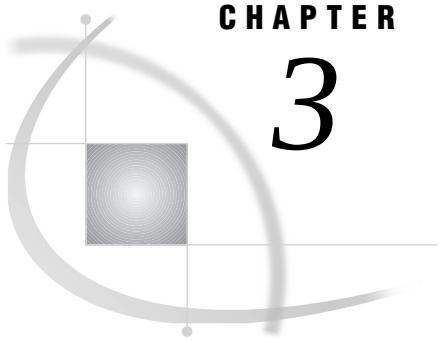

CHAPTER

3

Syntax for Compute Services

Introduction **21**
Dictionary **21**

Introduction

This chapter describes the statements and commands you can use with SAS/ CONNECT Software.

Dictionary

RSUBMIT Command and RSUBMIT Statement

Submit statements that are entered on the local host to a remote session for processing.

Local

Syntax

RSUBMIT <*remote-session-id*><CONNECTWAIT=YES|NO> <MACVAR=*value*>
<CONNECTSTATUS=YES|NO> <SYSRPUTSYNC=*value*>
<USER=*username*|_PROMPT_> <PASSWORD=*password*|_PROMPT_>
<PERSIST=YES|NO> <SCRIPT=*value*>;

Syntax Description

The options characterize the environment in which statements are submitted to a remote session for processing.

Details

The RSUBMIT command and the RSUBMIT statement cause SAS programming statements that are entered in the local environment to execute on a remote SAS session. The RSUBMIT command differs from the standard SUBMIT command because statements execute on the remote host. Even though the statements execute in the

remote environment, all results and output are available to your local SAS session log and output as they would be if you executed the program in the local SAS session. If the RSUBMIT is synchronous, then all results and output are displayed in your local SAS session. If the RSUBMIT is asynchronous, then you can use the RGET and RDISPLAY commands and statements to retrieve and view the results.

The primary difference between the command and the statement is that the command can be used only from a windowing environment session or within the DM statement. The RSUBMIT statement can be used in any type of SAS session on the local host.

Execute the RSUBMIT command from the command line of the local Program Editor window. Or you can embed the RSUBMIT command within a DM statement, which treats commands as if they were issued from a windowing environment command line. You can also use the KEYS window to assign the RSUBMIT command to a key. See the online help in the SAS windowing environment for details about the KEYS window.

RSUBMITs are processed in either synchronous or asynchronous modes.

synchronous
: This means that you do not regain local control until the RSUBMIT has completed. Synchronous processing is the default processing mode.

asynchronous
: This allows you to execute statements in a remote SAS session in parallel to your local session. You immediately regain control of your local session to continue with local processing or remote processing to another host.

The autosignon feature of RSUBMIT includes an implicit SIGNON in the absence of a current connection. The RSUBMIT command or statement automatically executes a SIGNON and uses any globally set SAS/CONNECT options along with any connection options that are specified with RSUBMIT. The autosignon first signs on to the remote session and then executes the RSUBMIT. All SIGNON options are also valid for RSUBMIT. Furthermore, a default automatic SIGNOFF occurs at the conclusion of RSUBMIT, but the SIGNOFF can be overridden by specifying PERSIST=YES.

Note: Any connection information that is specified in RSUBMIT for autosignon will be in effect for the entire connection. For example, if you specify WAIT=NO in an RSUBMIT that automatically signs on, then asynchronous RSUBMITs will be the default for the entire connection, but these RSUBMITS can be overridden in individual RSUBMITS. △

The RSUBMIT command can be used to execute most types of SAS programs on the remote host.*

The RSUBMIT statement is particularly useful for running SAS/CONNECT from an interactive line-mode session or as a non-interactive job. The RSUBMIT and the ENDRSUBMIT statements enable you to include statements that should be processed by the local host in the same file as statements that are to be processed by the remote host. The statements for the remote host are enclosed between the RSUBMIT and the ENDRSUBMIT statements. All of the other statements in the program are processed by the local host when you execute the program.

The following template can be used to build a file that includes statements for both the remote and local hosts in the same program:

```
statements for local host
rsubmit;
   statements for remote host
endrsubmit;
```

* You should not remote submit windowing procedures (such as SAS/FSP or SAS/AF procedures) or Version 5 full-screen procedures (such as the Version 5 DATASETS procedure).

Note: The DOWNLOAD and the UPLOAD procedures must be executed by using the RSUBMIT command or the RSUBMIT statement. You cannot execute them by using the SUBMIT command. △

The following are optional in the RSUBMIT command/statement. Any combination of these options may be used:

remote-session-id
CONNECTREMOTE=*remote-session-id*
REMOTE=*remote-session-id*
PROCESS=*remote-session-id*
is the name of the session where you want to submit the statements when you have multiple SAS/CONNECT sessions that are active. If you have only one active session, *remote-session-id* is not needed. When you have multiple remote sessions that are active and you omit this option, the statements are remote-submitted to the current remote session. The current remote session is the one that is specified in the most recently successful CONNECTREMOTE= system option, SIGNON command/statement, RGET command/statement, or RSUBMIT command/statement.

PROCESS= was made an alias for REMOTE= in order to give you the option of differentiating between an RSUBMIT to a remote session on a local host (MP CONNECT) and an RSUBMIT to a remote session on a remote host. REMOTE= and PROCESS= can be used interchangeably.

CONNECTWAIT=YES|NO
WAIT=YES|NO
specifies whether this particular RSUBMIT is to be executed synchronously or asynchronously. Synchronous processing indicates that you will wait for the remote processing to complete before regaining control in the local SAS session. WAIT=YES is the default processing technique for RSUBMIT.

In asynchronous processing, when the RSUBMIT begins to execute on the remote host, you regain control of your local SAS session to continue local processing or to use RSUBMIT to other remote sessions.

If the WAIT= option to RSUBMIT is omitted, the value assigned to WAIT=, if any, that is specified on SIGNON is used. Otherwise, the WAIT= global option is queried, and its value is used. The default is to execute synchronously.

The value for the WAIT= option must be either of these:

YES|Y indicates a synchronous RSUBMIT.

NO|N indicates an asynchronous RSUBMIT.

If WAIT=NO is specified, it will also be useful to specify the MACVAR= option. This will allow you to test the status of the current asynchronous RSUBMIT by determining whether it has completed or is still in progress.

When %SYSRPUT executes within a synchronous (WAIT=YES) remote submit, the macro variable is defined to the local SAS session as soon as it executes.

When %SYSRPUT is executed within an asynchronous (WAIT=NO) remote submit, the macro variable is not set in the local session until a synchronization point. This is the default. See “%SYSRPUT Statement” on page 29 for more details about synchronization points.

If WAIT=NO is specified and an autosignon is performed, an automatic SIGNOFF will not occur unless PERSIST=NO is also specified.

MACVAR=*value*
specifies the name of the macro variable to associate with this remote session. If specified in the RSUBMIT command/statement, the MACVAR= option overrides

any previous MACVAR= specifications for this remote session. The macro variable is NOT set if the RSUBMIT command fails due to incorrect syntax. Other than this one exception, the macro variable (*value*) is set at the completion of the RSUBMIT block. It will have one the following values:

0 indicates that the RSUBMIT is complete.

1 indicates that the RSUBMIT failed to execute.

2 indicates that the RSUBMIT is still in progress.

Note: If a synchronous RSUBMIT (WAIT=YES) is issued while an asynchronous RSUBMIT (WAIT=NO) is still in progress, all spooled log and output statements are merged into the local log and output windows. Then the RSUBMIT continues at whatever point it is at as if it were synchronous. That is, the user does not regain control until the RSUBMIT has completed. If you don't want this to happen, use the MACVAR= option in the SIGNON or the RSUBMIT statements so that you can check the progress of RSUBMIT without causing it to execute synchronously. △

CONNECTSTATUS=YES|NO
STATUS=YES|NO
specifies the setting for the display of the status window for this RSUBMIT only.

The value for this option must be one of the following:

YES|Y status window is displayed for file transfers within *this* RSUBMIT.

NO|N status window is NOT displayed for file transfers within *this* RSUBMIT.

If this option is omitted from the RSUBMIT statement, the value (if any) that is specified in the SIGNON statement is used. If not specified in either the RSUBMIT or the SIGNON statement, the CONNECTSTATUS= global option is queried, and its value is used. To display the Transfer Status window is the default. Again, the STATUS option in the RSUBMIT statement only affects transfers for that specific RSUBMIT.

SYSRPUTSYNC=*value*
allows you to override the default behavior so that you can force the %SYSRPUT macro variables to be set in the local SAS session when executed rather than waiting until the sync point. See the %SYSRPUT statement for more details about sync points.

Note: This option is useful only when an asynchronous (WAIT=NO) remote submit is executed; otherwise, it is ignored. △

The *value* for this option must be one of the following:

YES the user is able to override the default asynchronous remote submit behavior, and forces the macro variables to be defined as soon as %SYSRPUT executes.

NO the macro variables are not set in the local session until a *synchronization point*.

USER|USERNAME|USERID|UID=*username*|_PROMPT_
Valid values that can be assigned to USER are:

username
For details about a valid username, see "Username and Password Naming Conventions" on page 25.

PROMPT
a secure method, specifies that SAS prompt the user for a valid username.

PASSWORD | PASSWD | PWD | PW=*password* | _PROMPT_
For the autosignon feature only, specifies the password of the remote host. The platform on which the remote host runs can also affect password naming conventions. For details about password naming conventions imposed by the host, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

Valid values for PASSWORD are:

password
For details about a valid password, see "Username and Password Naming Conventions" on page 25.

PROMPT
a secure method, specifies that SAS prompt the user for a valid password.

PERSIST = YES | NO
For the autosignon feature only, specifies whether a signoff is automatically executed after the SIGNON and RSUBMIT have completed.

YES	A connection to the remote session on the local host persists, which means that a signoff is not automatically performed after the SIGNON and RSUBMIT have completed. A YES setting eliminates the need to sign on for subsequent task processing. A persistent connection to the remote session on the local host terminates when you perform an explicit SIGNOFF.
NO	A connection to the remote session does not persist. A signoff is automatically performed after the SIGNON and RSUBMIT have completed. A NO setting requires that you explicitly sign on for subsequent task processing. The default is NO. If WAIT=NO is specified and an autosignon is performed, an automatic SIGNOFF will not occur unless PERSIST=NO is also specified.

SCRIPT=*value*
For the autosignon feature only, specifies the script file for use during an autosignon by means of RSUBMIT. It may either be a fileref or a quoted, fully-qualified pathname. If the fileref, the filespec, and the SCRIPT= option are specified, the last specification overrides and takes precedence over the others.

When the RSUBMIT command executes, the usual SAS log messages for the remote SAS System display in your local LOG window. When the link has been successfully established, the following message is displayed:

```
NOTE: REMOTE SIGNON TO remote-session-id
COMPLETE.
```

Username and Password Naming Conventions

Each username and password is limited to 256 characters that follow these conventions:

- Mixed case is allowed.
- A null value, which is no value, that is delimited with quotation marks is allowed.
- Quotation marks must surround values that contain one or more spaces.
- Quotation marks must surround values that contain one or more special characters.
- Quotation marks must surround values that contain one or more quotation marks.

Examples:

```
user=joe password=Born2run
user=joe password='' # null space specified by contiguous quotation marks
user='joe black' password='Born 2 run'
user='joe?black' password='Born 2 run'
user='apexdomain\joe' password=born2run # Win NT username
user='"crazy joe"' pw=_prompt_;
user=_prompt_;
```

Example

Suppose you want to use the remote system to execute a SAS program that calculates summary statistics from variables in a very large SAS data set and then download the summary statistics to your local session. You enter the following program in the Program Editor window of your local session:

```
libname remtdata 'external-file-name';
  proc summary data=remtdata.clinic;
     class diagnose;
     var age income visits;
     output out=sumstat
        n= mean= mage mincome mvisits;
   run;

   proc download data=sumstat out=summary;
   run;
```

To execute the program on the remote system, enter **RSUBMIT** on the command line of the Program Editor window. Alternatively, you can press the RSUBMIT function key.

For an example of using compute sevices for MP CONNECT, see "Example 6. Compute Services: Using MP CONNECT for Multi-Processing" on page 45.

ENDRSUBMIT Statement

Indicates the end of a block of statements that should be submitted to the remote host for processing.

Local

Syntax

ENDRSUBMIT <CANCEL>;

Syntax Description

CANCEL
terminates the block of statements without executing the statements. This option is useful in a line-mode session if you see an error in a previously entered statement, and you want to cancel the step.

Details

The ENDRSUBMIT statement signals the end of a block of statements that begins with either:

```
dm 'rsubmit';  /* all releases */
```

or

```
rsubmit;       /* Release 6.06 or later */
```

The remote host processes the statements between either of these statements and the ENDRSUBMIT statement.

You do not use the ENDRSUBMIT statement when using the RSUBMIT command. Use it only when you use the RSUBMIT statement or the DM RSUBMIT statement.

The ENDRSUBMIT statement can be used in any type of SAS session on the local host, but it is particularly useful for running SAS/CONNECT from an interactive line-mode session or a non-interactive job. The RSUBMIT and ENDRSUBMIT statements enable you to include in the same file the statements that are processed by the local host and the statements that are processed by a remote host. The statements for the remote host are enclosed between the RSUBMIT and ENDRSUBMIT statements.

All of the other statements in the program are processed by the local host when you execute the program. The following template is used to build a file that includes statements for both the remote and local hosts in the same program:

```
statements for local host
rsubmit;
   statements for remote host
endrsubmit;
more statements for local host
```

RDISPLAY Command and RDISPLAY Statement

Create two windows. One window displays the contents of the log and the other window lists output generated from the execution of an asynchronous remote submit.

Local

Syntax

RDISPLAY <*remote-session-id*>;

Syntax Description

remote-session-id
specifies the name of the remote session which generated the spooled log and output that is to be displayed.

Details

The RDISPLAY command and the RDISPLAY statement create two windows to display the spooled log and the output that is generated by an asynchronous remote submit. One window displays the log statements, and the other window displays the output statements.

When an asynchronous remote submit executes, the log and the output are not merged into the local log and the output windows; instead, they are spooled until they are retrieved at a later time. RDISPLAY allows you to view the spooled log and output statements created by the asynchronous remote submit. RGET command and the RGET statement must be used to actually merge the spooled and local statements.

The primary difference between the RDISPLAY command and the RDISPLAY statement is that the command can only be used from a windowing environment session or within the DM statement. The RDISPLAY statement can be used in any type of SAS session on the local host.

The following are options for the RDISPLAY command and statement.

remote-session-id
CONNECTREMOTE=*remote-session-id*
REMOTE=*remote-session-id*
is the name of the remote session in which the asynchronous remote submit is executing or has executed. If you have only one active session, *remote-session-id* is not needed. When you have multiple remote sessions that are active and you omit this option, the spooled log and output statements from the current remote session are displayed. The current remote session is the one that is specified in the most recent, successful CONNECTREMOTE= system option, SIGNON command/ statement, RDISPLAY command/statement, RGET command/statement or RSUBMIT command/statement.

Note: This command/statement is available in Version 7 and later. △

RGET Command and RGET Statement

Retrieve the log and output that are created by an asynchronous remote submit and merge them into local log and output windows.

Local

Syntax

RGET <*remote-session-id*>;

Syntax Description

remote-session-id
specifies the name of the remote session which generated the spooled log and output that is to be retrieved.

Details

The RGET command and the RGET statement cause all the spooled log and output from the execution of an asynchronous remote submit to be merged into the local log

and output windows. When an asynchronous remote submit executes, the log and output statements are not merged into the local log and output windows, but instead they are spooled until retrieved at a later time.

If the RGET command or RGET statement is executed while the asynchronous remote submit is still in progress, all currently spooled log and output statements are retrieved and merged into local log and output windows, and the remote submit continues processing as if it were submitted synchronously. That is, you will NOT regain control until the remote submit has completed. If you don't want the remote submit to become synchronous, but you want to check its progress, use the MACVAR option in the SIGNON or the RSUBMIT statement. This allows you to check the progress of an asynchronous remote submit without causing it to execute synchronously.

Note: The system option _LAST_, used to specify the name of the most recently created data set, is not updated for asynchronous remote submits. The system option _LAST_ is not updated even when RGET forces an asynchronous remote submit to a synchronous remote submit. △

The following are options for the RGET command and statement.

remote-session-id
CONNECTREMOTE=*remote-session-id*
REMOTE=*remote-session-id*
is the name of the remote session that generated the spooled log and output that you want to retrieve. If you have only one active session, *remote-session-id* is not needed. When you have multiple remote sessions that are active and you omit this option, the spooled log and output statements from the current remote session are retrieved and merged into the local log and output windows. The current remote session is the one that is specified in the most recent, successful CONNECTREMOTE= system option, SIGNON command/statement, RDISPLAY command/statement, RGET command/statement or RSUBMIT command/statement.

Note: This command/statement is available in Version 7 and later. △

%SYSRPUT Statement

Assigns a value that is on the remote host to a macro variable on the local host.

Remote

Syntax

%SYSRPUT *macro-variable=value*;

Syntax Description

macro-variable
specifies the name of a macro variable on the local host.

value
is a macro variable reference or a character string on the remote host that will be assigned to the *macro-variable*.

Details

The %SYSRPUT statement is a macro statement submitted to the remote host to assign a value that is available on the remote host to a macro variable that can be accessed on the local host. *Value* can be a macro variable reference or a character string. The %SYSRPUT statement is similar to the %LET statement because it is used to assign a value to a macro variable; however, the %SYSRPUT statement assigns a value to a variable on the local host, not on the remote host where the statement is processed. The %SYSRPUT statement places the macro variable into the current referencing environment of the local host.

A *synchronization point* identifies the point during an asynchronous RSUBMIT at which the macro variable that is specified in the %SYSRPUT statement will be defined to the local SAS session so that users can use it in their local processing.

There are three possible synchronization points.

1. The first synchronization point occurs when the RGET command is executed. At this point, all macro variables that were specified by using %SYSRPUT are merged with the local SAS session and are available for processing.
2. The second synchronization point can occur if a synchronous RSUBMIT is started to the same session in which an asynchronous RSUBMIT is already running. When this occurs, all currently spooled log and output statements are retrieved and merged into the local log and output windows, and the remote submit continues from that point as if it were synchronous. That is, you do NOT regain control until the remote submit has completed. In addition, %SYSRPUT macro variables are synchronized with those variables that are generated during the asynchronous remote submit processing up to that point. However, from that point on, it becomes a synchronous remote submit, and macro variables are synchronized immediately when they are executed.

   To override the default for asynchronous remote submits, the CSYSRPUTSYNC option may be specified in the asynchronous RSUBMIT statement, so that local macro variables are set at the time of execution rather than waiting for a synchronization point.
3. The third synchronization point occurs when the SIGNOFF command or the SIGNOFF statement is executed. At this point, all macro variables that were specified by using %SYSRPUT are merged with the local SAS session and are available for processing.

Example 1

This example illustrates how to download a file and return information about the success of the step from a non-interactive job. When remote processing is completed, the job checks the value of the return code stored in RETCODE. Processing continues on the local host if the remote processing is successful.

The %SYSRPUT statement is useful for capturing the value that is returned in the SYSINFO macro variable and passing that value to the local host. The SYSINFO macro variable contains return-code information that is provided by SAS procedures. In the following example, the %SYSRPUT statement follows a PROC DOWNLOAD statement. The value that is returned by %SYSINFO indicates the success of the PROC DOWNLOAD statement:

```
rsubmit;
   %macro download;
      proc download data=remote.mydata
         out=local.mydata;
      run;
      %sysrput retcode=&sysinfo;
   %mend download;
   %download;
endrsubmit;

%macro checkit;
   %if &retcode=0 %then %do;
      further processing on local host
   %end;
%mend checkit;
%checkit;
```

A SAS/CONNECT batch (non-interactive) job always returns a system condition code of 0. To determine the success or failure of the SAS/CONNECT non-interactive job, use the %SYSRPUT macro statement to check the value of the automatic macro variable SYSERR. For more information about the SYSERR macro variable, refer to *SAS Macro Language: Reference*.

Example 2

This example executes an asynchronous remote submit. The CSYSRPUTSYNC= option is specified so that the local macro variable is set when %SYSRPUT executes, rather than waiting until the synchronization point is reached. This way, you are able to get status information about how the asynchronous remote submit is progressing by checking the value of the macro variable STATUS.

```
rsubmit cwait=no csysrputsync=yes;
   %sysrput status=start;
   proc download inlib=sales outlib=tmp
      status=n;
   run;
   %sysrput status=salescomplete;

   proc download inlib=inventry outlib=tmp
      status=n;
   run;
   %sysrput status=inventrycomplete;

   proc upload data=sales.store10 status=n;
   run;
   %sysrput status=storecomplete;
endrsubmit;
```

Example 3

This example shows how to determine what remote system the SAS/CONNECT conversation is attached to.

Remote submit the following statement:

```
%sysrput rhost=&sysscp;
```

To copy the value of RHOST into a local variable for further manipulation, use the following statement:

```
newvar="&rhost";
```

Double quotes (") *must* be used for character values.

%SYSLPUT Statement

Creates a macro variable on the remote host.

Local

Syntax

%SYSLPUT *macro-variable=value*;

Syntax Description

macro-variable
specifies the name of a macro variable on the remote host.

value
is an alphanumeric string that will be assigned to the remote *macro-variable*, which should not contain nested quotation marks.

Details

%SYSLPUT submits a macro assignment statement to the remote host to assign a value that is available on the local host to a macro variable that is accessed on the remote host. If you are signed on to multiple remote hosts, &SYSLPUT goes to the most recently used remote session. If you are signed on to only one host, &SYSLPUT goes to that host. If you are not signed on to any host, an error condition results.

Note: %SYSLPUT performs the opposite function of %SYSRPUT. It creates a macro variable in the remote environment based on a value in the local environment. △

macro-variable can be a macro variable reference or a character string. The %SYSLPUT statement is similar to the %LET statement because it assigns a value to a macro variable; however, the %SYSLPUT statement assigns a value to a variable on the remote host, not on the local host where the statement is processed. The %SYSLPUT statement places the macro variable into the current referencing environment of the remote host.

Example 1

This example illustrates how to set the macro variable FLAG to 1 on the only remote session.

```
%syslput flag=1;
```

Example 2

This example sets the macro variable REMDIR1 on the local host to the path of a directory located on the remote host. The macro statement %SYSLPUT is then used to create the macro variable REMDIR2 on the remote host with the same value as REMDIR1. PROC UPLOAD is used to transfer the data set ENG101 from the WORK library of the local host to the REMDIR2 directory on the remote host.

```
%let remdir1=/dept/engineering/staff/dr_smith;
%syslput remdir2=remdir1;
rsubmit;
   proc upload infile= eng101
      outfile="&remdir2/eng101";
   run;
endrsubmit;
```

WAITFOR Statement

Makes the current SAS session wait for the completion of one or more asynchronously executing tasks that are already in progress.

Local

Syntax

WAITFOR _ANY_|_ALL_ *task1 ... taskn* <TIMEOUT=*seconds*>;

Syntax Description

ANY
suspends the SAS session for the completion of any of the specified tasks (a logical OR of the completion task states).

ALL
suspends the SAS session for the completion of all of the specified tasks (a logical AND of the completion task states).

task
identifies one or more tasks to be completed asynchronously in an optionally allotted time period. A *remote-session-id* that is associated with a REMOTE= or PROCESS= option to the RSUBMIT statement corresponds to the name of the task that is specified in the WAITFOR statement. It can also be the name of an asynchronous X command or some other asynchronously executing SAS task.

TIMEOUT=*seconds*
allots the interval in seconds for asynchronous task processing. If the specified tasks have not finished processing by timeout, task processing is terminated, giving the SYSRC system macro variable a non-zero status. If the specified tasks finish processing before timeout, the WAITFOR statement returns control to the SAS session.

Details

The WAITFOR statement is used to make the current SAS session wait for the completion of one or more tasks that are already in progress as specified by the options _ANY_ or _ALL_. You can use WAITFOR only for asynchronously executing tasks (for example, RSUBMITS that are executed with the WAIT option set to NO). If you try to use WAITFOR and there are no asynchronous tasks executing, then the WAITFOR statement will not enforce the wait condition, but, instead, will continue task execution in the current SAS session.

The name of the task corresponds with the *remote-session-id* that is assigned to the REMOTE= or PROCESS= option in the RSUBMIT statement. Omission of the REMOTE= or PROCESS= option implies the current session.

The WAITFOR statement can wait for the completion of one or more tasks. If more than one task is specified, then the WAITFOR statement must include either the _ANY_ or the _ALL_ options. The _ANY_ option suspends the SAS session for the completion of any of the specified tasks (a logical OR of the completion task states). The _ALL_ option suspends the SAS session for the completion of all of the specified tasks (a logical AND of the completion task states). The WAITFOR statement does not support complex logical statements, such as A OR (B AND C).

Invalid tasks that are specified in the WAITFOR statement are ignored but are identified in notes in the SAS log.

Example 1

This example shows the suspension of the current SAS session until both tasks have completed or 300 seconds (5 minutes) pass, whichever occurs first.

```
waitfor _all_  remhost printjb  timeout=300;
```

This statement causes the current SAS session to suspend execution in the current session until the REMHOST and the PRINTJB tasks finish. REMHOST and PRINTJB are remote session ids that are assigned to either the REMOTE= or the PROCESS= option in the RSUBMIT statement. Both tasks must complete within the allotted time or the time must expire before the WAITFOR statement returns control to the local SAS session. Should time expire before the completion of both tasks, control is returned to the current SAS session and the asynchronous tasks continue to execute. The SYSRC macro in the autocall library can be queried for task status. Alternatively, if you specified macro variables for the REMHOST and PRINTJB tasks with the MACVAR option, you could query those macro variables for status information.

Example 2

This statement causes the suspension of the current SAS session until either task REMHOST or FORMATJB has completed.

```
waitfor _any_ remhost formatjb;
```

Because no time limit has been placed on the processing of these tasks, as much time as needed can be used for task completion. Upon completion of either task, the WAITFOR statement returns control to the current SAS session.

LISTTASK Statement

For asynchronous tasks, lists the active tasks and the completed tasks that were processed in the current SAS session.

Local

Syntax

LISTTASK _ALL_ | *task* ;

Syntax Description

ALL
gives information about asynchronous tasks, which are currently executing or have completed.

task
identifies a specific task by a name that corresponds to the *remote-session-id* that is associated with the REMOTE= or the PROCESS= option in the RSUBMIT statement. It is the name of the task that is specified in the WAITFOR statement.

Details

The LISTTASK statement lists information about a single active task by name or about all tasks in the current SAS session.

Example 1

This example lists information for all tasks.

```
listtask _all_;
```

Example 2

This example lists information for the REMHOST task only.

```
listtask remhost;
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/CONNECT User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 537.

SAS/CONNECT User's Guide, Version 8

ISBN 1–58025–477–2

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

1st printing, September 1999