CHAPTER

10

# OpenVMS: DECnet Access Method



## Tasks That Are Common to SAS/CONNECT and SAS/SHARE

*System Administrator or User*

To use the DECnet access method with an OpenVMS host for SAS/CONNECT and SAS/SHARE, perform these tasks:

1. Verify that you have met all your site and software requirements.
2. Verify that you know how to set variables in SAS software.
3. Set the desired SAS/CONNECT and SAS/SHARE environment variables.

## System and Software Requirements for SAS/CONNECT and SAS/SHARE

Ensure that the following conditions have been met:

1. DECnet has been installed at both the local and remote hosts at your site.
2. SAS software is installed on both the local and remote hosts at your site.
3. Each local and remote process that is running on an OpenVMS system has the TMPMBX and NETMBX privileges.

## System Requirements for SAS/SHARE Only

Ensure that the user has the SYSNAM privilege to start the SAS/SHARE server.

## Setting Variables in SAS

You may need to set specific variables in SAS to allow the desired connections with SAS/SHARE using the DECnet communications access method.

Consult with your network administrator to determine what variables must be set and what values to assign to them.

You may specify a variable by using one of the following forms:

- an environment variable at a DCL prompt:

  *variable*:==*value*

  Example:

  ```
  SASSECUR:==_SECURE_
  ```

- a SAS macro variable in a SAS session:

  %LET *variable*=*value*;

  Example:

  ```
  %let sassecur=_secure_;
  ```

Values for these variables can contain up to eight characters, consisting of alphanumeric characters, the percent sign (%), the dollar sign ($), the pound sign (#), the at sign (@), and the underscore.

If you set multiple forms of the same variable, the order of precedence follows:

SAS macro variable in SAS session

SAS macro variable in AUTOEXEC file

Environment variable.

## Setting Security for SAS/CONNECT and SAS/SHARE

For SAS/CONNECT, you must supply identifying information to sign on without a script to a remote host running a spawner program. A SAS/SHARE server, running secured, requires identification from each connecting client. The next two sections outline the version-specific methods for specifying client identification for SAS/CONNECT and SAS/SHARE.

### Providing Client Identification in a Version 8 Session

In Version 8, you provide client identification to a SAS/CONNECT remote host or a SAS/SHARE server using the USER= and PASSWORD= options. These options are valid in the following statements:

**SIGNON**

**RSUBMIT**

**LIBNAME**

**PROC SQL**
Connect to Remote

**PROC OPERATE**
(in the PROC statement)
set server
stop server
quiesce server
start server
display server

Specifying client identification in the SASSECUR variable is still accepted but is not recommended in Version 8. The USER= and PASSWORD= options take precedence over the client SASSECUR variable when both are specified. For example, a SAS/SHARE client's execution of a LIBNAME statement with values assigned to the USER= and PASSWORD= options would override a SASSECUR variable setting in the same client SAS session.

Here is the syntax and definitions for these options:

**USER** | **USERNAME** | **USERID** | **UID**=*username* | _PROMPT_

**PASSWORD** | **PASSWD** | **PASS** | **PWD** | **PW**=*password* | _PROMPT_

Specifying these options allows a user on the local host whose username and password have been verified to access the remote host.

*username*
is a valid userid for the remote host and is thus host-dependent in form. If the value contains blanks or special characters, it must be enclosed in quotes.

*password*
is the password, if any, required for authentication of the supplied username. This value will not be echoed in the SAS log. If the value contains blanks or special characters, it must be enclosed in quotes.

_PROMPT_
specifies that the SAS System prompts the client for *username* and *password*.

*Note:* The values provided when prompted must NOT be quoted. △

Specifying USER=_PROMPT_ and omitting the PASSWORD= specification will cause SAS to prompt you for both userid and password.

This is especially useful for allowing the SAS statements containing the USER= and PASSWORD= options to be copied and otherwise effectively reused by others.

For SAS/SHARE, the values supplied for the USER= and PASSWORD= options are valid for the duration of the remote host connection. Additional accesses of the remote host while the connection to that host is still in effect do not require re-supplying of the USER= and PASSWORD= options. For example, while the first connecting library assign to a SAS/SHARE server may require specification of the options, subsequent assigns to the same server will not need specification of these options as long as the original connection is in effect. A subsequent re-connect to the same server or connect to a different server would require re-supplying of the USER= and PASSWORD= options.

Here is a Version 8 example for SAS/SHARE:

```
libname test 'prog2 a' user=joeblue password="2muchfun" server=share1;
```

For SAS/CONNECT, these values are valid until SIGNOFF.
Here is a Version 8 example for SAS/CONNECT:

```
signon rmthost user=joeblack password=born2run;
```

As a security precaution, PASSWORD= field entries echoed in the log are replaced with Xs. If _PROMPT_ was specified for entering the password, the entry would not be displayed on the screen as it is typed.

### SAS/SHARE SASSECUR Variable

SASSECUR:==_PROMPT_ | *userid.password* |
_SECURE_

You may set the SASSECUR variable to provide security for the SAS/SHARE server, allowing access to clients whose userids and passwords have been verified.
Values that you set at a SAS/SHARE client are

_PROMPT_
must be set at the SAS/SHARE client.
_PROMPT_ causes SAS to prompt the user for userid and password information. When prompted for a password, the input field is not displayed. Choosing to prompt for the userid and password provides more security than assigning the userid and password to the variable.

*userid.password*
must be set at the SAS/SHARE client.
*userid.password* specifies both the userid and password. Assigning the userid and password directly to the SASSECUR variable at the SAS/SHARE client may inadvertently publicize this information and compromise the security of the SAS/SHARE server. Assigning the value to the variable in a file allows anyone to check it.

_SECURE_
must be set at the SAS/SHARE server.
The _SECURE_ value for the SASSECUR variable requires a SAS/SHARE client to supply a valid userid and password to the remote host on which the server is running in order to allow client access to the server.
Specify the SASSECUR variable before you create a server.

Examples:

```
SASSECUR:==_PROMPT_
SASSECUR:==bass.timego
SASSECUR:==_SECURE_
```

See “Setting Variables in SAS” on page 156 for examples of the forms you can use to specify the SASSECUR variable.

# SAS/CONNECT

## Local Host Tasks

*User or Applications Programmer*
To connect an OpenVMS local host to a remote host, perform these tasks at the local host:

1. Specify the communications access method.
2. Specify a remote host to connect to.
3. Sign on to the remote host.

## Specifying the DECnet Communications Access Method

You must specify the DECnet communications access method to make a remote host connection. Use the following syntax:

```
OPTIONS COMAMID=access-method-id;
```

where COMAMID is an acronym for Communications Access Method Identification. *access-method-id* identifies the method used by the local host to communicate with the remote host. DECnet (an acronym for Digital Equipment Corporation's Networking architecture) is an example of *access-method-id*.

Alternatively, you may set this option at a SAS invocation or in the SAS configuration file.

Example:

```
options comamid=decnet;
```

## Specifying the Remote Host Name

To connect an OpenVMS local host to a remote host, use the following syntax:

```
REMOTE=remote-session-id
```

where *remote-session-id* can be expressed as a node name or as ACI (Access Control Information), which is represented as:

```
nodename"username password" ::
                  |"? ?" ::
                  |"username ?" ::
                  |"? password" ::
```

If proxy access is enabled on the DECnet network, specify only the remote node name. Ask your network administrator if proxy access is enabled on the DECnet network. Proxy access precludes a need for you to assign your username and password to the ACI. Otherwise, include the username and password information in the ACI or substitute that information for the one or two question marks (?) to signify prompting for either username or password or both username and password.

*Note:* If a password is not required for an account, you may omit the password from the ACI. △

Instead of hard-coding username and password values, you are encouraged to use prompting as a security aid.

If you specify a username and password in the ACI, you must assign them to a SAS macro variable.

To connect an OpenVMS local host to a remote host, you may specify the remote host in either of the following forms:

- remote node name
- SAS macro variable for the remote node
- DCL symbol
- OpenVMS logical name format
- SAS$CONN command file.

Examples:

□ OpenVMS logical name format:

```
$ define alias "node""username password""::"
$ sas/comamid=decnet/remote=alias
$ options remote=alias;
```

Example of OpenVMS logical name format:

```
$ define fred"monarch""bass time2go""::"
$ sas/comamid=decnet/remote=fred
```

□ SAS macro variable format:

```
$ sas/comamid=decnet
1? %let alias=node"username password" ::;
2? options remote=alias;
```

Example of SAS macro variable format:

```
$ sas/comamid=decnet
1? %let fred=monarch"bass time2go" ::;
2? options remote=fred;
```

□ DCL symbol format:

```
$ alias:= "node""username password""::"
$ sas/comamid=decnet/remote=alias
```

Example of DCL symbol:

```
$ fred:= "monarch""bass time2go""::"
$ sas/comamid=decnet/remote=fred
```

In all three examples, the alias is FRED, the node is MONARCH, the username is BASS, and the password is TIME2GO.

*Note:* The colons (::) are optional, but the quotation marks must be used. You may use a question mark (?) to cause the local host to prompt for the username and password. Do not use the same value for the alias and the node name; they must be different. △

If you have defined all three forms of the remote host name, the order of precedence from highest to lowest is

□ OpenVMS logical name

□ SAS macro variable

□ DCL symbol.

## Signing On to the Remote Host

To complete your sign on to the remote host, enter the SIGNON statement, as follows:

```
signon user=_prompt_;
```

To set security at the remote host, specify valid values for the USER= and PASSWORD= options in the SIGNON statement. For details, see “Providing Client Identification in a Version 8 Session” on page 156.

Although no errors result from specifying a script file for sign on and sign off, you waste processing time. If you defined a RLINK fileref, when you sign on,

SAS/CONNECT processes and loads the script file that is identified by the RLINK fileref but DECnet will then ignore the script. If you do not want to free the RLINK fileref, you can save processing time by using the NOSCRIPT option in the SIGNON and SIGNOFF statements, shown as follows:

```
signon noscript;
.
.
.
signoff noscript;
```

## Local Host Example

The following example illustrates the statements that you specify in a SAS$CONN.COM command file at an OpenVMS local host to set the current working directory and to start a local SAS session.

```
set def disk:[bass.work]
sas/dmr/comamid=decnet
```

The following example illustrates the statements that you specify in a SAS session to access a remote host with the DECnet access method.

```
$ sas/comamid=decnet/remote=rmthost
1? signon user=_prompt_;
```

The statements in the first line invoke a SAS session, specify the DECnet access method, and specify a remote host with the DCL symbol format that is identified by RMTHOST. The USER= option in the SIGNON statement specifies that the connecting local host be prompted for a userid and a password that are valid on the remote host.

## Remote Host Tasks

*System Administrator*

You must perform these tasks at the remote host to allow a connection from a local host:

1. Create the SAS$CONN.COM command file for signing on.
2. Optionally set several remote host options.

## Creating the SAS$CONN.COM Command File for Signing On

The DECnet access method's ability to automatically create its remote partner (also known as connecting to an object) eliminates the need for using a script file for signing on or signing off. SAS uses the SAS$CONN.COM command file, which is provided in the SAS$ROOT:[TOOLS] directory, to create the SAS$CONN object.

This connection causes DECnet to create a process on the specified remote host by using proxy access or by using the username and password that is supplied as the ACI (Access Control Information). DECnet then finds and executes the command file that is associated with the object name SAS$CONN. This command file contains DCL commands, one of which invokes the remote SAS session.

You may use one of two methods to cause DECnet to associate a command file with the SAS$CONN object name.

- Put the command file SAS$CONN.COM in the user's default login directory.

The advantage of this method is that each user can tailor the command file, thus producing many copies of the command file throughout the system.

- Ask the system administrator to specify the location of the command file using the Network Control Program (NCP).

  The advantage of this method is that you need only one copy of the file on the system. However, if SAS$CONN is a known object, the system uses the declared file and ignores any tailored copies of SAS$CONN.COM in a user's login directory.

A sample SAS$CONN.COM command file follows:

```
set def disk:[user.directory]
sas/dmr/comamid=decnet/noterminal/no$syntaxcheck
```

where you supply the values for *user.directory*.

The first line sets the default directory to the directory where you want to run SAS; the second line invokes the remote SAS session.

## Setting Options at the Remote Host

Although sign-on script files are not used for the DECnet access method, you still may set these remote host options at the remote host.

NO$SYNTAXCHECK
allows the continuation of statement processing at the remote host regardless of syntax error conditions.

This option is valid as part of a configuration file, at a SAS invocation, or in an OPTIONS statement.

NOTERMINAL
specifies whether a terminal is attached at SAS invocation. If NOTERMINAL is specified, requestor windows are not displayed.

Setting NOTERMINAL at the remote host is advisable so that no terminal is associated with the remote session. This option prevents SAS from displaying error messages and dialog boxes on the remote host, which requires user intervention.

This option is valid as part of a configuration file or SAS invocation.

See *SAS Language Reference: Dictionary* for details about this option.

An example of configuration file entries for a remote host using the DECnet method follows:

```
no$syntaxcheck
noterminal
```

## Remote Host Example

The following example illustrates the SAS$CONN.COM file entries for an OpenVMS remote host:

```
sas/comamid=decnet/no$syntaxcheck/noterminal
```

# SAS/SHARE

## Client Tasks

*System Administrator, Applications Programmer, and User*

To prepare for accessing a SAS/SHARE server, perform the following tasks:

1. Set security for connecting clients.
2. Specify the DECnet access method.
3. Specify a server name.

## Setting Security for Connecting Clients

Requiring connecting clients to supply a valid userid and password enforces server security. At the client, set the preferred security method for specifying a userid and password that are valid on the server host. For details, see "Setting Security for SAS/CONNECT and SAS/SHARE" on page 156.

## Specifying the DECnet Communications Access Method

You must specify the DECnet communications access method at the client before you access a server.

Use the following syntax to specify the DECnet access method at each connecting client:

```
OPTIONS COMAMID=access-method-id;
```

where COMAMID is an acronym for Communications Access Method Identification. *access-method-id* is an abbreviation for the method used by the client to communicate with the server. DECnet (an acronym for Digital Equipment Corporation's Networking architecture) is an example of an *access-method-id*.

Example:

```
options comamid=decnet;
```

The server is accessed using the DECnet access method.

You may specify the COMAMID option in an OPTIONS statement, at a SAS invocation, or in a SAS configuration file.

Additionally you may use the COMAUX1 and COMAUX2 options to designate auxiliary communications access methods. See Table 1.3 on page 10 for the supported access methods by host. If the first method fails to access a server, the second method will be attempted, and so on. You can specify up to two auxiliary access methods, depending on the number of methods that are supported between client and server hosts.

COMAUX options can be specified only in an OPTIONS statement. The syntax for the COMAUX option follows:

```
OPTIONS COMAUX1=alternate-method;
```

An example of the OPTIONS statements for an OpenVMS client that is connecting to a server follows:

```
options comamid=decnet;
options comaux1=tcp;
```

If the server cannot be reached with the DECNET method, a second attempt is made with the TCP/IP access method.

## Specifying a Server Name

If the client and server sessions are running on different network nodes, you must include the DECnet node in the server identifier in the PROC SERVER command or in the LIBNAME and PROC OPERATE statements as follows:

SERVER=*node.server-id*

This representation is known as a two-level server name.

*node* must be a valid DECnet node name. If the server and the client sessions are running on the same node, you may omit the node name.

If the DECnet node name is not a valid SAS name (for example, it contains a digit or its length exceeds eight characters such as when you assign ACI to the node name) you have several options for validating the node name:

- Assign the name of the server's node to a SAS macro variable, then use the name of the macro variable for *node* in the two-level server name.
- Assign the name of the server's node to an OpenVMS logical name, then use the name of that OpenVMS logical name for *node* in the two-level server name.
- Assign the name of the server's node to a DCL symbol, then use the DCL symbol for *node* in the two-level server name.

The access method evaluates the node name, in this order of precedence:

- acceptable node name
- OpenVMS logical name
- SAS macro variable
- DCL symbol.

Examples:

- OpenVMS logical name format:

```
$ define alias "node"["userid password""]
```

Example of OpenVMS logical name format:

```
$ define fred "4sonar""bass time2go""
```

- SAS macro variable format:

```
1? %let alias=node ["username password"];
```

Example of SAS macro variable format:

```
1? %let fred=4sonar"bass time2go";
```

*Note:* Do not use an ampersand (&) in a two-level name. An ampersand causes the macro variable to be resolved by the SAS parser prior to syntactic evaluation of the SERVER= option. The access method evaluates the node name in a two-level server name. △

See *SAS Language Reference: Dictionary* for details about SAS naming rules. See *SAS/SHARE User's Guide* for details about the PROC SERVER, PROC OPERATE, and LIBNAME statements.

- DCL symbol format:

```
$ alias:="node"["username password""]
```

Example of DCL symbol:

```
$ fred:="4sonar""bass time2go""
```

- USER= and PASSWORD= options to applicable SAS/CONNECT statement:

  *statement* USER=*username* PASSWORD=*password*;

Example of USER= and PASSWORD= options to SIGNON:

```
signon user=bass password=time2go;
```

In the preceding examples, the alias is FRED, the node is 4SONAR, the username is BASS, and the password is TIME2GO.

For details about starting a SAS/SHARE server, see *SAS/SHARE User's Guide*.

## Client Example

The following example illustrates the statements that you specify in an OpenVMS client SAS session to access a server with the DECnet access method:

```
options comamid=decnet;
libname sasdata [edc.prog2.sasdata] user=_prompt_ server=rmthost.share1;
```

The DECnet access method is declared. The LIBNAME statement specifies the data library that is accessed through the server, which is specified by the two-level server name RMTNODE.SHARE1 by means of a prompt for a username and a password that are valid on the server

## Server Tasks

*Server Administrator*

To set up a secure server and to make it accessible to a client, perform the following tasks:

- Assign a privileged account to the server.
- Set DECNET access method security.
- Specify the communications access method.
- Specify the server name.

## Assigning a Privileged Account to the Server

Ask your system administrator to verify that the account in which the secured server runs has one of these privileges: GRPPRV, SYSPRV, or BYPASS.

*Note:* If the server has GRPPRV privilege, only users in the same UIC group as the server will be able to have access to files validated by the server. Users that are not in the same UIC group as the server will not be able to use the server. △

## Setting DECnet Access Method Security

Before you can create a secure SAS/SHARE server, you must make the access method secure by setting the SASSECUR macro or environment variable to _SECURE_. See “SAS/SHARE SASSECUR Variable” on page 158 for information about setting the SASSECUR variable.

## Specifying the DECnet Communications Access Method

You must specify the DECnet communications access method at the server before you create a SAS/SHARE server.

Use the following syntax to specify the DECnet access method at the server:

```
OPTIONS COMAMID=access-method-id;
```

where COMAMID is an acronym for Communications Access Method Identification. *access-method-id* is an abbreviation for the method used by the server to communicate with the client. DECnet (an acronym for Digital Equipment Corporation's Networking architecture) is an example of an *access-method-id*.

For a server that is running on a host on which only one communications access method is available, use only the COMAMID option.

Example:

```
options comamid=decnet;
```

The server will be available only to SAS/SHARE sessions that use the DECnet access method.

You may specify the COMAMID option in an OPTIONS statement, at a SAS invocation, or in a SAS configuration file.

If, however, the host on which a server runs supports multiple access methods, you may specify up to two auxiliary access methods by which clients may access the server using the COMAUX1 and COMAUX2 options. See Table 1.3 on page 10 for the supported access methods by host.

All of the access methods initialize when the server initializes. The activation of multiple access methods makes a server available to several groups of clients, each using a different communications access method simultaneously.

COMAUX options can be specified in an OPTIONS statement only. The syntax for the COMAUX option follows:

```
OPTIONS COMAUX1=alternate-method;
```

An example of the OPTIONS statements for a server that is running on an OpenVMS host follows.

```
options comamid=decnet;
options comaux1=tcp;
```

When the server starts, all of the communications access methods are initialized. The server is simultaneously available to client sessions that use the DECnet access method as well as to clients that use the TCP/IP access method.

## Specifying a Server Name

If the client and server sessions are running on different network nodes, you must include the DECnet node in the server identifier in the PROC SERVER command. See "Specifying a Server Name" on page 164 for server-naming rules.

## Server Example

The following example illustrates the statements that you specify in a SAS session on the OpenVMS host at which you start a server:

```
%let sassecur=_secure_;
options comamid=decnet;
proc server id=share1 authenticate=req;
run;
```

The SASSECUR variable requires clients to submit a userid and a password that are valid on the server. The DECnet access method is declared and a server with the *server-id* SHARE1 is started on the OpenVMS host. The additional options in the PROC SERVER statement allow only validated clients to access the server.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 643.

**Communications Access Methods for SAS/CONNECT and SAS/SHARE Software, Version 8**



ISBN 1–58025–479–9



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

1st printing, September 1999