fame server node name is STONES, and the port number is 5555, as was designated in the Fame master command. For more information about starting your Fame master server, see the section "Starting the Master Server" in *Guide to Fame Database Servers*.

```
options ls=78;
title1 "DRIECON Database, Using FAME with Remote Access via CHLI";
options validvarname=any;
libname test1 sasefame '#5555@stones $FAME/util';

data a;
   set test1.driecon;
   keep YP ZA ZB;
   where date between '01jan98'd and '31dec03'd;
run;

proc means data=a n;
run;
```

Output 48.7.1 shows the results.

Output 48.7.1 Summary Statistics for the Remote FAME Data

DRIECON Database, Using FAME with Remote Access via CHLI

The MEANS Procedure

Variable	Label	N
YP	PERSONAL INCOME	5
ZA	CORPORATE PROFITS AFTER TAX EXCLUDING IVA	4
ZB	CORPORATE PROFITS BEFORE TAX EXCLUDING IVA	4

Example 48.8: Selecting Time Series by Using the CROSSLIST= Option and KEEP Statement

This example shows how to use two Fame namelists to perform selection. Note that *fame_namelist1* could be easily generated using the Fame WILDLIST function. For more about the WILDLIST function, see the section "The WILDLIST Function" in the *Fame Command Reference, Volume 2, Functions*. In the following statements, four tickers are selected in *fame_namelist1*, but when you use the KEEP statement, the resulting data set contains only the desired IBM ticker:

```
options validvarname=any;

libname lib8 sasefame "%sysget(FAME_DATA)"
        convert=(frequency=business technique=constant)
        crosslist=(
           { IBM,SPALN,SUNW,XOM },
           { adjust, close, high, low, open, volume,
             uclose, uhigh, ulow,uopen,uvolume }
                 );

data trout;
   /* eleven companies, keep only the IBM ticker this time */
   set lib8.training;
   where date between '01mar02'd and '20mar02'd;
   keep IBM: ;
run;

title1 'TRAINING DB, Pricing Timeseries for IBM Ticker in CROSSLIST=';
proc contents
   data=trout;
run;

proc print
   data=trout;
run;
```

Output 48.8.1 and Output 48.8.2 show the results.

Output 48.8.1 Contents of the IBM Time Series in the Fame TRAINING Data

TRAINING DB, Pricing Timeseries for IBM Ticker in CROSSLIST=

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	IBM.ADJUST	Num	8
2	IBM.CLOSE	Num	8
3	IBM.HIGH	Num	8
4	IBM.LOW	Num	8
5	IBM.OPEN	Num	8
6	IBM.UCLOSE	Num	8
7	IBM.UHIGH	Num	8
8	IBM.ULOW	Num	8
9	IBM.UOPEN	Num	8
10	IBM.UVOLUME	Num	8
11	IBM.VOLUME	Num	8

Output 48.8.2 Listing of Ticker IBM Time Series in the Fame TRAINING Data

TRAINING DB, Pricing Timeseries for IBM Ticker in CROSSLIST=

Obs	IBM.ADJUST	IBM.CLOSE	IBM.HIGH	IBM.LOW	IBM.OPEN	IBM.UCLOSE	IBM.UHIGH
1	1	103.020	103.100	98.500	98.600	103.020	103.100
2	1	105.900	106.540	103.130	103.350	105.900	106.540
3	1	105.670	106.500	104.160	104.250	105.670	106.500
4	1	106.300	107.090	104.750	105.150	106.300	107.090
5	1	103.710	107.500	103.240	107.300	103.710	107.500
6	1	105.090	107.340	104.820	104.820	105.090	107.340
7	1	105.240	105.970	103.600	104.350	105.240	105.970
8	1	108.500	108.850	105.510	105.520	108.500	108.850
9	1	107.180	108.650	106.700	108.300	107.180	108.650
10	1	106.600	107.950	106.590	107.020	106.600	107.950
11	1	106.790	107.450	105.590	106.550	106.790	107.450
12	1	106.350	108.640	106.230	107.100	106.350	108.640
13	1	107.490	108.050	106.490	106.850	107.490	108.050
14	1	105.500	106.900	105.490	106.900	105.500	106.900

Obs	IBM.ULOW	IBM.UOPEN	IBM.UVOLUME	IBM.VOLUME
1	98.500	98.600	104890	104890
2	103.130	103.350	107650	107650
3	104.160	104.250	75617	75617
4	104.750	105.150	76874	76874
5	103.240	107.300	109720	109720
6	104.820	104.820	107260	107260
7	103.600	104.350	86391	86391
8	105.510	105.520	110640	110640
9	106.700	108.300	64086	64086
10	106.590	107.020	53335	53335
11	105.590	106.550	108640	108640
12	106.230	107.100	53048	53048
13	106.490	106.850	46148	46148
14	105.490	106.900	48367	48367

Example 48.9: Selecting Time Series by Using the CROSSLIST= Option and Fame Namelist

This example demonstrates selection by using the CROSSLIST= option. Only the ticker "IBM" is specified in the KEEP statement from the 11 companies in the Fame ticker namelist.

```
options validvarname=any;

libname lib9 sasefame "%sysget(FAME_DATA)"
        convert=(frequency=business technique=constant)
        range='07jul1997'd - '25jul1997'd
        crosslist=( nl(ticker),
                    { adjust, close, high, low, open, volume,
                      uclose, uhigh, ulow, uopen, uvolume }
                  );

data crout;
   /* eleven companies in the FAME ticker namelist */
   set lib9.training;
   keep IBM: ;
run;

title1 'TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST=';
title2 'Using TICKER Namelist';
proc print data=crout;
run;

proc contents data=crout;
run;
```

Output 48.9.1 and Output 48.9.2 show the results.

Output 48.9.1 Listing of OUT=CROUT Using CROSSLIST= Option in the Fame TRAINING Data

TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST= Using TICKER Namelist

Obs	IBM.ADJUST	IBM.CLOSE	IBM.HIGH	IBM.LOW	IBM.OPEN	IBM.UCLOSE	IBM.UHIGH
1	0.5	47.2500	47.7500	47.0000	47.5000	94.500	95.500
2	0.5	47.8750	47.8750	47.2500	47.2500	95.750	95.750
3	0.5	48.0938	48.3438	47.6563	48.0000	96.188	96.688
4	0.5	47.8750	48.0938	47.0313	47.3438	95.750	96.188
5	0.5	47.8750	48.6875	47.8125	47.9063	95.750	97.375
6	0.5	47.6250	48.2188	47.0000	47.8125	95.250	96.438
7	0.5	48.0000	48.1250	46.6875	47.4375	96.000	96.250
8	0.5	48.8125	49.0000	47.6875	47.8750	97.625	98.000
9	0.5	49.8125	50.8750	48.5625	48.9063	99.625	101.750
10	0.5	52.2500	52.6250	50.0000	50.0000	104.500	105.250
11	0.5	51.8750	53.1563	51.0938	52.6250	103.750	106.313
12	0.5	51.5000	51.7500	49.6875	50.0313	103.000	103.500
13	0.5	52.5625	53.5000	51.5938	52.1875	105.125	107.000
14	0.5	53.9063	54.2188	52.2500	52.8125	107.813	108.438
15	0.5	53.5000	54.2188	52.8125	53.9688	107.000	108.438

Obs	IBM.ULOW	IBM.UOPEN	IBM.UVOLUME	IBM.VOLUME
1	94.000	95.000	129012	64506
2	94.500	94.500	102796	51398
3	95.313	96.000	177276	88638
4	94.063	94.688	127900	63950
5	95.625	95.813	137724	68862
6	94.000	95.625	128976	64488
7	93.375	94.875	149612	74806
8	95.375	95.750	215440	107720
9	97.125	97.813	315504	157752
10	100.000	100.000	463480	231740
11	102.188	105.250	328184	164092
12	99.375	100.063	368276	184138
13	103.188	104.375	219880	109940
14	104.500	105.625	204088	102044
15	105.625	107.938	146600	73300

Output 48.9.2 Contents of OUT=CROUT Using CROSSLIST= Option in the Fame TRAINING Data

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	IBM.ADJUST	Num	8
2	IBM.CLOSE	Num	8
3	IBM.HIGH	Num	8
4	IBM.LOW	Num	8
5	IBM.OPEN	Num	8
6	IBM.UCLOSE	Num	8
7	IBM.UHIGH	Num	8
8	IBM.ULOW	Num	8
9	IBM.UOPEN	Num	8
10	IBM.UVOLUME	Num	8
11	IBM.VOLUME	Num	8

Example 48.10: Selecting Time Series by Using the CROSSLIST= Option and WHERE=TICK

Instead of having a Fame namelist with the ticker symbols for companies whose data you are interested in, you can designate an input SAS data set (INSETA) that specifies the tickers to select. Specify your selection by using the WHERE clause in the INSET= option as follows:

```
options validvarname=any;

data inseta;
   length tick $5;
   /* need $5 so SPALN is not truncated */

   tick='AOL';   output;
   tick='C';     output;
   tick='CPQ';   output;
   tick='CVX';   output;
   tick='F';     output;
   tick='GM';    output;
   tick='HWP';   output;
   tick='IBM';   output;
   tick='SPALN'; output;
   tick='SUNW';  output;
   tick='XOM';   output;
run;

libname lib10 sasefame "%sysget(FAME_DATA)"
        convert=(frequency=business technique=constant)
        range='07jul1997'd - '25jul1997'd
        inset=( inseta where=tick )
        crosslist=
            ( {adjust, close, high, low, open, volume,
               uclose, uhigh, ulow,uopen,uvolume} );
```

```
data trout;
   /* eleven companies with unique TICKs specified in INSETA */
   set lib10.training;
   keep IBM: ;
run;

title1 'TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST=';
title2 'Using INSET with WHERE=TICK';
proc print data=trout;
run;

proc contents data=trout;
run;
```

Output 48.10.1 and Output 48.10.2 show the results.

Output 48.10.1 Listing of OUT=TROUT Using CROSSLIST= and INSET= Options in the Fame TRAINING Data

**TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST=
Using INSET with WHERE=TICK**

Obs	IBM.ADJUST	IBM.CLOSE	IBM.HIGH	IBM.LOW	IBM.OPEN	IBM.UCLOSE	IBM.UHIGH
1	0.5	47.2500	47.7500	47.0000	47.5000	94.500	95.500
2	0.5	47.8750	47.8750	47.2500	47.2500	95.750	95.750
3	0.5	48.0938	48.3438	47.6563	48.0000	96.188	96.688
4	0.5	47.8750	48.0938	47.0313	47.3438	95.750	96.188
5	0.5	47.8750	48.6875	47.8125	47.9063	95.750	97.375
6	0.5	47.6250	48.2188	47.0000	47.8125	95.250	96.438
7	0.5	48.0000	48.1250	46.6875	47.4375	96.000	96.250
8	0.5	48.8125	49.0000	47.6875	47.8750	97.625	98.000
9	0.5	49.8125	50.8750	48.5625	48.9063	99.625	101.750
10	0.5	52.2500	52.6250	50.0000	50.0000	104.500	105.250
11	0.5	51.8750	53.1563	51.0938	52.6250	103.750	106.313
12	0.5	51.5000	51.7500	49.6875	50.0313	103.000	103.500
13	0.5	52.5625	53.5000	51.5938	52.1875	105.125	107.000
14	0.5	53.9063	54.2188	52.2500	52.8125	107.813	108.438
15	0.5	53.5000	54.2188	52.8125	53.9688	107.000	108.438

Obs	IBM.ULOW	IBM.UOPEN	IBM.UVOLUME	IBM.VOLUME
1	94.000	95.000	129012	64506
2	94.500	94.500	102796	51398
3	95.313	96.000	177276	88638
4	94.063	94.688	127900	63950
5	95.625	95.813	137724	68862
6	94.000	95.625	128976	64488
7	93.375	94.875	149612	74806
8	95.375	95.750	215440	107720
9	97.125	97.813	315504	157752
10	100.000	100.000	463480	231740
11	102.188	105.250	328184	164092
12	99.375	100.063	368276	184138
13	103.188	104.375	219880	109940
14	104.500	105.625	204088	102044
15	105.625	107.938	146600	73300

Output 48.10.2 Contents of OUT=TROUT Using CROSSLIST= and INSET= Options in the Fame TRAINING Data

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	IBM.ADJUST	Num	8
2	IBM.CLOSE	Num	8
3	IBM.HIGH	Num	8
4	IBM.LOW	Num	8
5	IBM.OPEN	Num	8
6	IBM.UCLOSE	Num	8
7	IBM.UHIGH	Num	8
8	IBM.ULOW	Num	8
9	IBM.UOPEN	Num	8
10	IBM.UVOLUME	Num	8
11	IBM.VOLUME	Num	8

Example 48.11: Selecting Boolean Case Series with the FAMEOUT= Option

This example shows how to extract all Boolean case series from the Fame ALLTYPES database. The following statements write all Boolean case series to the SAS data set BOOOUT:

```
title1 '***famallt: FAMEOUT Option, Different Type Values***';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname lib4 sasefame "%sysget(FAME_DATA)"
   fameout=boolcase wildcard="?" ;

data booout;
   set lib4.alltypes;
run;

title1 'ALLTYPES FAMEOUT=BOOLCASE for Boolean Case Series';
title2 'Using FAMEOUT=CASE BOOLEAN Option without Range';
proc contents
   data=booout;
run;

proc print
   data=booout;
run;
```

Output 48.11.1 and Output 48.11.2 show the results for the Boolean case.

Output 48.11.1 Contents of OUT=BOOOUT Using FAMEOUT=BOOLCASE for Boolean Case Series

ALLTYPES FAMEOUT=BOOLCASE for Boolean Case Series
Using FAMEOUT=CASE BOOLEAN Option without Range

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	BOO0	Num	8
2	BOO1	Num	8
3	BOO2	Num	8
4	BOOM	Num	8
5	BOO_RES	Num	8

Output 48.11.2 Listing of OUT=BOOOUT Using FAMEOUT=BOOLCASE for Boolean Case Series

ALLTYPES FAMEOUT=BOOLCASE for Boolean Case Series
Using FAMEOUT=CASE BOOLEAN Option without Range

Obs	BOO0	BOO1	BOO2	BOOM	BOO_RES
1	0	1	0	1	.
2	0	0	1	0	.
3	0	0	0	251	.
4	0	1	1	1	.
5	0	1	0	1	.
6	0	0	.	0	.
7	0	0	.	0	.
8	0	1	.	1	.
9	0	.	0	.	.
10	0	.	.	.	.
11	1	.	.	.	.
12	1	.	.	.	.
13	1	.	1	.	.
14	1	.	.	.	.
15	1	.	.	.	.
16	1	.	.	.	.
17	1	.	0	.	.
18	1	.	.	.	.
19	1	.	.	.	.
20	1	.	.	.	.

Example 48.12: Selecting Numeric Case Series with the FAMEOUT= Option

This example extracts numeric case series. In addition to the already existing numeric case series in the Fame database, you can also have formulas that expand to numeric case series. The SASEFAME engine resolves all formulas that belong to the class and type of series data object that you specify in the FAMEOUT= option. The following statements write all numeric case series to the SAS data set CSOUT:

```
libname lib5 sasefame "%sysget(FAME_DATA)"
   fameout=case wildcard="?" ;

data csout;
   set lib5.alltypes;
run;

title1 'Using FAMEOUT=CASE Option without Range';
title2 'ALLTYPES, FAMEOUT=CASE and Open Wildcard for Numeric Case Series';
proc contents
   data=csout;
run;

proc print
   data=csout;
run;
```

Output 48.12.1 and Output 48.12.2 show the results.

Output 48.12.1 Contents of OUT=CSOUT Using FAMEOUT=CASE and Open Wildcard for Numeric Case Series

Using FAMEOUT=CASE Option without Range
ALLTYPES, FAMEOUT=CASE and Open Wildcard for Numeric Case Series

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	FRM1	Num	8
2	NUM0	Num	8
3	NUM1	Num	8
4	NUM2	Num	8
5	NUMM	Num	8
6	NUM_RES	Num	8
7	PRC0	Num	8
8	PRC1	Num	8
9	PRC2	Num	8
10	PRCM	Num	8
11	PRC_RES	Num	8

Output 48.12.2 Listing of OUT=CSOUT Using FAMEOUT=CASE and Open Wildcard for Numeric Case Series

Using FAMEOUT=CASE Option without Range
ALLTYPES, FAMEOUT=CASE and Open Wildcard for Numeric Case Series

Obs	FRM1	NUM0	NUM1	NUM2	NUMM	NUM_RES	PRC0	PRC1	PRC2	PRCM	PRC_RES
1	0.00000	-9	0	1.33333	0	.	-18	0	1.33333	0	.
2	1.00000	-8	1	1.00000	1	.	-16	1	1.00000	1	.
3	0.66667	-7	2	0.66667	1.7014E38	.	-14	2	0.66667	1.7014E38	.
4	3.00000	-6	3	0.33333	3	.	-12	3	0.33333	3	.
5	4.00000	-5	4	0.00000	4	.	-10	4	0.00000	4	.
6	.	-4	5	.	5	.	-8	5	.	5	.
7	.	-3	6	.	6	.	-6	6	.	6	.
8	7.00000	-2	7	.	7	.	-4	7	.	7	.
9	.	-1	.	-1.33333	.	.	-2	.	-1.33333	.	.
10	.	0	.	.	.	.	0	.	.	.	.
11	.	1	.	.	.	.	2	.	.	.	.
12	.	2	.	.	.	.	4	.	.	.	.
13	.	3	.	-2.66667	.	.	6	.	-2.66667	.	.
14	.	4	.	.	.	.	8	.	.	.	.
15	.	5	.	.	.	.	10	.	.	.	.
16	.	6	.	.	.	.	12	.	.	.	.
17	.	7	.	-4.00000	.	.	14	.	-4.00000	.	.
18	.	8	.	.	.	.	16	.	.	.	.
19	.	9	.	.	.	.	18	.	.	.	.
20	.	10	.	.	.	.	20	.	.	.	.

Example 48.13: Selecting Date Case Series with the FAMEOUT= Option

This example shows how to extract date case series. In addition to the existing date case series in the Fame database, you can have formulas that resolve to date case series. The SASEFAME engine resolves all formulas that belong to the class and type of series data object that you specify in the FAMEOUT= option. The following statements write all date case series to the SAS data set CDOUT:

```
libname lib6 sasefame "%sysget(FAME_DATA)"
   fameout=datecase wildcard="?" ;

data cdout;
   set lib6.alltypes;
run;

title1 'Using FAMEOUT=DATECASE Option without Range';
title2 'ALLTYPES: FAMEOUT=DATECASE and Open Wildcard for Date Case Series';
proc contents
   data=cdout;
run;

proc print
   data=cdout;
run;
```

Output 48.13.1 and Output 48.13.2 show the results.

Output 48.13.1 Contents of OUT=CDOUT Using FAMEOUT=DATECASE

Using FAMEOUT=DATECASE Option without Range
ALLTYPES: FAMEOUT=DATECASE and Open Wildcard for Date Case Series

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat
1	DAT0	Num	8	YEAR4.	4.
2	DAT1	Num	8	YEAR4.	4.
3	DAT2	Num	8	YEAR4.	4.
4	DATM	Num	8	YEAR4.	4.
5	FRM2	Num	8	YEAR4.	4.

Output 48.13.2 Listing of OUT=CDOUT Using FAMEOUT=DATECASE

Using FAMEOUT=DATECASE Option without Range
ALLTYPES: FAMEOUT=DATECASE and Open Wildcard for Date Case Series

Obs	DAT0	DAT1	DAT2	DATM	FRM2
1	1991	1981	1987	1981	1987
2	1992	1982	1986	1982	1986
3	1993	1983	1985	1983	1985
4	1994	1984	1984	1984	1984
5	1995	1985	1983	1985	1983
6	1996	1986	.	1986	.
7	1997	1987	.	1987	.
8	1998	1988	.	1988	.
9	1999	.	1979	.	1979
10	2000	.	.	.	.
11	2001	.	.	.	.
12	2002	.	.	.	.
13	2003	.	1975	.	.
14	2004	.	.	.	.
15	2005	.	.	.	.
16	2006	.	.	.	.
17	2007	.	1971	.	.
18	2008	.	.	.	.
19	2009	.	.	.	.
20	2010	.	.	.	.

Example 48.14: Selecting String Case Series with the FAMEOUT= Option

This example shows how to extract string case series. In addition to the existing string case series in your Fame database, you can have formulas that resolve to string case series. The SASEFAME engine resolves all formulas that belong to the class and type of series data object that you specify in the FAMEOUT= option. The following statements write all string case series to the SAS data set CSTROUT:

```
libname lib7 sasefame "%sysget(FAME_DATA)"
   fameout=stringcase wildcard="?" ;

data cstrout;
   set lib7.alltypes;
run;

title1 'Using FAMEOUT=STRINGCASE Option without Range';
title2 'ALLTYPES, FAMEOUT=STRINGCASE and Open Wildcard for String Case Series';
proc contents
   data=cstrout;
run;

proc print
   data=cstrout;
run;
```

Output 48.14.1 and Output 48.14.2 show the results.

Output 48.14.1 Contents of OUT=CSTROUT Using FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

Using FAMEOUT=STRINGCASE Option without Range
ALLTYPES, FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	STR0	Char	16
2	STR1	Char	16
3	STR2	Char	16
4	STRM	Char	16

Output 48.14.2 Listing of OUT=CSTROUT Using FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

Using FAMEOUT=STRINGCASE Option without Range
ALLTYPES, FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

Obs	STR0	STR1	STR2	STRM
1	-9	0	1.333333	0
2	-8	1	1	1
3	-7	2	0.6666667	2
4	-6	3	0.3333333	3
5	-5	4	0	4
6	-4	5		5
7	-3	6		
8	-2	7		7
9	-1		-1.333333	
10	0			
11	1			
12	2			
13	3		-2.666667	
14	4			
15	5			
16	6			
17	7		-4	
18	8			
19	9			
20	10			

Example 48.15: Extracting Source for Formulas

This example shows how to extract the source for all the formulas in the Fame database by using the FAMEOUT=FORMULA and WILDCARD="?" options. The following statements show the source of all formulas written to the SAS data set CFOROUT. Another example of the FAMEOUT=FORMULA option is shown in Example 48.6.

```
libname lib8 sasefame "%sysget(FAME_DATA)"
   fameout=formula wildcard="?" ;

data cforout;
   set lib8.alltypes;
run;

title1 'Using FAMEOUT=FORMULA Option without Range';
proc contents
   data=cforout;
run;
```

Output 48.15.1 and Output 48.15.2 show the results.

Output 48.15.1 Contents of OUT=CFOROUT Using FAMEOUT=FORMULA and Open Wildcard

Using FAMEOUT=FORMULA Option without Range

The CONTENTS Procedure

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	S.DFRM	Char	27
2	S.FRM1	Char	27
3	S.FRM2	Char	27

```
title3 'ALLTYPES, FAMEOUT=FORMULA, and Open Wildcard for FORMULA Series';
proc print
   data=cforout noobs;
run;
```

Output 48.15.2 Listing of OUT=CFOROUT Using FAMEOUT=FORMULA and Open Wildcard

Using FAMEOUT=FORMULA Option without Range

ALLTYPES, FAMEOUT=FORMULA, and Open Wildcard for FORMULA Series

S.DFRM	S.FRM1	S.FRM2
IF DBOO THEN DPRC ELSE DNUM	IF BOO1 THEN NUM1 ELSE NUM2	IF BOO0 THEN DAT1 ELSE DAT2

If you want all series of every type, you can merge the resulting data sets. For more information about merging SAS data sets, see *SAS Language Reference: Concepts*.

Example 48.16: Reading Time Series by Defining Fame Expression Groups in the INSET= Option with the KEEP= Clause

To keep all the numeric time series that are listed in the expressions given in the input data set, INSETA, use the INSET=(*setname* KEEPLIST=*fame_expression_group*) and WILDCARD="?" options. The following statements show how to select time series that are specified in a KEEP expression group and are written to the SAS output data set:

```
data inseta; /* Use this for d8690 training database */
   length express $52;
   express='cvx.close;'; output;
   express='{ibm.high,ibm.low,ibm.close};'; output;
   express='mave(ibm.close,30);'; output;
   express='crosslist({gm,f,c},{volume});'; output;
   express='cvx.close+ibm.close;'; output;
   express='ibm.close;'; output;
   express='sum(pep.volume);'; output;
   express='mave(pep.close,20);'; output;
run;
```

```
title1 'TRAINING DB, Pricing Timeseries for Expressions in INSET=';
proc print
   data=inseta;
run;
```

Output 48.16.1 shows the expressions that are stored as observations in the input data set, INSETA.

Output 48.16.1 Listing of INSETA Defining Fame Expression Group

TRAINING DB, Pricing Timeseries for Expressions in INSET=

Obs	express
1	cvx.close;
2	{ibm.high,ibm.low,ibm.close};
3	mave(ibm.close,30);
4	crosslist({gm,f,c},{volume});
5	cvx.close+ibm.close;
6	ibm.close;
7	sum(pep.volume);
8	mave(pep.close,20);

The following statements show how to use the INSET= option to keep all time series that are represented in the input data set, INSETA, as the group variable EXPRESS:

```
libname libX sasefame "%sysget(FAME_DATA)"
   wildcard="?"
   convert=(frequency=business technique=constant)
   range='23jul1997'd - '25jul1997'd
   inset=( inseta KEEP=express)
   ;


data trout;
   set libX.trainten;
run;

title1 'TRAINING DB, Pricing Timeseries for Expressions in INSET=';
proc print data=trout;
run;

proc contents data=trout;
run;
```

Output 48.16.2 and Output 48.16.3 show the results.

Output 48.16.2 Listing of TROUT Using INSETA with KEEP=EXPRESS

TRAINING DB, Pricing Timeseries for Expressions in INSET=

Obs	DATE	C.VOLUME	VOLUME	GM.VOLUME	IBM.CLOSE	IBM.HIGH	IBM.LOW	SASTEMP1
1	23JUL1997	33791.88	45864.05	37392	52.5625	53.5000	51.5938	76.8125
2	24JUL1997	41828.85	29651.34	27771	53.9063	54.2188	52.2500	76.8750
3	25JUL1997	46979.83	36716.77	24969	53.5000	54.2188	52.8125	78.0000

Obs	SASTEMP3	SASTEMP5	SASTEMP6	SASTEMP8
1	47.0894	129.375	52.5625	37.6118
2	47.4289	130.781	53.9063	37.6250
3	47.7392	131.500	53.5000	37.6546

Output 48.16.3 Listing of Contents of TROUT

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
2	C.VOLUME	Num	8			
1	DATE	Num	8	DATE9.	9.	Date of Observation
4	GM.VOLUME	Num	8			
5	IBM.CLOSE	Num	8			
6	IBM.HIGH	Num	8			
7	IBM.LOW	Num	8			
8	SASTEMP1	Num	8			
9	SASTEMP3	Num	8			
10	SASTEMP5	Num	8			
11	SASTEMP6	Num	8			
12	SASTEMP8	Num	8			
3	VOLUME	Num	8			

Example 48.17: Optimizing Cache Sizes with the TUNEFAME= and TUNECHLI= Options

This example shows how to use the TUNEFAME= option, the TUNECHLI= option, and a RANGE= option to select pricing time series in the TRAINTEN database. The selected time series are written to the SAS output data set. The Fame database engine's virtual memory is given in megabytes (MB), so this example sets the cache size to 100 MB. The Fame CHLI engine's virtual memory is also given in megabytes (MB), so this example sets the CHLI cache size to 100 MB. These two settings correspond to the default settings. Both the Fame 4GL engine and the Fame CHLI engine can use a cache size that ranges from 0.1 MB to 17,592,186,000,000 MB.

```
libname lib5 sasefame "%sysget(FAME_DATA)"
  wildcard="?UHIGH"
  tunefame=nodes 100
  tunechli=nodes 100
  convert=(frequency=business technique=constant)
  range='23jul1997'd - '25jul1997'd
  ;

data trout(drop=C:);
   set lib5.trainten;
run;
title1 'TRAINTEN DB, Pricing Time Series, TUNEFAME=NODES and TUNECHLI=NODES Options';
proc print data=trout;
run;

proc contents data=trout;
run;
```

Output 48.17.1 and Output 48.17.2 show the results.

Output 48.17.1 Listing of TRAINING DB, Pricing Time Series, TUNEFAME=NODES, and TUNECHLI=NODES Options

TRAINTEN DB, Pricing Time Series, TUNEFAME=NODES and TUNECHLI=NODES Options

Obs	DATE	DJ30IN.UHIGH	DJ__30.UHIGH	F.UHIGH	F___I.UHIGH	GM.UHIGH	GM__PP.UHIGH
1	23JUL1997	8199.15	8199.15	41.0625	41.0625	59.1250	59.1250
2	24JUL1997	8174.53	8174.53	42.0000	42.0000	59.2500	59.2500
3	25JUL1997	8200.31	8200.31	41.5000	41.5000	57.8125	57.8125

Obs	HPQ.UHIGH	HWP.UHIGH	IBM.UHIGH	INDUT.UHIGH	INTC.UHIGH	JAVA.UHIGH	JAVAD.UHIGH
1	67.3125	67.3125	107.000	8199.15	90.750	46.9375	46.9375
2	65.8750	65.8750	108.438	8174.53	90.625	46.8750	46.8750
3	66.1250	66.1250	108.438	8200.31	91.125	47.3750	47.3750

Obs	KO.UHIGH	PEP.UHIGH	SPAL.UHIGH	SPALN.UHIGH	SPALNS.UHIGH	SPX.UHIGH	SP_CI.UHIGH
1	70.7500	38.4375	941.800	941.800	941.800	941.800	941.800
2	70.4375	38.0625	941.510	941.510	941.510	941.510	941.510
3	70.9375	38.7500	945.650	945.650	945.650	945.650	945.650

Obs	SP__50.UHIGH	SP___C.UHIGH	SUNW.UHIGH	XOM.UHIGH	XON.UHIGH
1	941.800	941.800	46.9375	63.125	63.125
2	941.510	941.510	46.8750	62.000	62.000
3	945.650	945.650	47.3750	63.000	63.000

Output 48.17.2 Listing of Contents of TROUT for TUNEFAME=NODES and TUNECHLI=NODES Options

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
1	DATE	Num	8	DATE9.	9.	Date of Observation
2	DJ30IN.UHIGH	Num	8			
3	DJ__30.UHIGH	Num	8			
4	F.UHIGH	Num	8			
5	F___I.UHIGH	Num	8			
6	GM.UHIGH	Num	8			
7	GM__PP.UHIGH	Num	8			
8	HPQ.UHIGH	Num	8			
9	HWP.UHIGH	Num	8			
10	IBM.UHIGH	Num	8			
11	INDUT.UHIGH	Num	8			
12	INTC.UHIGH	Num	8			
13	JAVA.UHIGH	Num	8			
14	JAVAD.UHIGH	Num	8			
15	KO.UHIGH	Num	8			
16	PEP.UHIGH	Num	8			
17	SPAL.UHIGH	Num	8			
18	SPALN.UHIGH	Num	8			
19	SPALNS.UHIGH	Num	8			
20	SPX.UHIGH	Num	8			
21	SP_CI.UHIGH	Num	8			
22	SP__50.UHIGH	Num	8			
23	SP___C.UHIGH	Num	8			
24	SUNW.UHIGH	Num	8			
25	XOM.UHIGH	Num	8			
26	XON.UHIGH	Num	8			

For more information about tuning the use of virtual memory, read about TUNE CACHE nodes in the section "TUNE CACHE Option" in the online document *Fame 10 Online Help*.

Example 48.18: Remote Access Using the MCADBS Server

Instead of accessing the local Fame training database, as shown in Example 48.10, this example shows how to access the remote Fame training database that is located on a remote Fame MCADBS server whose host name is "txa006". First, specify an explicit connection by using the CONNECT=YES option. Then name the connection in the AS_NAME= option, specify the host name of the remote MCADBS server in the ON_HOST= option, and specify the service to use in the TO_SERVICE= option. In addition, specify the user name and password for the connection by using the USER= and PASS= options. Designate an input SAS data set (INSETZ) that specifies the tickers to select, and specify your selection by using the WHERE clause in the INSET= option as follows:

```
option validvarname=any;

data insetz;
   length tick $6;
   /* need $6 so DJ30IN is not truncated */

   tick='C'; output;
   tick='CVX'; output;
   tick='DJ30IN'; output;
   tick='F'; output;
   tick='HPQ'; output;
   tick='IBM'; output;
   tick='INTC'; output;
   tick='KO'; output;
   tick='ORCL'; output;
   tick='PEP'; output;
   tick='SPX'; output;
   tick='XOM'; output;
   tick='YUM'; output;
run;

libname lib10 sasefame "C:\PROGRA~1\FAME\util"
        debug=on
        connect=yes to_service="2961" on_host="txa006" as_name="C"
        user="famekff" pass="XXXXXXXXX"
        convert=(frequency=business technique=constant)
        range='07jul1997'd - '25jul1997'd
        inset=( insetz where=tick )
        crosslist=
            ( {adjust, close, high, low, open, volume,
               uclose, uhigh, ulow, uopen, uvolume} );

data trout;
   /* thirteen companies with unique TICKs specified in INSETZ */
   /* Use tr since this is the MCADBS dbid for the training.db */
   set lib10.tr;
   keep DATE IBM: ;     /* only keep IBM for brevity of output results */
run;

title1 'TRAINING DB, Pricing Timeseries for IBM';
title2 'Using INSET with WHERE=TICK.';
proc print data=trout;
run;

proc contents data=trout;
run;
```

Output 48.18.1 and Output 48.18.2 show the results.

Output 48.18.1 Listing of OUT=TROUT Using CROSSLIST= and INSET= Options in the Fame MCADBS Remote TRAINING Data

TRAINING DB, Pricing Timeseries for IBM
Using INSET with WHERE=TICK.

Obs	DATE	IBM.ADJUST	IBM.CLOSE	IBM.HIGH	IBM.LOW	IBM.OPEN	IBM.UCLOSE	IBM.UHIGH
1	07JUL1997	0.5	47.2500	47.7500	47.0000	47.5000	94.500	95.500
2	08JUL1997	0.5	47.8750	47.8750	47.2500	47.2500	95.750	95.750
3	09JUL1997	0.5	48.0938	48.3438	47.6563	48.0000	96.188	96.688
4	10JUL1997	0.5	47.8750	48.0938	47.0313	47.3438	95.750	96.188
5	11JUL1997	0.5	47.8750	48.6875	47.8125	47.9063	95.750	97.375
6	14JUL1997	0.5	47.6250	48.2188	47.0000	47.8125	95.250	96.438
7	15JUL1997	0.5	48.0000	48.1250	46.6875	47.4375	96.000	96.250
8	16JUL1997	0.5	48.8125	49.0000	47.6875	47.8750	97.625	98.000
9	17JUL1997	0.5	49.8125	50.8750	48.5625	48.9063	99.625	101.750
10	18JUL1997	0.5	52.2500	52.6250	50.0000	50.0000	104.500	105.250
11	21JUL1997	0.5	51.8750	53.1563	51.0938	52.6250	103.750	106.313
12	22JUL1997	0.5	51.5000	51.7500	49.6875	50.0313	103.000	103.500
13	23JUL1997	0.5	52.5625	53.5000	51.5938	52.1875	105.125	107.000
14	24JUL1997	0.5	53.9063	54.2188	52.2500	52.8125	107.813	108.438
15	25JUL1997	0.5	53.5000	54.2188	52.8125	53.9688	107.000	108.438

Obs	IBM.ULOW	IBM.UOPEN	IBM.UVOLUME	IBM.VOLUME
1	94.000	95.000	129012	64506
2	94.500	94.500	102796	51398
3	95.313	96.000	177276	88638
4	94.063	94.688	127900	63950
5	95.625	95.813	137724	68862
6	94.000	95.625	128976	64488
7	93.375	94.875	149612	74806
8	95.375	95.750	215440	107720
9	97.125	97.813	315504	157752
10	100.000	100.000	463480	231740
11	102.188	105.250	328184	164092
12	99.375	100.063	368276	184138
13	103.188	104.375	219880	109940
14	104.500	105.625	204088	102044
15	105.625	107.938	146600	73300

Output 48.18.2 Contents of OUT=TROUT Using CROSSLIST= and INSET= Options in the Fame MCADBS Remote TRAINING Data

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
1	DATE	Num	8	DATE9.	9.	Date of Observation
2	IBM.ADJUST	Num	8			
3	IBM.CLOSE	Num	8			
4	IBM.HIGH	Num	8			
5	IBM.LOW	Num	8			
6	IBM.OPEN	Num	8			
7	IBM.UCLOSE	Num	8			
8	IBM.UHIGH	Num	8			
9	IBM.ULOW	Num	8			
10	IBM.UOPEN	Num	8			
11	IBM.UVOLUME	Num	8			
12	IBM.VOLUME	Num	8			

The DEBUG=ON option gives tracing information in the SAS log that shows the Fame CHLI commands that are used to communicate with the remote server. This debugging information can be useful in explaining the communication between the client and server machines. An abbreviated version of the SAS log follows:

```
NOTE: Libref LIB10 was successfully assigned as follows:
      Engine:        SASEFAME
      Physical Name: C:\PROGRA~1\FAME\util
155
156        data trout;
157           set lib10.tr;
NOTE: The SASEFAME engine is using Version 11.43000 of the HLI.
len4=0
FAME COMMAND line 913 is:
OPEN <ACCESS READ> tr ON C; OVERWRITE ON; GLUE DOT;
ITEM ALIAS ON
STATUS from first OPEN is: 0
FAME COMMAND line 1255 is: GLUE DOT; LOOP FOR LCV IN CROSSLIST
({C,CVX,DJ30IN,F,HPQ,IBM,INTC,KO,ORCL,PEP,SPX,XOM,YUM},{ADJUST,CLOSE,HIGH,LOW,
OPEN,VOLUME,UCLOSE,UHIGH,ULOW,UOPEN,UVOLUME}); NEW WORK'LCV = LCV; END LOOP;
STATUS from LOOP for LCV in CROSSLIST is: 0
setting the dbkey to the wkkey which is: 0
STATUS from cfmopcn is: 0
cfmopdc dbname line 1459 is: tr
STATUS from cfmopdc is: 0
C.ADJUST -- SERIES (NUMERIC by BUSINESS)
FAME COMMAND line 2300 is: IGNORE ON;
C.CLOSE -- SERIES (NUMERIC by BUSINESS)
.
.
.
YUM.VOLUME -- SERIES (NUMERIC by BUSINESS)
FAME COMMAND line 2300 is: IGNORE ON;
entering fmoinfo, nobs=-1
C.ADJUST -- SERIES (NUMERIC by BUSINESS)
```

```
C.CLOSE -- SERIES (NUMERIC by BUSINESS)
.
.
.
IBM.ADJUST -- SERIES (NUMERIC by BUSINESS)
IBM.CLOSE -- SERIES (NUMERIC by BUSINESS)
IBM.HIGH -- SERIES (NUMERIC by BUSINESS)
IBM.LOW -- SERIES (NUMERIC by BUSINESS)
IBM.OPEN -- SERIES (NUMERIC by BUSINESS)
IBM.UCLOSE -- SERIES (NUMERIC by BUSINESS)
IBM.UHIGH -- SERIES (NUMERIC by BUSINESS)
IBM.ULOW -- SERIES (NUMERIC by BUSINESS)
IBM.UOPEN -- SERIES (NUMERIC by BUSINESS)
IBM.UVOLUME -- SERIES (NUMERIC by BUSINESS)
IBM.VOLUME -- SERIES (NUMERIC by BUSINESS)
.
.
.
YUM.UOPEN -- SERIES (NUMERIC by BUSINESS)
YUM.UVOLUME -- SERIES (NUMERIC by BUSINESS)
YUM.VOLUME -- SERIES (NUMERIC by BUSINESS)
entering fmoinfo, nobs=-1
entering fmoinfo, nobs=8637
158        run;

entering fmoinfo, nobs=8637
inside fmoinfo, nobs=8637
NOTE: There were 8637 observations read from the data set LIB10.TR.
NOTE: The data set WORK.TROUT has 8637 observations and 144 variables.
```

Because you specify the DEBUG=ON option, the SAS log includes the Fame commands and reports the status of the Fame CHLI commands that are issued during the execution of the SAS DATA step. The first Fame command shown is **OPEN**; it is important to note that instead of using training in the SAS SET statement, it is necessary to use the database ID, tr. For the MCADBS server, a list of databases is given in the *mcadbs.config* file, which for the host txa006 contains the following information:

```
# The databases to open
OPEN %OL% %FAME%\util\training.db TR
                              # Clients refer to this as TR.
```

The first **OPEN** command listed in the SAS log (inside the **FAME** command) refers to the named connection, C:

```
OPEN <ACCESS READ> tr ON C;
```

So the connection is named C, which is specified in the AS_NAME= option in the SASEFAME LIBNAME statement.

References

DRI/McGraw-Hill (1997). *DataLink*. Lexington, MA: DRI/McGraw-Hill.

DRI/McGraw-Hill Data Search and Retrieval for Windows (1996). *DRIPRO User's Guide*. Lexington, MA: DRI/McGraw-Hill.

IHS Global Insight (2009). "Global Economic Data." Available at http://www.ihs.com/products/global-insight/industry-analysis/financial/global-economic-data.aspx.

Organisation for Economic Co-operation and Development (1992a). *Annual National Accounts, Vol. 1: Main Aggregates Content Documentation for Magnetic Tape Subscription*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992b). *Annual National Accounts, Vol. 2: Detailed Tables Technical Documentation for Magnetic Tape Subscription*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992c). *Main Economic Indicators Database Note*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992d). *Main Economic Indicators Inventory*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992e). *Main Economic Indicators OECD Statistics on Magnetic Tape Document*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992f). *OECD Statistical Information Research and Inquiry System Magnetic Tape Format Documentation*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992g). *Quarterly National Accounts Inventory of Series Codes*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992h). *Quarterly National Accounts Technical Documentation*. Paris: OECD.

SunGard Solutions for Data Management (2009a). *FAME 10 Online Help*. Ann Arbor, MI: SunGard Solutions for Data Management. https://fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009b). *FAME Command Reference for Release 9 and Earlier*. Ann Arbor, MI: SunGard Solutions for Data Management. https://fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009c). *FAME Functions for FAME Release 9 and Earlier*. Ann Arbor, MI: SunGard Solutions for Data Management. https://fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009d). *Guide to FAME Database Servers*. New York: SunGard Solutions for Data Management. https://fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009e). *Reference Guide to Seamless C HL*. Ann Arbor, MI: SunGard Solutions for Data Management. https://fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009f). *User's Guide to FAME*. Ann Arbor, MI: SunGard Solutions for Data Management. https://fame.sungard.com/support.html.

Subject Index

Syntax Index