PN 00449

Connect:Direct®

Process Statements Guide

sterling commerce

PN 00449

Connect:Direct®

Process Statements Guide

Connect:Direct Process Statements Guide
First Edition
February 2004

CDPRSTG402

Contents

Preface

The *Connect:Direct Process Statements Guide* provides the information you need to write Connect:Direct Processes. This includes statement syntax and descriptions of statement parameters.

Connect:Direct Process Documentation

The Connect:Direct Process documentation consists of the following publications:

- *Connect:Direct Process Concepts and Examples Guide* provides an overview of Connect:Direct, describes the general structure and syntax rules for the Process language, and includes numerous examples.
- *Connect:Direct Process Statements Guide* describes the Process statements for the following platforms:
  - Connect:Direct OS/390
  - Connect:Direct VM/ESA
  - Connect:Direct VSE
  - Connect:Direct Tandem NonStop Kernel
  - Connect:Direct OS/400
  - Connect:Direct OpenVME
  - Connect:Direct OpenVMS
  - Connect:Direct UNIX
  - Connect:Direct Windows

The Connect:Direct Process documentation assumes knowledge of the operating systems for Connect:Direct products. If you are not familiar with a specific operating system, including its applications, network, and environment, refer to the appropriate library of manuals. For additional information about Connect:Direct, refer to the library of Connect:Direct documentation.

Notational Conventions

The Connect:Direct Process documentation set uses certain notational conventions. This following table describes the conventions used the documentation set.

Notation	Description
Uppercase Letters	Uppercase letters in the statement format indicate that you type in information as shown. For Connect:Direct UNIX, and OpenVME: Values must be in the proper case for the node on which they will be processed. For Connect:Direct Windows: This convention does not apply.
Uppercase and Lowercase Letters	A statement in uppercase letters followed by lowercase letters indicates an alternative to typing the entire statement. For example, PROCess means that you need only type PR0C for the statement to be valid. For Connect:Direct UNIX, OpenVME and Windows: This convention does not apply.
Lowercase Letters	Lowercase letters or words in statements or syntax boxes require substitution by the user. For example, PNODE=primary-node-name indicates that you must provide the name of the primary node. For Connect:Direct UNIX, OpenVME, and Windows: This convention does not apply. Refer to Italic Letters for the notational convention used for variable substitution for these products.
Bold Letters	Bold print in syntax boxes indicates required labels, statements, and parameters. For example, **DSN=filename** indicates that the parameter DSN is required. For Connect:Direct UNIX, OpenVME, and Windows: In text, parameters are shown in boldface characters to differentiate them from surrounding text.
Italic Letters	For Connect:Direct UNIX, OpenVME and Windows, italic letters act as placeholders for information you must provide. For example, *dsn=filename* indicates that you would type the actual name for a file instead of the word shown in italic type.
Underlined Letters	Underlining indicates default values for parameters and subparameters. For example, RETAIN=Yes|No|Initial specifies that the default for RETAIN is No.
Vertical Bars	Vertical bars indicate that you can supply one of a series of values separated by the vertical bars. For example HOLD=Yes|No|Call specifies that Yes or No or Call is valid.
Brackets	Brackets indicate optional information within an optional parameter. For example, STARTT=([date|day][,hh:mm:ssXM]) indicates that you can specify either a date or a day, a date or a day plus a time, or just a time. For Connect:Direct OpenVMS: OpenVMS uses brackets in its directory naming convention. Because brackets are not available on most keyboards on IBM systems, use less than and greater than signs (< >) in a Connect:Direct Process that is submitted from the IBM node.

Notation	Description
Horizontal Ellipsis	Horizontal ellipsis indicates that the preceding item may be repeated. For example, &symbolic_name_2=variable-string-2. . . indicates that you can specify multiple symbolic parameters.
Vertical Ellipsis	Vertical ellipsis indicates that not all of the data is displayed. Typically, the vertical series of ellipses is used in examples.
Additional Notations	Code all commas and parentheses as they appear. Process with a capital P refers to a Connect:Direct Process. Monospaced characters (characters of equal width) represent information for screens, commands, Processes, and reports.

Getting Support for Sterling Commerce Products

Sterling Commerce provides intuitive technical products and superior Help and documentation to enable you to work independently. However, if you have a technical question about a Sterling Commerce product, use the Sterling Commerce Customer Support Web site.

The Sterling Commerce Customer Support Web site at www.sterlingcommerce.com is the doorway to Web support, information, and tools. This Web site contains several informative links, including a solutions database, an issue tracking system, fix information, documentation, workshop information, contact information, sunset and retirement schedules, and ordering information. Refer to the Customer Support Reference Guide at www.sterlingcommerce.com/customer/tech_support.html for specific information on getting support for Sterling Commerce products.

Chapter 1

Connect:Direct OS/390 Statement Models

Connect:Direct OS/390 includes a sample Process library, containing models of the Process statements. The members of this Process library may be used as templates for building Connect:Direct Processes. Statement models can be combined to create a Process.

Two forms of the Process statement models are represented in the Process library. Commented files are assigned file names preceded by the at sign (@); those files with the pound sign (#) prefix do not include comments.

Creating a Process with Statement Models

These instructions outline the procedure for copying the various Process statement models into a new file if you are using the Connect:Direct OS/390 Interactive User Interface (IUI) or the Interactive System Productivity Facility (ISPF) EDIT facility. Standard ISPF editing rules apply.

1. Access the Process definition screen (DF on the Connect:Direct OS/390 Primary Option Menu), and specify a new PDS member. Press **ENTER**. Note that the PROCESS and the PUBLIC LIBRARY NAMES display the partitioned data sets (PDSs) allocated by the signon CLIST or the TSO logon procedure.
2. Type **COPY** on the command line at the top of the blank member. Press **ENTER**. An ISPF Edit-Copy screen displays.
3. In the DATA SET NAME field, type the name of the Process library and the member name of the model you want copied into your new member. Press **ENTER**.

   For example, you build your Process by first including the PROCESS statement. If you specify @PROCESS with the Process library, the Connect:Direct PROCESS statement model is copied into the member. (Use #PROCESS if you do not want to include the comments in your Process.)

   The PROCESS statement model is copied into your new member.
4. To add additional statement models as needed, type **a**, over the 0 (zero) in the number column of the last line of the member and repeat steps 2 and 3. The specified

Connect:Direct statement model is copied into the member following the PROCESS statement.

5. Continue adding statement models until your Process is complete.

Editing Statement Models

Each statement model must be edited according to the following guidelines:

- Replace underscores with the appropriate parameter values. Provide an appropriate Process name in the PROCESS statement.
- Delete all lines that are not applicable.
- Continuation marks are necessary on all but the last line of each statement model.
- Comment lines are optional and may be deleted.

The Process can be saved to the PDS member and submitted from the Submit panels in the IUI. See the *Connect:Direct OS/390 User's Guide* for instructions on executing Processes.

Chapter 2

Connect:Direct OS/390 PROCESS Statement

A Connect:Direct OS/390 PROCESS statement defines the attributes of a Process and is always the first statement in a Process.

Note: The maximum storage area allowed for a Process statement is 64K. To accommodate a larger Process, split the Process into two separate Processes. Include a SUBMIT statement in the first Process to run the second Process.

Format

The Connect:Direct OS/390 PROCESS statement format follows. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the PROCESS statement format.

Label	Statement	Parameters
process name	**PROCess**	**SNODE=secondary-node-name | TCPNAME=tcpvalue**
		CRC=OFF | ON
		CLASS=n
		DEBUG=trace bits
		HOLD=Yes | No | Call
		%JDATE
		MAXDELAY [=Unlimited | Queued | hh:mm:ss | 0]
		NOTIFY=%USER | userid
		%NUM1=process-submit-time
		PACCT='pnode-accounting-data'

Label	Statement	Parameters
		PLEXCLASS=(pnode class, snode class)
		%PNODE=pnode
		PNODE=primary-node-name
		PNODEID=(id | [,pswd] [,newpswd])
		PRTY=nn
		REQUEUE=Yes | No
		RETAIN=Yes | No | Initial
		SACCT='snode-accounting-data'
		SNODEID=(id | [,pswd] [,newpswd])
		STARTT=([date |day][,hh:mm:ssXM])
		%SUBDATE
		%SUBDATE1
		%SUBDATE2
		%SUBDATE3
		%SUBDATE4
		%SUBTIME
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n
		%USER=userid

Field Descriptions

process name

specifies the name of the Process. The Process name must exactly match the member name under which it is stored in the Process Library PDS. Accordingly, every process name must be 1-8 characters in length, begin with an alphabetic character, and contain only the characters A-Z, 0-9, @, #, -, and $. The Process name must start in column one. This label is used to find the Process in the Process Library and to identify the Process in any messages or statistics relating to the Process.

PROCess
identifies the statement with all its parameters as the PROCESS statement. This statement identifier can be abbreviated to PROC.

Required Parameters

SNODE = secondary-node-name | TCPNAME = tcpvalue
specifies the secondary node to be used in the Process.

Note: This parameter is not required if SNODE is specified on the SUBMIT command.

secondary-node-name is a 1-16 character alphanumeric name that is defined in the network map. The name may contain only alphanumerics or nationals (@ # $), with embedded periods.

Use the **TCPNAME=tcpvalue** form of the SNODE parameter to specify TCP/IP connections not defined in the Connect:Direct network map, where **tcpvalue** can be one of the following:

- A DNS name up to 255 characters
- A dotted decimal TCP/IP address up to 15 bytes
- An alias name up to 16 characters long.

You can append a semicolon followed by a port number up to 5 characters long using a semicolon. If you do not specify a port number, Connect:Direct gets the port number from the TCP.IP.DEFAULT entry of the network map.

If the DNS must continue to the next statement within the Process, concatenate it using the || character. For example:

```
PROC1    PROCESS                                                    -
         SNODE=TCPNAME=THIS.IS.A.LONG.DOMAIN.NAME.THAT.SPANS. ||    -
         MORE.THAN.ONE.LINE ||                                      -
         ;04199
```

If you use TCPNAME=**tcpvalue** in the SNODE parameter, Processes in HO WC status do not automatically restart when the node becomes available and another Process is submitted to that node. This works differently than SNODE=secondary-node-name.

For Interlink TCP/IP support: If **tcpvalue** specifies the name of the network, Connect:Direct uses this name to get the real address from the network name server on the TCP/IP network.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. If CLASS is not specified in the Connect:Direct Process, it will default to the class value specified in

the ADJACENT.NODE NETMAP record for the destination node (SNODE). Values range from 1-255.

CRC=(OFF|ON)

Provides an override of the initial CRC setting in the initialization parameters. This value is ignored if the initialization parameter does not allow for overrides. This parameter is only valid for TCP/IP transfers. The default is OFF.

DEBUG = trace bits

specifies the 8-position trace setting for the Process. This allows you to specify a trace for a specific Process. The **DEBUG Setting** column in the following table lists acceptable trace values.

DEBUG Setting	Trace Type	Output DD
80000000	COPY Routine and RUN TASK trace	RADBDD01
10000000	Full TPCB/SYMBOLICS from DMCBSUBM	DMCBSUBM
08000000	Session manager trace	RADBDD05
04000000	Separate trace per task (Example: “R0000005” to trace TASK 5)	Rnnnnnnn
02000000	API session trace	RADBDD07
01000000	DMGCBSUB trace	RADBDD08
00800000	NETEX task termination disconnect trace	NTXTRACE
00400000	TCQSH from DMCOPYRT	DMCOPYRT
00200000	Make each SVC dump unique	N/A
00040000	GETMAIN/FREEMAIN trace	RADBDD16
00008000	I/O buffer trace	RADBDD21
00004000	WTO all dynamic allocation parameters	RADBDD22
00002000	Connect:Direct/Plex traces	
	ACTION queue manager trace	CDPLXACT
	CKPT queue manager trace	CDPLXCKP
	TCQ queue manager trace	CDPLXTCQ
	STATS queue manager trace	CDPLXSTA
	First REQUEST queue manager trace	CDPLXREQ
	Second and subsequent REQUEST queue manager trace. For example, “CDPLXR03” traces the third REQUEST queue manager. The number of REQUEST queue manager traces is based on the maximum number of servers from the asset protection (APKEY) file.	CDPLXRnn

DEBUG Setting	Trace Type	Output DD
	JOIN queue manager trace	CDPLXJOI
00001000	Workload Balancing trace	CDPLXWLB
00000080	RPL trace - long	RPLOUT
00000040	RPL trace - short	RPLOUT
00000020	Version 2 session trace	RADBDD33
00000008	Logon exit trace	RADBDD35
00000004	Logon processor trace	RADBDD36
00000002	SCIP exit trace	RADBDD37
00000001	Extended dump Information	ESTAE

HOLD = Yes | No | Call

specifies whether the Process is to be placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process. When both HOLD=YES and MAXDELAY=YES, this value is not valid; the PROCESS statement returns message SCBI219I.

No specifies that the Process is to execute as soon as possible. HOLD=NO is the default.

Call specifies that Connect:Direct is to place the Process in the hold queue until a session is established with the specified SNODE. This session is established by either another Process starting on the PNODE destined for the same SNODE or the SNODE contacting the PNODE. For example, a Process submitted with HOLD=NO establishes a session and causes execution of any Processes residing on the SNODE destined for this node that are submitted with HOLD=CALL.

Note the following:

- Connect:Direct ignores the HOLD parameter if RETAIN=Y.
- When the SNODE is a Windows operating system, a null or ENABLE Process is required from the Windows operating system to release the held Processes. A normal send or receive of a file does not release them. This functionality enables those who dial in to send or receive files without executing held Processes until they are ready.

%JDATE

specifies the date the Process was submitted in Julian format. The variable is resolved as the submission date of the process in the format yyyyddd. Among other uses, the value returned is suitable for constructing a file name on the node receiving the file.

When used for this purpose, the %JDATE value must be preceded by an alphabetic character.

Note: The value of the variable is resolved at Process submit time. The value will correspond to the date on which the Process was submitted, regardless of when or how many times the Process is actually executed.

MAXDELAY [=UNLIMITED | QUEUED | hh:mm:ss | 0]
specifies that the session is returned to the API after a specific amount of time. UNLIMITED is the default value.

Unlimited specifies that the established session is held until the Process is complete. This is the default when no parameters are specified.

Queued specifies that the established session is held for 30 minutes, or until the Process is complete, whichever occurs first.

hh:mm:ss specifies that the established session is held until the time specified, or until the Process is finished, whichever occurs first.

0 specifies that the established session returns to the API immediately following execution of the Process. If the session cannot be started and all session retry attempts fail, the Process fails.

NOTIFY = %USER | userid
specifies the user to receive Process completion messages.

%USER specifies that the user on the host Connect:Direct who submitted the Process receives the completion messages. If the Connect:Direct userid is different from the host userid, the user is not notified.

userid specifies the TSO userid that will receive Process completion messages.

Note: The NOTIFY capability is not supported across OS/390 images in the sysplex environment. Process completion messages cannot be sent across OS/390 images in a sysplex.

%NUM1 = process-submit-time
specifies unique temporary data set names in the DSN parameter of the COPY TO statement. The variable is resolved as the submission time of the Process in a 6-digit numeric-value format of minutes, seconds, and fraction of seconds. The %NUM1 value must be preceded by an alphabetic character.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

PLEXCLASS = (pnode class, snode class)
specifies the class that directs the Process to only certain servers in a Connect:Direct/Plex. This parameter is only used in a Connect:Direct/Plex.

Each server in a Connect:Direct/Plex can be designated to support only certain PLEXCLASSes through the CDPLEX.PLEXCLASSES initialization parameter. Processes can then be limited to only those servers by specifying the PLEXCLASS in the Process definition.

The pnode class controls which Connect:Direct/Server runs the Process. The snode class controls what other node is used by the Process.

The pnode class and snode class are each 1-8 characters long. An asterisk (*) indicates that the Process will run on any server with an asterisk designated in the CDPLEX.PLEXCLASSES initialization parameter. If no PLEXCLASS is specified, the network map is checked for a default PLEXCLASS. If the network map does not specify a default PLEXCLASS, then an asterisk is used as the default.

If a Process must run on a specific Connect:Direct/Server, specify the Connect:Direct/Server name in the field. The Process will only run on that server.

Note: NETEX processes do not pass the SNODE PLEXCLASS to the C:D/Plex Manager for workload balancing. NETEX processes directed to the C:D/Plex Manager are scheduled to an available C:D/Plex Server based on the SNODE PLEXCLASS specified (if specified) in the NETMAP ADJACENT.NODE entry for the NETEX node and the C:D/Plex Servers that support NETEX.

%PNODE = pnode
is an intrinsic symbolic variable that resolves to the PNODE of the Process.

PNODE = primary-node-name
specifies the primary node to be used in the Process.

primary-node-name is a 1-16 character alphanumeric name that is defined in the network map. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

Note: The node to which the Process is submitted is always the PNODE. This parameter defaults to the name of the node submitting the Process and need not be specified. It is used for documentation purposes only.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should be used only to validate security with an ID different from the one you used to sign on to Connect:Direct.

id specifies the security ID passed to the security system at the PNODE (1-64 alphanumeric characters).

pswd specifies the current security password (1-64 alphanumeric characters) for the specified ID. This parameter can be used by the security system at the PNODE to validate the current security password. The password is optional unless the user has security set to require a password.

newpswd specifies the new security password (1-64 alphanumeric characters). It can be used by the security system to change the current security password to the new security password.

PRTY = nn

specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ holds all Processes that have been submitted to Connect:Direct. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect VTAM transmission priority. The range is from 0-15. If PRTY is not specified, the default is the priority defined by the PRTYDEF keyword in the Connect:Direct initialization parameters. See the *Connect:Direct OS/390 Administration Guide* for more information.

REQUEUE = Yes | No

specifies whether a COPY step should requeue if an x37 abend occurs during processing. This parameter is valid when checkpointing and in conjunction with ALLOC.RETRIES. See the *Connect:Direct OS/390 Administration Guide* for details.

Yes allows the requeued Process to be placed in the Hold queue with a status of HELD IN ERROR (HE). Corrective action can be taken and the Process restarted with the failing step. Checkpointing resumes at the last successful checkpoint. Note that the Process must be explicitly released from the Hold queue when the status is HELD IN ERROR (HE).

No causes the Process to run to completion, executing subsequent steps when a COPY step fails with an abend (such as x37). The default is NO.

When MAXDELAY is specified, REQUEUE=NO is forced even when REQUEUE=YES is specified. The Process executes as if REQUEUE=NO were specified.

RETAIN = Yes | No | Initial

keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains in the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval. However, a date is invalid as a STARTT subparameter when used in conjunction with RETAIN.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time Connect:Direct is initialized. The Process will not execute when initially submitted. Note that STARTT should not be coded with RETAIN=INITIAL.

When MAXDELAY is specified, RETAIN=YES and RETAIN=INITIAL are ignored. The Process is executed as if RETAIN=NO were coded.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit. This parameter is ignored when the SNODE is a Connect:Direct OS/400 node.

SNODEID = (id [,pswd] [,newpswd])

specifies security user IDs and passwords at the secondary node (SNODE). For OS/390 and UNIX platforms, each value can be 1-64 alphanumeric characters. For other operating environments unless noted, each value can be 1-8 alphanumeric characters.

For Connect:Direct OS/400: If an SNODEID and password of the Process submitter is not coded in the PROCESS statement, the user ID and password of the Process submitter will be used for the security ID and password check by Connect:Direct OS/400.

id specifies the security ID passed to the security system on the SNODE.

For Connect:Direct Tandem: This subparameter specifies the Tandem group number and user number. These numbers can range from 0-255. When specifying a Tandem value, you must use a period character as a separator between the group number and the user number.

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles may be 10 characters long.

PassTicket Support: If the SNODE DTF is initialized indicating the ability to authenticate using a PassTicket password, then specifying an SNODE userid override without password override results in the creation of a PassTicket password. The creation of a PassTicket by the PNODE depends on proper information in the PNODE Authorization File and the appropriate creation of the PNODE stage 2 Security Exit. See the *Connect:Direct OS/390 Administration Guide* for more information about the Authorization File and the stage 2 Security Exit.

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password. The password is optional unless the user has security set to require a password.

For Connect:Direct Tandem: Passwords for an IBM OS/390 node must be specified in uppercase alphanumeric characters. Therefore, a Process cannot be successfully initiated from Connect:Direct OS/390 with Connect:Direct Tandem unless the Connect:Direct Tandem SNODEID password follows the same convention (no lowercase characters and no control characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password.

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

For Connect:Direct OS/400: This subparameter is ignored.

STARTT = ([date | day] [,hh:mm:ssXM])

specifies that the Process will execute at a selected date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

date specifies that the Process is to be held until a specific date. You can specify the day (dd), month (mm), and year (yy for 2-digit year and yyyy for 4-digit year).

You can use periods or backslashes (/) to separate the components of a date value.

Note: You can omit the separators only for transfers between mainframe nodes.

To specify the order of a Gregorian day, month, and year, you *must* define the DATEFORM initialization parameter. If you do not specify the DATEFORM parameter, Connect:Direct OS/390 defaults to the DATEFORM=MDY formats described in this section.

After you designate the date order in your initialization parameters, you can use the following date formats according to the DATEFORM setting shown:

DATEFORM=MDY

- mm/dd/yy *or* mm/dd/yyyy
- mm.dd.yy *or* mm.dd.yyyy

DATEFORM=DMY

- dd/mm/yy *or* dd/mm/yyyy
- dd.mm.yy *or* dd.mm.yyyy

DATEFORM=YMD

- yy/mm/dd *or* yyyy/mm/dd
- yy.mm.dd *or* yyyy.mm.dd

DATEFORM=YDM

- yy/dd/mm *or* yyyy/dd/mm
- yy.dd.mm *or* yyyy.dd.mm

Connect:Direct processes Julian dates the same as previous releases. The following formats are valid:

- yyddd *or* yyyyddd
- yy/ddd *or* yyyy/ddd
- yy.ddd *or* yyyy.ddd

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

day specifies the day of the week the Process will be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value can be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. Therefore, if a Process is submitted on Monday, with Monday as the only STARTT parameter, the Process does not run until the following Monday.

You can also specify TODAY, which releases the Process for execution the day and time of Process submission (unless the time of day is specified), or TOMORROW, which releases the Process for execution the next day. If a time of day is not specified with TOMORROW, the Process executes after midnight.

hh:mm:ssXM indicates the time of day the Process will be released in hours (hh), minutes (mm), and seconds (ss). XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT to release the Process for execution at midnight.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed in the Hold queue even if a start time is specified.

%SUBDATE

specifies the date the Process was submitted in Gregorian format. The variable is resolved as the submission date of the Process in the format cyymmdd where **c** is the century indicator and is set to **0** for year 19yy or **1** for year 20yy. Among other uses, the value returned is suitable for constructing a file name on the node receiving the file. When used for this purpose, the %SUBDATE value must be preceded by an alphabetic character.

Note: The value of the variable is resolved at Process submit time. The value will correspond to the date on which the Process was submitted, regardless of when or how many times the Process is actually executed.

%SUBDATE1

Use this parameter to substitute the submitted date in the yyyymmdd date format.

%SUBDATE2

Use this parameter to substitute the submitted date in the yyyyddmm date format.

%SUBDATE3
Use this parameter to substitute the submitted date in the mmddyyyy date format.

%SUBDATE4
Use this parameter to substitute the submitted date in the ddmmyyyy date format.

%SUBTIME
specifies the time the process was submitted. The variable is resolved as the submission time of the process in the format hhmmss. Among other uses, the value returned is suitable for constructing a file name on the node receiving the file. The %SUBTIME value must be preceded by an alphabetic character.

Note: The value of the variable is resolved at Process submit time. The value will correspond to the time at which the Process was submitted, regardless of when or how many times the Process is actually executed.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n
specifies the default value for a symbolic parameter in the Process. This default can be overridden in the SUBMIT command.

A symbolic parameter containing special characters must be enclosed in single quotation marks (‘ ’).

%USER = userid
is an intrinsic symbolic variable that resolves to the user submitting the Process.

Example

The following is an example PROCESS statement. Its description follows:

```
PROC1          PROCESSSNODE=CD.NODE.A                                  -
                     SNODEID=(JONES,OPENUP)                            -
                     CLASS=4                                           -
                     HOLD=YES                                          -
                     NOTIFY=%USER                                      -
                     PACCT=‘OPERATIONS, DEPT. 87’                      -
                     RETAIN=NO                                         -
                     &DATESUB=%SUBDATE
```

The Process named PROC1 specifies a secondary node (SNODE) of CD.NODE.A. The corresponding security userids and passwords (SNODEID) have been included.

The Process will run in CLASS 4.

The Process will be placed in the Hold queue until it is released for execution with a CHANGE PROCESS command. As indicated by the NOTIFY parameter, the TSO user who submitted the Process is notified upon completion of the Process.

The PACCT parameter specifies that all accounting information will be attributed to the operations account, department 87, if the node has a program that maintains this information.

Once the Process executes, it is deleted because the RETAIN parameter is set to NO. Note that NO is the default value for the RETAIN parameter.

The symbolic name &DATESUB will be set to the date the process was submitted in *cyymmdd* format.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 3

Connect:Direct OS/390 COPY Statement

The Connect:Direct OS/390 COPY statement allows you to copy files from one node to another.

Supported File Types

The Connect:Direct OS/390 COPY statement copies the following types of files:

- Sequential Access Method (SAM)
- Virtual Storage Access Method (VSAM)
- Basic Direct Access Method (BDAM)
- Indexed Sequential Access Method (ISAM)
- Generation Data Group (GDG)
- Partitioned Data Sets (PDSs)
- Library (PDSEs)
- Linear
- IBM Data Facility Data Set Services (DFDSS) volumes

Both disk and tape transfers are supported.

General Information on PDS Support

Connect:Direct OS/390 supports transmission of all PDSs, including load modules and overlay files. The user can specify whether:

- An entire PDS is to be sent.
- Certain members are selected for transmission (SELECT parameter).
- Certain members are excluded from transmission (EXCLUDE parameter).
- One member is sent to a sequential file.
- A sequential file is sent to a PDS member.
- File aliases are sent along with the requested file (ALIAS parameter).

- A member is renamed (NEWNAME parameter).
- Members replace existing members of the same name at the receiving node (REPLACE and NOREPLACE parameters, **R** and **NR** subparameters of the SELECT parameter).

PDS Guidelines

Refer to the following guidelines when copying a PDS:

- EXCLUDE or SELECT cannot be used if FROM DSN contains a member name or if the TO clause specifies SYSOPTS="UNIQUE=YES".
- The hierarchy for the SELECT and EXCLUDE parameters proceeds from top to bottom (highest override priority to lowest) as follows:
  - Exclude by member name
  - Select by member name
  - Exclude by generic (or range)
  - Select by generic (or range)
  - Combine various specifications in a list after the EXCLUDE parameter

  If EXCLUDE is specified and SELECT is not specified, all members not excluded are copied. If EXCLUDE is not specified and SELECT is specified, only selected members are copied.
- If a non-PDS is specified in the FROM DSN, the TO DSN must specify a single member.
- When the COPY statement involves two PDSs, all members are sent unless one of the following conditions exists:
  - The SELECT option is specified.
  - The EXCLUDE option is specified.
  - A member name is specified in the COPY FROM statement.
  - A member name is specified in the COPY TO statement.
  - UNIQUE=YES is specified in the COPY TO SYSOPTS statement. With UNIQUE=YES, only single-member transfers are supported.
- When the TO DSN contains a member name, EXCLUDE (in the COPY FROM statement) cannot be used. In addition, the SELECT entry (also in the COPY FROM statement) is only valid if it contains one member.
- The FROM DSN must specify a single member when a non-PDS is specified in the TO DSN.
- If specifying a non-PDS in the TO DSN, only the data portion of a PDS member is stored in a SAM file. Directory information is ignored.
- When the TO DSN is a tape file, the FROM DSN must specify a single member.

General Information on VSAM Support

Copies to existing VSAM files are processed in EXTEND mode. The VSAM file is extended rather than copied over and replaced. To completely copy over an existing VSAM file, that is to copy to a VSAM file in LOAD mode, you must set the DISP parameter to DISP=RPL. The destination file must be defined with the REUSE attribute; otherwise OPEN will fail.

Copies to new VSAM files propagate those attributes supported by SMS. Specifically, attributes that are supported by SVC 99 for dynamic allocation are propagated. See *General Information on SMS Support* on page 19 for the specific attributes that are propagated. If the file requires other VSAM attributes, you must pre-allocate the file outside of Connect:Direct using IDCAMS, or in a Process step using the DMRTAMS utility prior to the COPY step. See the *Connect:Direct OS/390 User's Guide* for information about DMRTAMS.

General Information on SMS Support

The Connect:Direct OS/390 COPY statement supports the transmission and creation of data sets with SMS attributes. The following restrictions apply to the transmission of these data sets:

- If the TO data set is specified with SMS attributes, no DCB or SPACE parameters are propagated by default from the FROM data set, because dynamic allocation treats them as overrides and they replace attributes from the SMS DATA CLASS. Use the DATACLAS parameter and an appropriate SMS definition at the receiving site to acquire default values for DCB and SPACE. See *SMS Propagation* on page 21 for more information.
- VSAM data sets can be created as new data sets as part of the COPY TO statement. All forms of VSAM data sets (KSDS, ESDS, RRDS, and LINEAR) are supported as new data sets.

  If allocating a NEW VSAM file, the following VSAM attributes are propagated to the receiving node if they are not specifically coded in the process or in the TYPE file definition, and no DATACLAS or LIKE= keywords are coded in the process or TYPE file definition (and no dataclass is propagated):

  - KEYLEN - For KSDS datasets only. The length of the key.
  - KEYOFF - For KSDS datasets only. The offset where the key starts.
  - LRECL - The MAX RECSIZE. If this is propagated, the output file will be allocated with a MAX and AVG record size of the input file's MAX RECSIZE.

- RECORG - The record organization (KS, RR, ES, or LS).

This way, you do not need to code any file attributes for the TO file when dynamically allocating a NEW VSAM file.

Note: The parameters not supported by SMS cannot be propagated by Connect:Direct.

- For platforms other than OS/390, all SMS parameters must be specified as subparameters in SYSOPTS. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.
- For a data set to be SMS-controlled, it must be created with either a STORAGE CLASS or a MANAGEMENT CLASS or both.
- SMS attributes are not propagated from the FROM data set by default because dynamic allocation treats them as overrides. See *SMS Propagation* on page 21 for more information
- The disposition of all SMS controlled data sets is always CATLG. For example, if you code DISP=(NEW, KEEP) on the COPY TO statement, the system uses DISP=(NEW, CATLG) to control the data set.

SMS-Specific Parameters

The following parameters are specific to SMS support:

- AVGREC
- DATACLAS
- DSNTYPE
- KEYLEN
- KEYOFF
- LIKE
- LRECL
- MGMTCLAS
- RECORG
- SECMODEL
- STORCLAS
- SYSOPTS

The AVGREC parameter is valid only when SPACE is coded on the COPY TO statement.

SMS Propagation

You can propagate any of the SMS classes from the sending side to the receiving side by coding a value of $$$$$$$$ for the class name that you want to propagate. Following is an example:

```
SMSTEST1   PROC    SNODE=YOUR.OTHER.NODE
STEP01     COPY    FROM(                                  PNODE  -
                        DSN=HLQ.SMSTEST.DS                       -
                        DISP=SHR )                               -
                   TO(DSN=HLQ.SMSTEST.TODS                 SNODE  -
                        UNIT=SYSDA                               -
                        STORCLAS=$$$$$$$$ -
                        MGMTCLAS=$$$$$$$$ -
                        DATACLAS=$$$$$$$$ -
                        DISP=(RPL))
```

For any class coded that contains the class name value of $$$$$$$$, Connect:Direct does an INFO call to dynamic allocation to obtain the SMS classes associated with the sending data set. If Connect:Direct returns an SMS class, this class substitutes for the $$$$$$$ value. If Connect:Direct does not return an SMS class, the keyword and value are blanked and not passed to the receiving node. The class name propagated must be defined on the receiving node (if the receiving node is an OS/390 OS) or an error occurs.

Format

The COPY statement includes a FROM parameter that includes the source file name and a TO parameter that includes the destination file name. Additional parameters and subparameters can be specified to further customize the file transfer operation.

Note: The length of the entire COPY statement cannot exceed 2040 bytes.

The following pages provide the format of the COPY statement. All platform-specific parameters and subparameters are listed along with their possible values. A description of each parameter and subparameter follows the COPY statement format.

To copy from one Connect:Direct platform to another, refer to the appropriate COPY FROM and TO sections for those environments. For example, if the source file is located on a Connect:Direct Tandem node, refer to the COPY FROM information for Connect:Direct Tandem. If the destination for the file is a Connect:Direct OS/390 node, refer to the TO information in this chapter.

Label	Statement	Parameters	
stepname	**COPY**	**FROM**	**(**
			DSN=data set name/password|FILE=filename
			PNODE | SNODE
			DCB=([model-file-name] [,BLKSIZE=no.-bytes] [,DEN=0 | 1 | 2 | 3 | 4] [,DSORG=DA | IS | PS | PO | VSAM] [,KEYLEN=no.-bytes] [,LIMCT=no.-blocks-or-tracks] [,LRECL=no.-bytes] [,OPTCD=W | Q | Z] [,RECFM=record-format] [,RKP=first-byte-of-record-key] [,TRTCH=C | E | T | ET | COMP | NOCOMP])
			DISP=([OLD | SHR] ,[KEEP | DELETE] ,[KEEP | DELETE])
			RESGDG=Sub |Run
			LABEL=([file-sequence-number] ,[SL | AL | BLP | LTM | NL] ,[PASSWORD | NOPWREAD] ,[IN | OUT] ,[RETPD=nnnn | EXPDT=[yyddd | yyyy/ddd]])
			MSVGP=MS-group-name
			UNIT=([unit-address | device-type | group-name] ,[unit-count | P])
			VOL=([PRIVATE],[RETAIN] ,[volume-sequence-no] ,[volume-count] ,[SER=(serial-no [,serial-no,...])])| ([SER=(serial-no ,[serial-no,...])| ,REF=dsn])
			ALIAS=Y | N
			EXCLUDE=(generic | member | (start-range/stop-range) | list)
			PDS.DIRectory=Y | N
			REPLACE | NOREPLACE

Label	Statement	Parameters	
			SELECT=(member | generic | (*) | (member, [new-name],[NR | R]) | (generic,, [NR | R]) (start-range/stop-range,,[NR | R]) | list)
			BUFND=number
			DBPARMS=(dbid, dbsubid)
			SQL=('sql-statement' [,'sql-statement',...] | 'DSN=data-set-name')
			IOEXIT=exit-name | (exit-name [,parameter,...])
			DATAEXIT=exit-name | (exit-name [,parameter,...])
			SYSOPTS="DBCS=(tablename,so,si,PAD|PAD=pc)" "DATATYPE=**text**|binary" "XLATE=**no**|yes" "STRIP.BLANKS=**no**|yes" "PERMISS=nnn" "PRECOMP=yes|no"
			DATATYPE=**text**|binary
			XLATE=**no**|yes
			STRIP.BLANKS=**no**|yes
			PERMISS=nnn
			)
		TO	**(**
			DSN=data set name/password|FILE=filename
			PNODE | SNODE
			TYPE=typekey
			DCB=([model-file-name] [,BLKSIZE=no.-bytes] [,DEN=0 | 1 | 2 | 3 | 4] [,DSORG=DA | IS | PS | PO | VSAM] [,KEYLEN=no.-bytes] [,LIMCT=no.-blocks-or-tracks] [,LRECL=no.-bytes] [,OPTCD=W | Q | Z] [,RECFM=record-format] [,RKP=first-byte-of-record-key] [,TRTCH=C | E | T | ET | COMP | NOCOMP])

Label	Statement	Parameters	
			DISP=([NEW | OLD | MOD | RPL | SHR] ,[KEEP | CATLG] ,[KEEP | CATLG | DELETE])
			AVGREC=U | K | M
			DATACLAS=data-class-name
			DSNTYPE=PDS | LIBRARY
			KEYLEN=bytes
			KEYOFF=offset-to-key
			LIKE=model-data-set-name
			LRECL=bytes
			MGMTCLAS=management-class-name
			RECORG=KS | ES | RR | LS
			SECMODEL=(profile-name [,GENERIC])
			STORCLAS=storage-class-name
			LABEL=([file-sequence-number] ,[SL | AL | BLP | LTM | NL] ,[PASSWORD | NOPWREAD] ,[IN | OUT] ,[RETPD=nnnn | EXPDT=[yyddd | yyyy/ddd]])
			MSVGP=MS-group-name
			SPACE=(CYL | TRK | blk, (prim , [sec] , [dir]), [RLSE] , [CONTIG] , [ROUND]) | (av-rec-len , (primary-rcds , [secondary-rcds] , [dir]))
			UNIT=([unit-address | device-type | group-name] , [unit-count | P])
			VOL=([PRIVATE],[RETAIN] ,[volume-sequence-no] ,[volume-count] ,[SER=(serial-no [,serial-no,...])])| ([SER=(serial-no ,[serial-no,...])|,REF=dsn])
			BUFND=number
			IOEXIT=exit-name | (exit-name [,parameter,...])

Label	Statement	Parameters	
			DATAEXIT=exit-name | (exit-name [,parameter,...])
			SYSOUT=(sysout_parameter1, sysout_parameter2,. . .)
			SYSOPTS="UNIQUE=YES" "DBCS=(tablename,so,si,PAD|PAD=pc)" "parameter1[,parameter2,...]" “DATATYPE=**text**|binary” “XLATE=**no**|yes” “STRIP.BLANKS=**no**|yes” “PERMISS=nnn” “SYSOUT=(sysout_keyword1, sysout_keyword2, . .)”
			DATATYPE=**text**|binary
			XLATE=**no**|yes
			STRIP.BLANKS=**no**|yes
			PERMISS=nnn
			)
			CKPT=nK | nM
			COMPRESS [[PRIMEchar= X’40’ | X’xx’ | C’c’] | EXTended]

Field Descriptions

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source file characteristics.

(FROM) DSN = data set name/password | FILE=filename
specifies the source data set name when used with the FROM parameter. Data set names are verified based on the standard OS/390 data set name conventions. If the data set name does not follow OS/390 naming conventions, the data set name must be enclosed in single quotation marks to allow for special characters. Note that GDG data sets can be copied by either the relative generation number or the absolute generation

number. Also, you can submit a Process from a GDG using the relative generation number.

Note: DSN is optional when used with the IOEXIT parameter.

When FROM DSN=NULLFILE is specified, a null file is copied. The NULLFILE option must be accompanied by any DCB information for the output file on either the FROM or TO parameter of the COPY statement. At a minimum, a block size must be specified. This allows you to allocate a file using the COPY function by specifying a disposition of (NEW,CATLG).

If the source data set being copied from requires a password for read or the destination data set requires a password for write, the password may be specified in the COPY statement after the data set name. A slash (/) must follow the data set name and precede the password. This password is used at data set allocation. If it is not correct, OS/390 issues a WTOR requesting the password when Connect:Direct opens the data set. For example:

```
COPY FROM DSN=data-set-name/pwd...
```

If the dataset is an HFS file, the filename must begin with a slash (/). The name is limited to a maximum of 255 characters. It does not have to be enclosed in quotes. For example:

```
DSN=/u/directory/subdirectory/anotherdirectory/filename
```

TO

specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN = data set name/password | FILE=filename

specifies the destination data set name when used with the TO parameter.

Note: DSN is optional when used with the IOEXIT parameter.

If the data set name does not follow standard OS/390 data set name conventions, the data set name must be enclosed in single quotation marks to allow for special characters.

If the data set being copied from requires a password for read or the data set being copied to requires a password for write, the password may be specified in the COPY statement after the data set name. A slash (/) must follow the data set name and precede the password. This password is used at data set allocation. If it is not correct, OS/390 issues a WTOR requesting the password when Connect:Direct OS/390 software opens the data set. For example:

```
COPY TO DSN=data-set-name/pwd...
```

If the dataset is an HFS file, the filename must begin with a slash (/). The name is limited to a maximum of 255 characters. It does not have to be enclosed in quotes. For example:

```
DSN=/u/directory/subdirectory/anotherdirectory/filename
```

Optional Parameters

ALIAS = Y | N

specifies whether aliases are copied when their associated member names are copied. The default is ALIAS=Y.

Guidelines for alias entries when ALIAS=Y follow:

- If the name specified in the SELECT parameter is a true member name, that member is sent and any of its aliases are sent unless they are specified in the EXCLUDE parameter. If the **R** subparameter of the SELECT parameter is specified, it also applies to the aliases.
- If the name specified in the SELECT parameter is an alias, any other aliases plus their associated true member are sent unless they are specified in the EXCLUDE statement. If the **R** subparameter of the SELECT parameter is specified, it applies to the true member and the other aliases sent.
- If the true member name is specifically excluded and any of its aliases are selected, a completion code of **4** results. When copying a PDS with aliases but no corresponding true members, Connect:Direct software does not copy the aliases and returns a completion code of **4**.

Guidelines for alias entries when ALIAS=N follow:

- If the name specified in the SELECT parameter is a true member name, only the member is sent.
- If the true member name is also specified and if the name specified in the SELECT parameter is an alias, then the directory of the alias that is specified is sent. If the R subparameter is specified, it must be used with the true member name or it results in a completion code of **4**. No entry is then made for that alias.
- If the true member name has not been specified and if the name specified in the SELECT parameter is an alias, then it results in a completion code of **4**, and no entry is made for that alias.

AVGREC = U | K | M

requests that the data set be allocated in records. The primary and secondary space quantities represent number of records requested in units, thousands, or millions of records. This parameter is mutually exclusive with the TRK/CYL subparameter of the SPACE parameter. This parameter is only valid on systems with SMS support.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you

to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

U specifies a record request where primary and secondary space quantities are the number of records requested. The value of the primary space request is a multiple of **1**.

K specifies a record request where primary and secondary space quantities are the number of records requested in thousands of records. The value of the primary space request is a multiple of 1024.

M specifies a record request where primary and secondary space quantities are the number of records requested in millions of records. The value of the primary space request is a multiple of 1,048,576.

BUFND = number
specifies the number of I/O buffers VSAM will use for transmitting data between virtual and auxiliary storage. A buffer is the size of a control interval in the data component.

Valid values range from 1-510. The default is **2**. Increasing this number generally improves I/O performance, but requires more memory.

CKPT = nK | nM
specifies the byte interval for checkpoint support, which allows restart of interrupted transmissions at the last valid transmission point, avoiding the need to restart transmission from the beginning. (**K** denotes thousands; **M** denotes millions.) A checkpoint value of zero stops automatic checkpointing.

Connect:Direct converts the value to a block boundary, and a data transmission checkpoint is taken at that position.

For Connect:Direct OS/400: Connect:Direct OS/400 does not support checkpointing for versions prior to 1.4.00.

If DISP=MOD is specified on the COPY TO statement, checkpoint-restart is not possible; duplicate data would be difficult to detect.

Sequential files, VSAM files, or PDSs can be checkpointed. For PDS-to-PDS transmission, any value specified causes Connect:Direct OS/390 to checkpoint each member. Note that sequential-to-PDS and PDS-to-sequential transmissions cannot be checkpointed.

COMPRESS [[PRIMEchar = X'40' | X'xx' | C'c'] | EXTended]
specifies that the data is to be compressed, reducing the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at its destination. The default subparameter for the COMPRESS parameter is PRIMEchar=X'40'. COMPRESS PRIMEchar is used for text data or single-character repetitive data. COMPRESS EXTended generally offers improved compression for all types of data.

Note: Compression is CPU-intensive, and its effectiveness is data dependent. It should only be used if its benefits are known.

If compression is specified, Connect:Direct reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character are compressed to 1 byte.
- Repetitive occurrences (ranging from 3-63) of any other character are compressed to 2 bytes.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X'40').

EXTended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited, CPU is available, and data is repetitive.

The following are valid options for EXTended:

CMPrlevel determines the compression level. The valid value range is 1-9. Level **1** is the fastest compression, but it offers the lowest degree of compression. A higher value produces a higher quality of compression, but has a slower rate of compression. The default is 1.

WINdowsize determines the size of the compression window or history buffer. This memory is above the line. The valid values are 8-15. Higher windowsize specifications increase the degree of compression and use more virtual memory. Size **8** uses **1** KB of memory where Size 15 requires 128 KB of memory. The default is 13.

MEMlevel identifies how much virtual memory is allocated to maintain the internal compression state. This memory is above the line memory. The valid value range is 1-9. Level **1** requires the least memory (1K), but it reduces the degree of compression. Level **9** provides the fastest speed, but it uses the most memory (256K). The default is 4.

The following example shows one way to specify the various EXTended options in a COPY statement:

```
COMPRESS EXT = ( CMP=4
                 WIN=12
                 MEM=8)
```

DATACLAS = data-class-name

requests the data class for a new data set. The class selected must have been previously defined by the SMS administrator. This parameter may be used with VSAM data sets, sequential data sets, or partitioned data sets.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

data-class-name is the 1-8 character name of the data class to which this data set belongs. The name of the data class is assigned by the SMS administrator. The user should contact the SMS administrator for a valid list of the available data classes.

You can use **data-class-name=$$$$$$$$** to propagate a data class from an input file to the receiving node and to the output data set. When DATACLAS is propagated, the DCB and SPACE attributes are ignored.

DCB =([model-file-name]
[,BLKSIZE = no.-bytes]
[,DEN = 0 | 1 | 2 | 3 | 4]
[,DSORG = DA | IS | PS | PO | VSAM]
[,KEYLEN = no.-bytes]
[,LIMCT = no.-blocks-or-tracks]
[,LRECL = no.-bytes]
[,OPTCD = W | Q | Z]
[,RECFM = record-format]
[,RKP = first-byte-of-record-key]
[,TRTCH = C | E | T | ET | COMP | NOCOMP])

specifies attributes to be used in allocating source and destination files. For existing source and destination files, DCB attributes are determined from the operating system unless specified. For a new destination file, the DCB attributes of the source file are used to allocate the destination file unless DCB information is provided in the Process.

model-file-name specifies a model data set control block (DSCB).

BLKSIZE specifies the length in bytes of the block. The minimum length is 18 bytes and the maximum length is 32,760 bytes.

BLKSIZE=0 for all RECFMs except RECFM=U allows the operating system to select the block size for the physical DASD. For RECFM=U the operating system creates a null file Connect:Direct OS/390 does not allow this combination. The Process fails with an SVSG005I message.

DEN specifies the magnetic tape mode setting. The values for the DEN parameter for 7- and 9-track tape are shown in the following table. When coded together, the DEN and TRTCH values are used to select a tape device for allocation by Connect:Direct.

DEN	7-Track Tape	9-Track Tape
0	200 bpi	-
1	556 bpi	-
2	800 bpi	800 bpi (NRZI) NRZI is Non-Return-to-Zero Inverted recording mode
3	-	1600 bpi (PE) PE is Phase Encoded recording mode
4	-	6250 bpi (GCR) GCR is Group Coded Recording mode

DSORG specifies the file organization. Supported file organizations are: BDAM, ISAM, PO, PS, and VSAM.

KEYLEN specifies the length of the keys used in a file. The maximum length in bytes is 255.

LIMCT specifies the blocks or tracks searched to find a free block or available space.

LRECL specifies the record length in bytes.

Note: When using RECFM=V or RECFM=VB type files, the LRECL value must be at least the size of the largest record in the file plus 4 bytes. If RECFM=V, the BLKSIZE value must be at least the LRECL value plus another 4 bytes. If RECFM=V, the BLKSIZE value does not need to be an even multiple of LRECL.

For Connect:Direct Tandem: An entry-sequenced file coming from Tandem to an IBM PS/VB file must be specified with an LRECL 4 bytes larger than the Tandem file. This is specified on the COPY TO statement to account for a 4-byte-length area required on the IBM file but not required on Tandem.

OPTCD specifies optional processing associated with this file. This specification only applies to this file and is not automatically applied to the other files involved in the COPY operation. Valid options are as follows:

W performs write validity checks on direct access storage devices.

Q performs ASCII-to-EBCDIC conversion for input files and EBCDIC-to-ASCII conversion for output files. Note that Q is the default and is only used for AL-labeled tape files.

Z performs reduced error recovery for tape files.

RECFM specifies the format of the records in the file. Any valid record format, such as F (Fixed), FA (Fixed ASA printer control), FB (Fixed Block), FBA (Fixed Block ANSI carriage control), FM (Fixed Machine code control character), U (Undefined), V (Variable), VB (Variable Block), VBA (Variable Block ASA printer control), VBM (Variable Block Machine code control character), VS (Variable Spanned), and VBS (Variable Block Spanned), can be specified. For FDR volumes and DFDSS files, you must specify RECFM=U on the FROM parameter.

Note: When transmitting VBS and VS files to a non-370 platform, the record descriptor word (RDW) is transmitted to the receiving node.

For Connect:Direct OpenVMS: An OpenVMS file with a record format of undefined (U) cannot be copied to OS/390.

RKP specifies the position of the first byte of the record key within each logical record. The beginning byte of a record is addressed as **0**.

TRTCH specifies the magnetic tape mode setting. When coded together, the TRTCH and DEN values are used to select a tape device for allocation by Connect:Direct. Valid options are as follows:

C specifies data conversion, odd parity, and no translation.

E specifies no data conversion, even parity, and no translation.

T specifies no data conversion, odd parity, and BCD or EBCDIC translation.

ET specifies no data conversion, even parity, and BCD or EBCDIC translation.

COMP is a feature for 3480X tape drives only. It enables Improved Data Recording Capability (IDRC), which compresses the data. COMP overrides the system-wide IDRC setting for no compression. If you are specifying COMP, you must also include a UNIT= parameter that specifies either 3480X or a systems-programmer-defined name equivalent to a 3480X tape drive.

NOCOMP overrides the system-wide IDRC setting for compression. It applies to 3480X tape drives only.

(FROM) DISP =([OLD | SHR]
, [KEEP | DELETE]
, [KEEP | DELETE])

specifies the status of the file and what is done with the file after notification of successful transmission. Subparameters are as follows:

First Subparameter specifies the status of the file prior to execution of the Process. This subparameter applies to all files. Options for this subparameter are as follows:

OLD specifies that the source file existed before the Process began executing and the Process is given exclusive control of the file.

SHR specifies that the source file existed before the Process began executing and that the file can be used simultaneously by another job or Process. The default is SHR.

Second Subparameter specifies the disposition of the file following a normal Process step termination resulting in a zero completion code. This subparameter applies to non-VSAM files. Valid source file dispositions are as follows:

KEEP specifies that the system keeps the file after the Process step completes.

DELETE specifies that the system deletes the file after the Process step completes successfully.

Third Subparameter specifies the disposition of the file after an abnormal Process step termination resulting in a non-zero completion code. This subparameter applies to non-VSAM files. Valid source file dispositions are as follows:

KEEP specifies that the system keeps the file after the Process step terminates abnormally or with a non-zero return code.

DELETE specifies the system deletes the file if the Process step terminates abnormally.

(TO) DISP = ([NEW | OLD | MOD | RPL | SHR]
, [KEEP | CATLG]
, [KEEP | CATLG | DELETE])

specifies the status of the file on the receiving node. Subparameters are as follows:

Note: If you are decompressing a file using CDSACOMP, you cannot allocate VSAM files as DISP=(NEW,CATLG). You must predefine the VSAM output file.

For tape files only: If a COPY statement specifies DISP=(NEW,CATLG) on the TO clause and the tape file already exists as a cataloged file, the COPY statement fails.

First Subparameter specifies the status of the file before the Process executes. Only the OLD and RPL dispositions apply to VSAM files. Valid options for this subparameter are as follows:

NEW specifies that the Process step will create the destination file. NEW is the default.

OLD specifies that the destination file already exists. The Process will have exclusive control of the file. If DISP=OLD, the destination file may be a VSAM file, SAM file, or PDS.

MOD specifies that the Process step will modify the SAM file by appending data at the end of the file. If a system failure occurs when MOD is specified, the system is designed not to restart even if CKPT is specified; data loss or duplication would be difficult to detect.

RPL specifies that the destination file will replace any existing file or, if none exists, will allocate a new file. DISP=RPL may be specified for SAM or VSAM files. If the file is VSAM, it must be defined with the REUSE attribute, which specifies that the file can be opened and reset to the beginning.

SHR specifies that the destination file already exists. The file can be used simultaneously by another job or Process.

Second Subparameter specifies the normal termination disposition, but does not apply to VSAM files. Valid destination file dispositions are as follows:

KEEP specifies that the system keeps the file after the Process step completes. If DISP=(NEW,KEEP), a volume serial number also must be specified.

CATLG specifies that the system keeps the file after the Process step completes and an entry is to be placed in the catalog. CATLG is the default.

Third Subparameter specifies the disposition of the file after an abnormal Process step termination resulting in a non-zero completion code. This subparameter applies only to non-VSAM files. Valid destination file dispositions are as follows:

KEEP specifies that the system keeps the file after the Process step terminates abnormally or with a non-zero return code.

CATLG specifies that the system keeps the file after the Process step terminates abnormally and that an entry is to be placed in the catalog.

DELETE specifies the system deletes the file if the Process step terminates abnormally.

DSNTYPE = LIBRARY | PDS
defines a specific data set organization for an SMS controlled data set of a PARTITIONed type.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

LIBRARY specifies an SMS data set organization of PDSE (partitioned data set extended).

PDS specifies partitioned data set organization, but the data set is under SMS control.

EXCLUDE = (generic | member | (start-range/stop-range) | list)
specifies criteria that identifies the PDS members that are not to be copied. The EXCLUDE parameter can be specified only in the FROM clause of the COPY statement. EXCLUDE allows the user to make exceptions to members specified generically or by range in the SELECT option. See *PDS Guidelines* on page 18 for the override priority for the SELECT and EXCLUDE parameters.

Note: EXCLUDE cannot be used if a member name is specified as part of the FROM DSN or TO DSN.

---

generic specifies a generic member name. For example, if CDM* is specified, all member names beginning with CDM are excluded. The only way to override an excluded generic is to specify an individual member name in the SELECT parameter.

member specifies an individual member name. When a member is specified in the EXCLUDE parameter, its exclusion cannot be overridden.

start-range specifies the first name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the EXCLUDE statement, the first and last members specified in the range, as well as all members between, are not copied.

stop-range specifies the last name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the EXCLUDE statement, the first and last members specified in the range, as well as all members between, are not copied.

Note: The only way to override an excluded range is to specify an individual member name in the SELECT parameter.

---

list specifies a list of member names.

IOEXIT = exit-name | (exit-name[,parameter,...])
indicates that a user-written program is to be called to perform I/O requests for the associated data. See the *Connect:Direct OS/390 Administration Guide* for instructions on writing I/O exits.

exit-name specifies the name of the user-written program to be given control for I/O-related requests for the associated data.

parameter specifies a parameter, or list of parameters, to be passed to the specified exit. For valid parameter formats, refer to the parameters described in the *RUN TASK Statement* chapter.

DATAEXIT = exit-name | (exit-name[,parameter,...])
indicates that a user-written program is to be called. This exit is similar to the I/O Exit except it does not require the same I/O management. See the *Connect:Direct OS/390 Administration Guide* for instructions on writing Data exits.

exit-name specifies the name of the user-written program that receives control for data requests.

parameter specifies a parameter, or list of parameters, to be passed to the specified exit. For valid parameter formats, refer to the parameters described in the *RUN TASK Statement* chapter.

KEYLEN = bytes
specifies, in bytes, the length of the keys used in the file. This parameter is valid for SMS data sets. The value must be a decimal integer from 0-255 for non-VSAM data sets or 1-255 for VSAM data sets.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

KEYOFF = offset-to-key
specifies the offset within the record to the first byte of the key in a new VSAM KS data set. The first byte of the record is byte 0.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

offset-to-key is the position of the first byte of the key in the record. The value ranges from 0-32760.

LABEL =([file-sequence-number]
,[SL | AL | BLP | LTM | NL]
,[PASSWORD | NOPWREAD]
,[IN | OUT]
,[RETPD = nnnn | EXPDT = [yyddd | yyyy/ddd]])
specifies label information for the tape.

file-sequence-number specifies the relative file position on the tape.

The label type is designated as follows:

SL specifies IBM standard labels.

AL specifies American National Standard labels.

BLP specifies bypass label processing.

LTM specifies bypass leading tape marks.

NL specifies no labels.

PASSWORD specifies that a password must be supplied by the operator or user before the data set can be accessed.

NOPWREAD indicates that a password is not required to read the data set.

IN specifies that a BSAM data set opened for INOUT or a BDAM data set opened for UPDAT is to be read only.

OUT specifies that a BSAM data set opened for OUTIN or OUTINX is to be write only.

RETPD specifies the retention period for the data set in days, where nnnn is 1-4 digits.

EXPDT specifies the expiration date for the data set, where yyddd or yyyy/ddd is a valid Julian date.

Note: No slash is used in the 4-digit year EXPDT parameter of a process submitted on a UNIX or Windows platform.

LIKE = model-data-set-name

requests that allocation attributes for a new data set be copied from an existing cataloged data set. Any or all of the following attributes are copied to the new data set: RECORG or RECFM, LRECL, KEYLEN, KEYOFF, DSNTYPE, AVGREC, and SPACE. Any attributes specified for the data set override the values from the model data set. Neither EXPDT nor RETPD are copied from the model data set.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

model-data-set-name is the name of the data set from which the allocation attributes are copied.

LRECL = bytes

specifies the length, in bytes, of the records in the new data set. This parameter is valid for SMS VSAM data sets. LRECL must not be specified with RECORG=LS type data sets.

Note: When RECFM=V or RECFM=VB type files are used, the LRECL value must be at least the size of the largest record in the file plus 4 bytes. If RECFM=V, the BLKSIZE value must be at least the LRECL value plus another 4 bytes. If RECFM=V, the BLKSIZE value does not need to be an even multiple of LRECL.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

bytes is the length of the records in the data set. For non-VSAM data sets, valid values range from 1-32760 bytes. For VSAM data sets, valid values range from 1-32761 bytes. The LRECL must be longer than the KEYLEN value for VSAM KSDS.

MGMTCLAS = management-class-name

determines the previously defined management class to which a new data set belongs. Available classes are determined and named by the SMS administrator. For example, attributes in this class can determine when a data set is migrated or backed up. The system Automatic Class Selection (ACS) routine can override this parameter.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

management-class-name is the 1-8 character name of the management class to which a data set belongs. The name of the management class is assigned by the SMS administrator.

You can use **management-class-name=$$$$$$$$** to propagate a management class from an input file to the receiving node and to the output data set.

MSVGP = MS-group-name

specifies the group of mass storage volumes that reside on a mass storage system (MSS) device. This must be a valid DD (data definition) name, ranging from 1-8 alphanumeric characters with the first character alphabetic.

NOREPLACE

specifies that members of a sending PDS do not replace existing members of the same name at the receiving PDS. The NOREPLACE parameter takes effect only when the FROM and TO files are PDSs. The default is REPLACE. Note that NOREPLACE applies to an entire PDS as opposed to the NR option of the SELECT parameter, which applies to members within a PDS.

PDS.DIRectory = Y | N

specifies whether user-related information in the directory is sent.

If the PDS is a loadlib and PDS.DIR is set to NO, the directory information is lost and the modules are no longer executable.

PNODE

is the primary node, defining the direction of transfer (with SNODE). When PNODE is coded with the FROM parameter, a *send* takes place. When PNODE is coded with the TO parameter, a *receive* takes place. PNODE is the default for the FROM parameter.

RECORG = KS | ES | RR | LS
defines the organization of records in a new VSAM data set. If RECORG is not specified, then SMS assumes that the data set is either a physical sequential (PS) data set or a partitioned (PO) data set.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

KS specifies a VSAM key-sequenced data set.

ES specifies a VSAM entry-sequenced data set.

RR specifies a VSAM relative record data set.

LS specifies a VSAM linear data set.

REPLACE
specifies that the sending PDS replaces members of the same name at the receiving PDS. REPLACE is the default.

RESGDG = Sub | Run
allows users to specify submit or execution time for resolution input GDGs. The parameter, valid only for input GDGs, is coded on the FROM clause of the COPY statement.

Sub specifies GDG resolution at Process submit time. Sub is the default.

If you specify GDG resolution at Process submit time, note the following conditions:

- Before the submit, if you perform a single session cross-domain signon to another node or a multiple session signon to another node (for example, the API doing the submit is logged on to another DTF), GDG resolution occurs at execution time, regardless of the parameter specified on the PROCESS statement.

  It is possible that the API is running on a different system than the DTF or a GDG of the same name is on both systems and the wrong generation of the DTF GDG might be copied. This error can also occur if the GDG to be copied is not on shared DASD.

- If the LOCATE for the data set fails at Process submit time, GDG resolution occurs at execution time.

Run specifies GDG resolution at Process run or execution time within the DTF.

SECMODEL = (profile-name [,GENERIC])
copies an existing Resource Access Control Facility (RACF) profile as the discrete profile for a new data set. The following information is copied along with the profile: OWNER, ID, UACC, AUDIT/GLOBALAUDIT, ERASE, LEVEL, DATA, WARNING, and SECLEVEL.

profile-name is the name of the model RACF profile, discrete data set profile, or generic data set profile to be copied to the discrete data set profile created for the new data set.

GENERIC identifies that the profile-name refers to a generic data set profile.

SELECT =(member | generic | (*) | (member, [newname]
,[NR | R]) | (generic,, [NR | R]) (start-range/stop-range
,,[NR | R]) | list)

specifies selection criteria by which PDS members are to be copied. The SELECT parameter can be specified only with the FROM parameter. See *PDS Guidelines* on page 18 for the override priority for the SELECT and EXCLUDE parameters.

Various specifications can be combined in a list after the SELECT parameter.

If SELECT is specified and EXCLUDE is not specified, all selected members are copied. If SELECT is not specified and EXCLUDE is specified, all members not excluded are copied.

generic specifies a generic member name. If CDM* is specified as either a parameter or subparameter, all member names beginning with CDM are selected for copying.

(*) represents a global generic. A global generic indicates that all members of the file are to be included. A global generic is valid only with the SELECT parameter.

When a generic is specified in the SELECT parameter, its selection can be overridden with any type of specification in the EXCLUDE parameter.

When using a generic and specifying NR or R, the second positional parameter (NEWNAME) must be null.

member specifies an individual member name. Note that specifying a member name in the DSN is the same as specifying a SELECT statement with only that member.

The only way to override a selection by member name is to specify that member name in the EXCLUDE parameter.

newname specifies a new name for a member. The NEWNAME parameter must be null, if a generic name or range is used in the first subparameter position.

NR specifies that a member does not replace an existing member of the same name at the receiving PDS. **NR** overrides the REPLACE parameter. **R** is the default.

When used with NEWNAME, **NR** applies to the NEWNAME and not to the original member name. When used with a generic name or with a range, **NR** applies to all members selected for that criteria.

Note: NOREPLACE applies to an entire PDS as opposed to **NR**, which applies to members within a PDS.

R specifies that a member replaces an existing member of the same name at the receiving PDS. **R** overrides the NOREPLACE parameter.

When used with NEWNAME, **R** applies to the NEWNAME and not to the original member name. When used with a generic name or with a range, **R** applies to all members selected for that criteria.

start-range specifies the first name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the SELECT statement, the first and last members specified in the range, as well as all members between, are copied.

stop-range specifies the last name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the SELECT statement, the first and last members specified in the range, as well as all members between, are copied.

When a range in the SELECT parameter is specified, its selection can be overridden with any type of specification in the EXCLUDE parameter.

The second positional parameter (NEWNAME) of SELECT must be null when using a range and specifying **NR** or **R**.

list specifies a list of selected members.

SNODE

is the secondary node, defining the direction of transfer (with PNODE). When SNODE is coded with the FROM parameter, a *receive* takes place. When SNODE is coded with the TO parameter, a *send* takes place. SNODE is the default for the TO parameter.

If you do not code either the PNODE or SNODE parameter, the file is sent from the PNODE to the SNODE.

SPACE =(CYL | TRK | <u>blk</u> | av-rec-len, (prim , [sec] , [dir]) , [RLSE] , [CONTIG], [ROUND])

specifies the amount of storage to be allocated for new files on the destination node. If SPACE is specified, the DISP of the destination file must be NEW. If SPACE is not specified in the Process or the TYPE file, and the DISP is NEW, the output file is allocated as follows:

- If no secondary space allocation exists on the input file, then the primary amount of space allocated (rather than used) is used to allocate the NEW output file.
- If secondary space exists on the input file, then space used (rather than allocated) is used to allocate the output file and it is allocated with secondary extents.

If the AVGREC parameter is also coded in the TO clause of the COPY statement, the allocation of the data set is done on a record size basis instead of TRK, CYL, or blk. The TRK, CYL and blk subparameters are not valid when the AVGREC parameter is coded in the COPY TO statement.

Valid choices for this parameter are as follows:

CYL specifies that space will be allocated by cylinder.

TRK specifies that space will be allocated by track.

<u>blk</u> specifies that space will be allocated by the average block length of the data. The system computes the number of tracks to be allocated. If the subparameter ROUND is also specified, the system allocates the space in cylinders. ROUND is preferred because allocation is performed on cylinders in a device-independent manner. If no space information is coded, allocation is in blocks, due to device dependencies.

av-rec-length specifies the average record length, in bytes, of the data. The system computes the BLKSIZE and the number of tracks to allocate. The record length must be a decimal value from 1-65535.

prim specifies the primary allocation of storage (number of units).

sec specifies the secondary allocation of storage (number of units).

dir specifies the number of PDS directory blocks to be created in the file.

RLSE specifies the release of the unused storage allocated to the output file.

CONTIG specifies that the storage for the primary allocation must be contiguous.

ROUND specifies that the storage allocated by average block length is rounded to an integral number of cylinders.

STORCLAS = storage-class-name
specifies the storage class to which a new data set is assigned. The storage class name must be previously defined to the SMS system. Storage class defines a storage service level for the data set and replaces the UNIT and VOLUME parameters for SMS data sets. None of the attributes in the storage class can be overridden by JCL parameters, and an ACS routine can override the specified class.

For platforms other than OS/390: All SMS parameters must be specified as subparameters of the SYSOPTS parameter. SYSOPTS is a mechanism that allows you to pass system-specific parameters between platforms. See the COPY statement of the appropriate platform for syntax requirements for the SYSOPTS parameter.

storage-class-name is the 1-8 character name of the storage class to which the data set is assigned. These names are defined by the SMS administrator.

You can use **storage-class-name=$$$$$$$$** to propagate a storage class from an input file to the receiving node and to the output data set.

SYSOUT=(sysout_keyword1, sysout_keyword2, . .)
Refer to the *Connect:Direct Spool Transfer User's Guide* for a list and description of the SYSOUT keywords.

SYSOPTS = "UNIQUE=YES"
"DBCS=(tablename,so,si,PAD|PAD=pc)"
"parameter1[parameter2,...]"
"DATATYPE=text|binary"
"XLATE=no|yes"
"STRIP.BLANKS=no|yes"
"PERMISS=nnn"
"DECOMP=yes|no"
"SYSOUT=(sysout_keyword1, sysout_keyword2, . .)"
specifies system operation parameters.

UNIQUE=YES specifies that the Unique Member Name Allocation Exit (AXUNIQ) will be invoked in order to force the PDS member name to be unique on the OS/390 TO node.

Note: The initialization parameters must specify ALLOCATION.EXIT=AXUNIQ in order to use SYSOPTS="UNIQUE=YES".

DBCS=(tablename,so,si,PAD|PAD=pc) is used to invoke the double-byte character set translation facility.

tablename is the name of the requested DBCS translation table. The tablename is required with DBCS. If you only specify **tablename**, you do not need to enclose the parameters in parentheses.

The following tables are provided by Connect:Direct:

EBCXKSC translates data from host EBCDIC to ASCII KS5601.

KSCXEBC translates data from ASCII KS5601 to host EBCDIC.

EBCXKPC translates data from host EBCDIC to DBCS-PC Korean.

KPCXEBC translates data from DBCS-PC Korean to host EBCDIC.

NHCXBG5 translates data from Chinese new host code to Chinese Big5.

BG5XNHC translates data from Chinese Big5 to Chinese new host code.

NHCXC55 translates data from Chinese new host code to Chinese 5550.

C55XNHC translates data from Chinese 5550 to Chinese new host code.

so is the SHIFT-OUT character denoting a shift from single-byte character set (SBCS) to double-byte character set (DBCS) mode. The default is the IBM standard x'0E'.

si is the SHIFT-IN character denoting a shift from DBCS to SBCS mode. The default is the IBM standard x'0F'.

Note: NOSO indicates no shift-out or shift-in character and is denoted by the use of x'00' for the SO and SI characters. NOSO is used when the data is not in mixed form and is assumed to contain all DBCS characters.

PAD|PAD=pc specifies that padding characters are in use. When DBCS data is translated from EBCDIC to ASCII, **PAD** specifies that the SHIFT-OUT and SHIFT-IN characters will be replaced by a pad character. This allows the displacement of fields within a record to remain unchanged during translation.

When DBCS data is translated from ASCII to EBCDIC, **PAD** specifies that the input ASCII DBCS file is in a padded format. The character immediately preceding a DBCS character or string will be overlaid by the SHIFT-OUT character. The character immediately following a DBCS character or string will be overlaid with the SHIFT-IN character. This allows the displacement of fields within a record to remain unchanged during translation.

pc is the pad character to be used during EBCDIC to ASCII translation. **pc** is ignored for ASCII to EBCDIC translations. The default value for **pc** is x'00'.

parameter1[parameter2,...] is used in conjunction with the IOEXIT parameter. It specifies the parameters to be passed to the I/O exit for copies from a non-370 node to a Connect:Direct OS/390 node.

DATATYPE = text | binary specifies the type of data in the file. Data can be either text or binary format. Can be a key word. Valid for HFS files only.

XLATE = no | yes specifies whether ASCII/EBCDIC character translation occurs. Can be a key word. Valid for HFS files only.

STRIP.BLANKS = no | yes specifies whether trailing blanks characters are removed from the text record before writing or transmitting the record. Can be a key word. Valid for HFS files only.

PERMISS = nnn specifies the HFS file permissions for a file being created. This subparameter is ignored if the file already exists. Can be a key word. Valid for HFS files only.

The first digit indicates the owner's file permissions, the second digit indicates the owner's group's file permissions, and the third digit indicates the file permissions for all others.

The following table shows permission values:

Value	Permissions
0	No file access allowed
1	Execute access allowed
2	Write access allowed
3	Write and Execute access allowed
4	Read access allowed
5	Read and Execute access allowed
6	Read and Write access allowed
7	Read, Write, and Execute access allowed

For example, permiss=**634** indicates that the file owner has read and write permissions, the owner's group has write and execute permissions, and all others are allowed only read access. Permiss=**751** indicates that the file owner has read, write, and execute access, the owner's group has read and execute access, and all others have execute access to the file.

PRECOMP=yes|no specifies that Connect:Direct will automatically decompress a file that was precompressed with the CDSACOMP utility. See the *Connect:Direct OS/390 User's Guide* for more information on the CDSACOMP utility.

SYSOUT=(sysout_keyword1, sysout_keyword2, . .) specifies various SYSOUT keywords. See the *Connect:Direct Spool Transfer User's Guide* for a list and description of the SYSOUT keywords.

TYPE = typekey

specifies the entry name of the type defaults file. The type defaults file contains the default file attributes used to allocate the destination file. The typekey is specified only when defaults are requested by the user.

For OS/390 to OpenVMS copies: For OS/390 to OpenVMS copies where the typekey exceeds eight characters, the typekey must be entered into the SYSOPTS parameter on the TO clause of the Connect:Direct OpenVMS COPY statement.

For OpenVMS to OS/390 copies: The typekey must not be greater than eight characters.

UNIT = ([unit-address | device-type | group-name] , [unit-count | P])

specifies the unit address, device type, or user-assigned group name where the file resides or will reside. For SAM-to-SAM copies where the destination file is new and the UNIT parameter is not coded with the TO parameter, the device type from the source file is used. When unit address is 4 bytes in length, begin the value with a slash (/). For example, if the unit address is 1FA2,
UNIT=/1FA2.

Specify a unit-count to allow additional units to be allocated if required, or specify **P** to allocate the same number of units as volumes and then parallel mount the volumes.

VOL = ([PRIVATE],[RETAIN],[volume-sequence-no],[volume-count]
,[SER=(serial-no [,serial-no,...])]) |
([SER=(serial-no,[serial-no,...]) | ,REF=dsn])

specifies the volume serial number(s) containing the file and optional processing associated with the file. If VOL is not specified with the FROM parameter, the file must be cataloged. Valid options are as follows:

PRIVATE specifies allocation of an output file only if the volume is specifically requested and is used for direct access storage devices only.

RETAIN has no significance because Connect:Direct does dynamic deallocation of data sets. However, if RETAIN is omitted, a comma must be coded. If RETAIN is specified, the volume is not retained as it would be in a regular batch.

volume-sequence-no specifies the volume of an existing multivolume data set to be used to begin processing the data set. The volume sequence number must be less than or equal to the number of volumes on which the data set exists or the job fails.

volume-count specifies the maximum number of volumes required by an output file.

SER identifies by serial number the volumes on which the output file resides or will reside.

REF allows you to place a data set on the same volume as the referenced data set. It must be cataloged on the system where it is referenced.

Duplicate Parameter Resolution

If a Process contains duplicate parameters, they are handled in one of the following ways:

- If the duplicates are positional parameters, and both parameters contain a value in the same position, the second occurrence of the value is used.

  For example, if a Process contains two DISP parameters, where the first DISP parameter is DISP=(SHR) and the second DISP parameter is DISP=(OLD), the Process uses DISP=(OLD).

  If the first DISP parameter is DISP=(NEW,CATLG,DELETE) and the second DISP parameter is DISP=(,,KEEP), the Process uses DISP=(NEW,CATLG,KEEP).

- If the duplicates are keyword parameters, and the second occurrence of the parameter contains different keywords than the first occurrence, the Process uses all defined keywords.

Example

In this example, the COPY statement copies a data set from one Connect:Direct OS/390 node to another. A description of the statement follows.

```
STEP1    COPY  FROM  (                                                    -
                     DSN=SRCDATA.SET                                      -
                     DISP=(SHR,DELETE,KEEP)                               -
                     SELECT=(AA*)                                         -
                     EXCLUDE=(AABC,AABF)                                  -
                     PDS.DIR=N                                            -
                     ALIAS=N                                              -
                     PNODE                                                -
                     UNIT=3380                                            -
                     VOL=SER=VOL003                                       -
                     )                                                    -
                COMPRESS                                                  -
                TO    (                                                  -
                     DSN=DESTDATA.SET                                     -
                     DCB=(DSORG=PO,LRECL=80,RECFM=FB,BLKSIZE=3120)        -
                     DISP=(NEW,CATLG)                                     -
                     UNIT=SYSDA                                           -
                     SNODE                                                -
                     )
```

- The copy step is named STEP1.
- The input data set is deleted after successful completion of the Process.
- All members of the PDS SRCDATA.SET beginning with AA, except AABC and AABF, are copied from the PNODE to a new PDS that Connect:Direct allocates on the SNODE.
- The DCB attributes specified in the TO clause of the COPY statement are used for file allocation.
- Unit and volume serial number have been specified on the PNODE; however, only unit is specified on the SNODE.

- Neither user-related PDS directory information or aliases are sent to the SNODE.
- Specifying COMPRESS without a subparameter indicates that blanks will be compressed during transmission and converted back to the original string during decompression.
- Space and directory allocation parameters for the new TO data set will be the same as for the FROM data set.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Unique Member Name Allocation Exit

The Unique Member Name Allocation Exit provides the capability within Connect:Direct OS/390 to enforce a unique PDS member name during COPY processing.

This support is provided for PDS flows that name the member as part of the DSN specification. No support is provided for SELECT/EXCLUDE lists. This release does not provide any support for creating unique data set names.

Exit Features

The AXUNIQ allocation exit ensures that a PDS member name is unique during the receive operation. This is done by modifying the member name if an identical member is found in the destination data set. With this option in place, a Connect:Direct COPY that specifies a member name as part of the DSN= specification never replaces an existing member, even if it is identical to the COPY specification, nor does it fail due to the same condition.

Exit Functions

AXUNIQ is implemented as a Connect:Direct Allocation Exit. This exit is invoked during the COPY preparation (TO node only) prior to any OS/390 allocation activity. The exit, if present, is invoked on every COPY. The specification of SYSOPTS="UNIQUE=YES" in the TO clause of the COPY statement triggers the exit to perform its function. The exit returns for any other invocation.

The following is an example COPY statement:

```
STEP01   COPY  FROM  (DSN=HLQ.FILE.NAME DISP=SHR)                    -
               TO    (DSN=QLH.LIB.PDS(NEWMWM) DISP=OLD              -
                     SYSOPTS="UNIQUE=YES")
```

Required Conditions

The following conditions must be met for the exit to ensure a unique member name is used:

- Only single member transfers are supported.
- Only support for sequential files to a PDS member is provided. No support is provided in this release for PDS member to PDS member.
- The output member must be specified as part of the DSN keyword (either FROM or TO). SELECT and EXCLUDE clauses are not supported with UNIQUE. This case is flagged as an error.
- The member name resolution is performed on the TO node only.
- The UNIQUE support works in either direction, from SNODE or from PNODE.
- The UNIQUE=YES specification, in SYSOPTS, is mutually exclusive with any SMS keywords.
- The ALLOCATION.EXIT initialization parameter must be set to AXUNIQ.

Resolution Method

The resolution of a unique member name is performed as follows:

- The member name specified in the TO DSN (or defaulted from the FROM DSN) is used as a comparison seed. If the name is seven characters or less, a numeral, starting with 1, is appended to the name. This numeric suffix is incremented by 1 until a unique name in the output PDS is achieved.
- If numerals 1 through 9 are exhausted without finding an available name, the name is formed with two numeric values as the suffix. The appended number never exceeds two digits.
- If the seed name does not allow for two numeric values, due to the length of the seed, then a character is truncated to attempt to achieve a unique name.
- The numeric suffix may go as high as 99. If no match is achieved at this point, the COPY is aborted with a *noreplace* message ID.

Example 1

The following COPY statement resolves the name to DATA3 if the HLQ.PDS already contains the members: DATA, DATA1, DATA2, and DATA11.

```
COPY     FROM       (FILE=\appl01\data)                                  -
         TO         (FILE=HLQ.PDS(DATA) SYSOPTS="UNIQUE=YES")
```

Example 2

The following COPY statement resolves the name to DATA121 if the HLQ.PDS already contains the members: DATA, DATA1, DATA2, DATA11, DATA12, and DATA1210. This occurs because the name resolver adds one digit to the name for it to be unique.

```
COPY     FROM       (FILE=\appl01\data)                                  -
         TO         (FILE=HLQ.PDS(DATA12) SYSOPTS="UNIQUE=YES")
```

Example 3

The following COPY statement resolves the name to DATAFIL1 if the HLQ.PDS already contains the members DATAFIL. This occurs because it is impossible to append any numeric values to make the member name unique. Therefore, one trailing position is dropped.

```
COPY     FROM       (FILE=\appl01\data)                                  -
         TO         (FILE=HLQ.PDS(DATAFILE) SYSOPTS="UNIQUE=YES")
```

Example 4

The following copy statement resolves the name to DATAFI10 if the HLQ.PDS already contains the members: DATAFILE, DATAFIL1, DATAFIL2, DATAFIL3, DATAFIL4, DATAFIL5, DATAFIL6, DATFIL7, DATAFIL8, DATAFIL9. This occurs because an additional character at the end of the name is dropped to create a unique name.

```
COPY     FROM       (FILE=\appl01\data)                                  -
         TO         (FILE=HLQ.PDS(DATAFILE) SYSOPTS="UNIQUE=YES")
```

In this example, if the output PDS contains all members from DATAFIL1 to DATAFIL9 and DATAFI10 TO DATAFI99, COPY fails with the *noreplace* error.

Interfaces

The standard allocation exit is used. See the *Connect:Direct OS/390 Administration Guide*. If a unique name is successfully resolved, the new name replaces the specified name in the DDESCR of the Copy Control Block. The name may be longer than initially specified, which requires a length adjustment in the DDESCR. Space is provided for name extension when the control structure is built.

RAS

No impact to the RAS characteristics of Connect:Direct is seen by the addition of this unique name support.

Installation and Packaging

This support is provided as a sample allocation exit in the SAMPLIB. It is written in Assembler. Then Connect:Direct facilities required by AXUNIQ are shipped in the SAMPLIB distribution library.

Performance Considerations

Performance considerations relating to AXUNIQ are:

- Directory size
- Number of accesses to the same PDS concurrently

Security

AXUNIQ runs under the security of the subtask of the requester. Therefore, it participates in the same security as the Connect:Direct Process.

DBCS Support

Notification that a Process is transferring a DBCS file is done by means of the SYSOPTS statement. This statement must be included on the host node definition of the COPY statement.

By requiring the Process to notify Connect:Direct of its DBCS capability through the SYSOPTS statement, support for multiple transfers with multiple translation tables is possible. Furthermore, all Processes support compression and checkpointing.

Using SYSOPTS for DBCS Examples

The following example has a **tablename** of EBCXKSC and the default values x'0E', for **so**, and x'0F' for **si**. This table translates host EBCDIC to Korean Standard ASCII code.

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F)"
```

The following example has a **tablename** of KSCXEBC and the default values x'0E' for **so**, and x'0F' for **si**. This table translates Korean Standard ASCII code to host EBCDIC.

```
SYSOPTS="DBCS=(KSCXEBC,0E,0F)"
```

The following example has a **tablename** of EBCXKSC and the NOSO value x'00' for **so** and **si**. This translation assumes all data to be in DBCS mode.

```
SYSOPTS="DBCS=(EBCXKSC,00,00)"
```

The following example has a **tablename** of EBCXKSC and takes the defaults for **so** and **si**. Overriding the defaults for **so/si** are usually not required.

```
SYSOPTS="DBCS=(EBCXKSC)"
```

The following example has a **tablename** of USERTAB and takes the defaults for **so** and **si**. USERTAB is a user-defined, customized translation table. Check with installation standards for a reference to new or updated user-defined translate tables. See the *DBCS Support* appendix in the *Connect:Direct OS/390 Administration Guide* for instructions on how to customize your own translation tables.

```
SYSOPTS="DBCS=USERTAB"
```

The following example has a **tablename** of EBCXKCS and the default values x'0E', for **so**, and x'0F' for **si**. It shows the use of the **PAD** subparameter with its default value of x'00'.

```
SYSOPTS="DBCS=(EBCXKCS,0E,0F,PAD)"
```

The following example has a **tablename** of EBCXKCS and the default value for **so** and **si**. It shows the use of the **PAD=pc** format to specify 01 as the pad character.

```
SYSOPTS="DBCS=(EBCXKCS,PAD=01)"
```

Defining a DBCS-Capable Process

The following PC-to-host DBCS translation uses the supplied translation table KSCXEBC. Required parameters for this translation are in bold print.

```
/* PC to HOST DBCS translation using table KSCXEBC */
 PCTOHOST   PROCESS          SNODE=HOSTNODE                                  -
                             HOLD=CALL
 STEP01     COPY      TO     (PNODE                                          -
                             DSN='hlq.PCFILE'                                -
                             DISP=(RPL,CATLG)                                -
                             UNIT=SYSDA                                      -
                             DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS)  -
                             SPACE=(254,(1000,100))                          -
                             SYSOPTS="DBCS=KSCXEBC"                          -
                             )                                               -
                       FROM  (SNODE                                          -
                             DSN=PCFILE                                      -
                             TYPE=ASC2ASC                                    -
                             DISP=SHR                                        -
                             )
```

The previous example COPY statement copies a data set from a PC to a host Connect:Direct OS/390 node; its description follows:

- The copy step is named STEP01.
- The input data set is cataloged after successful completion of the Process.
- The DCB attributes specified in the TO clause of the COPY statement are used for file allocation.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the supplied translation table KSCXEBC.
- Unit has been specified on the PNODE only.
- The TYPE parameter on the FROM clause of the COPY statement must be set to ASC2ASC.

The following host-to-PC DBCS translation uses the supplied translation table EBCXKSC. Required parameters for this translation are in bold print.

```
/* HOST to PC DBCS translation using table EBCXKSC */
HOSTTOPC  PROCESS      SNODE=PCNODE                                    -
                       HOLD=CALL
STEP01    COPY    FROM (PNODE                                          -
                       DSN='hlq.HOSTFILE'                              -
                       SYSOPTS="DBCS=EBCXKSC"                          -
                       DISP=(SHR)                                      -
                       )                                               -
                  TO   (SNODE                                          -
                       DSN=PCFILE                                      -
                       TYPE=ASC2ASC                                    -
                       DISP=RPL                                        -
                       )
```

The previous example COPY statement copies a data set from a host Connect:Direct OS/390 to a PC node; its description follows:

- The copy step is named STEP01.
- The SYSOPTS attribute specified in the FROM clause of the COPY statement is used to define the default translation table EBCXKSC.
- The TYPE parameter on the TO clause of the COPY statement must be set to ASC2ASC.

The following UNIX-to-host DBCS translation uses the default translation table EBCXKSC. Required parameters for this translation are in bold print.

```
/* UNIX to HOST DBCS translation using table EBCXKSC */
 STEP01 COPY   FROM    (PNODE                                          -
                       DSN='hlq.UNIXFILE'                              -
                       SYSOPTS="DBCS=EBCXKSC"                          -
                       DISP=(SHR)                                      -
                       )                                               -
                TO     (SNODE                                          -
                       DSN='/unixfile'                                 -
                       SYSOPTS=":xlate=no:strip.blanks=no:"            -
                       DISP=RPL                                        -
                       )
```

The previous example COPY statement copies a data set from a UNIX node to a host Connect:Direct OS/390 node; its description follows:

- The copy step is named STEP01.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the default translation table EBCXKSC.
- The SYSOPTS parameter on the FROM clause of the COPY statement is required.

The following host-to-UNIX DBCS translation uses the default translation table KSCXEBC. Required parameters for this translation are in bold print.

```
/* HOST to UNIX DBCS translation using table KSCXEBC */
STEP02  COPY   TO    PNODE                                              -
                     DSN=’hlq.HOSTFILE’                                 -
                     SYSOPTS="DBCS=KSCXEBC"                             -
                     DISP=(RPL,CATLG)                                   -
                     UNIT=SYSDA                                         -
                     DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS      -
                     SPACE=(254, (1000,100))                            -
                     )                                                  -
               FROM  (SNODE                                             -
                     DSN=’/unixfile’                                    -
                     SYSOPTS=":xlate=no:strip.blanks=no:"               -
                     DISP=SHR                                           -
                     )
```

The previous example COPY statement copies a data set from a host Connect:Direct OS/390 to a UNIX node; its description follows:

- The copy step is named STEP02.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the default translation table KSCXEBC.
- The DCB attributes specified on the TO clause of the COPY statement are used for file allocation.
- Unit is specified on the PNODE.
- The SYSOPTS parameter on the FROM clause of the COPY statement is required.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

HFS Support

HFS (Hierarchical File System) is an IBM OS/390 data structure that allows UNIX programmers to develop programs that read and write data files in the OS/390 environment. Connect:Direct can read files from or write files to HFS. An HFS file is identified by a fully qualified HFS filename beginning with a ‘/’ (slash). HFS options are specified through the use of SYSOPTS parameters in the COPY statement or as a keyword. (See SYSOPTS parameters on page 41.)

Note: Connect:Direct only supports a 255-byte HFS filename including the pathname. Also, HFS filenames are case sensitive. The Process must be submitted with CASE = YES.

The following is a sample process that copies a file from an OS/390 data set to an HFS data set, and copies a file from an HFS data set to an OS/390 data set.

```
COPYHFS   PROCESS   SNODE=SC.OS390.CDA
*********************************************************************
* STEP1 – Copy from OS/390 MVS to OS/390 HFS Dataset  *
*********************************************************************
STEP1     COPY FROM(   PNODE                                    -
                       DSN=CSDOPS.USERDATA                      -
                       DISP=(SHR)                               -
                   )                                            -
                TO(    SNODE                                    -
                       DSN=/u/direct1/subdirect1/filename       -
                       DATATYPE=TEXT                            -
                       PERMISS=440                              -
                       DISP=RPL                                 -
                   )
*********************************************************************
* STEP2 – Copy from OS/390 HFS to OS/390 MVS Dataset  *
*********************************************************************
STEP2     COPY FROM(   PNODE                                   -
                       DSN=/u/direct1/subdirect1/               -
                       subdirectory/filename                    -
                       SYSOPTS=”XLATE=NO,STRIP.BLANKS=NO”       -
                       DISP=(SHR)                               -
                   )                                            -
                TO(    SNODE                                    -
                       DSN=CSDOPS.USERDATA                      -
                       DISP=RPL                                 -
                   )
```

Chapter 4

Connect:Direct OS/390 RUN JOB Statement

The Connect:Direct OS/390 RUN JOB statement allows a job to be submitted through the OS/390 internal reader, a facility that transfers jobs to the job entry subsystem (JES). Connect:Direct supports job submission only from a file existing on the node that executes the RUN JOB statement. Connect:Direct does not verify job statements. To determine the completion status of a RUN JOB statement, check your Connect:Direct statistics records.

Format

The RUN JOB statement is comprised of the parameters shown in the following table. The required parameters and keywords appear in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN JOB**	**(DSN = dsn**[(member)]**)**
		PNODE | SNODE

Field Descriptions

RUN JOB
identifies the statement with all its parameters as the RUN JOB statement.

Required Parameters

DSN = dsn[(member)]
specifies the name of the data set containing the job to be submitted. If the file is a PDS, the member containing the job must be specified. The data set containing the job must already exist on the node where the job will be submitted.

Any JCL data set used as input for RUN JOB statements cannot have an LRECL greater than 254 bytes.

Values for the DSN parameter must be in the proper case for the node where they will be processed. For example, code the value for DSN in uppercase letters when submitting a Process from UNIX that runs a job on OS/390.

Optional Parameters

PNODE
specifies that the job is to be submitted on the primary node (PNODE), which is the node with Process control. PNODE is the default value.

SNODE
specifies that the job is to be submitted on the secondary node (SNODE), which is the node that interacts with the PNODE.

Example

In this example, the RUN JOB statement named STEP1 executes job TEST in data set SRCDATA.SET on the SNODE. SRCDATA.SET(TEST) must exist as a data set on the SNODE system.

```
STEP1    RUN JOB    (DSN=SRCDATA.SET(TEST))                              -
                    SNODE
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 5

Connect:Direct OS/390 RUN TASK Statement

The Connect:Direct OS/390 RUN TASK statement allows user programs, or subtasks, to be attached during Process execution. The RUN TASK statement is valid only within a Process and is structured so that the Process waits until the subtask is finished before the next step in the Process executes.

The RUN TASK statement can be used to attach an application subtask during Process execution. A list of parameters for the subtask can be specified in the RUN TASK statement. The subtask can be attached at either node involved in Process execution.

The program must be placed in a load library that can be accessed by the Connect:Direct DTF.

You can use the RUN TASK statement to pass user parameters to the subtask. The RUN TASK statistics log records the return code, program name, and parameter list for the subtask. The log also records the dates and times for starting and completing the subtask.

Writing a Program for the RUN TASK Statement

ASMTASK is a sample program in the SAMPLIB data set supplied on the distribution tape. It can be used as a guideline for writing a program to be attached to a Process with a RUN TASK statement.

If the program requires libraries, load modules, or other files for execution, ensure that the library containing the modules is part of the STEPLIB or JOBLIB concatenation string when Connect:Direct is started. It may be possible to dynamically allocate files needed using DMRTDYN.

The programs DMNOTIFY, DMNOTFY2, DMRTAMS, DMRTSUB, DMRTAMS, DMRTPOMV, DMRTPOVM, and DMRTDYN can also be invoked by RUN TASK . See the *Connect:Direct OS/390 User's Guide* for further details concerning the use of these programs.

Format

The Connect:Direct OS/390 RUN TASK statement is comprised of the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN TASK**	**(PGM=program-name**
		PARM=(parameter [,parameter,...])
		)
		SYSOPTS="parameter [,parameter,...]"
		PNODE | SNODE
		RESTART=YES|NO

Field Descriptions

RUN TASK
identifies the statement with all its parameters as the RUN TASK statement.

Required Parameters

PGM = program-name
specifies the name of the program to be attached as the subtask. The program runs on the node specified and has access to the DD cards allocated on that node only.

Optional Parameters

PARM = (parameter [,parameter,...]) |
SYSOPTS = “parameter [,parameter,...]”
specifies the parameters to be passed to the subtask when that subtask is attached. These parameters are the actual parameters rather than a list of addresses. Null parameters can be specified by adjacent commas.

You must use SYSOPTS in addition to PGM when submitting a Process from OS/390 to run a program on UNIX. Values for the SYSOPTS parameter must be in the proper case for the node where the program is processed.

Caution: Do not use PARM on a RUN TASK Process submitted from an OS/390 to run on a Tandem system using TCP/IP or LU6.2 protocol. Use SYSOPTS in place of PARM instead.

The actual format of the parameter list that is passed to the program consists of a 2-byte field, indicating the length of the parameter followed by the parameter itself. The valid data types for the PARM parameter follow.

CLn′value′ specifies a data type of character with a length of n, where n is the number of bytes. The length is optional. If it is not specified, the actual length of the value is used. If the length specified is less than the real value, the data is truncated. If the length specified is longer than the value, the value is padded with blanks on the right. For example, CL44’FILE.NAME’.

XLn’**value**’ specifies a data type of hexadecimal with a length of n, where n is the number of bytes. The length is optional. If it is not specified, the length of the value is used. If the length specified is less than the real value, the data is truncated. If the length specified is longer than the value, the value is padded on the left with binary zeros. For example, XL8’FF00’.

H’**value**’ specifies a halfword value. No length can be specified. The value can be specified with a plus (+) or minus (-) sign. If no sign is given, plus is assumed. For example, H’-32’.

F’**value**’ specifies a fullword value. No length can be specified. The value can be specified with a plus (+) or minus (-) sign. If no sign is given, plus is assumed. For example, F’4096’.

PLn’**value**’ specifies a packed value. The length is optional; if it is not specified, the length of the value is used. If the length specified is longer than the value, the value is padded on the left with zeros. The length specifies the size of the field in bytes and cannot be longer than 16. The value can be specified with a plus (+) or minus (-) sign. If no sign is given, plus is assumed. For example, PL10’+512’.

If no data type or length is specified, the parameter is assumed to be character type and the length of the parameter is used. For example, if PARM=(’FILE.NAME’) is specified, the length used is **9**.

The parameter can also be specified as a symbolic value that is resolved when the Process is submitted. If a symbol is used, the parameter must be specified without a data type designation or length. For example, &PARM1.

When using strings that include symbolics, the strings must be enclosed in double quotes. If an ampersand (&) is to be passed as part of the parameter, then the data-type format must be used. For example, CL8’&PARM1’ uses no substitution; CL8"&PARM1" indicates that the value is substituted.

PNODE
specifies that the program will be executed on the PNODE, which is the default.

SNODE
specifies that the subtask will be attached on the secondary node (SNODE), which is the destination node. The program must exist in a load library allocated to Connect:Direct on the specified node.

RESTART=YES|NO
specifies whether or not the subtask is restarted if interrupted.

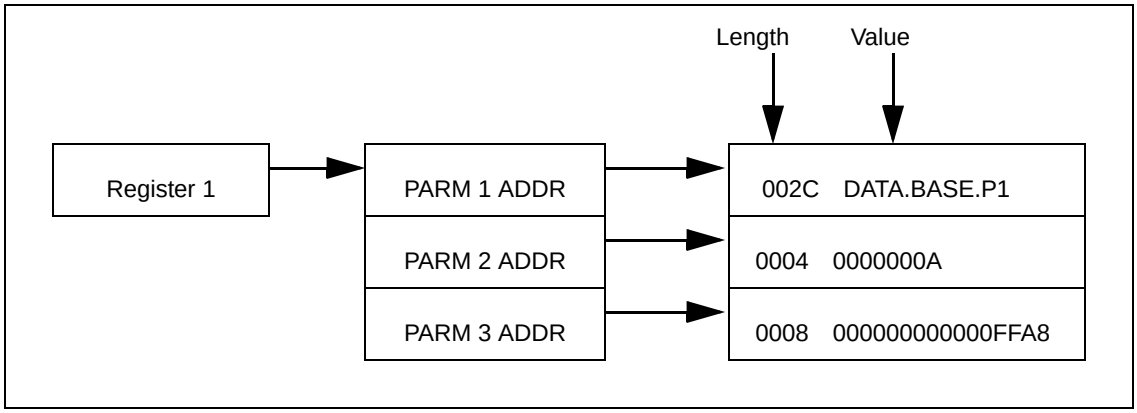
```
WAITPROC   PROCESS SNODE=NODE.NAME.HERE
WAIT1  RUN TASK (PGM=WAIT5)    RESTART=YES
               SNODE
```

Example

In this example, the RUN TASK statement runs the program named MYTASK. It is attached to the Process on the secondary node (SNODE) and is passed a list of three parameter addresses.

```
STEP1    RUN TASK   (PGM=MYTASK                                    -
                    PARM=(CL44'DATA.BASE.P1',                      -
                    F'0010', XL8'FFA8'))                           -
                    SNODE
```

The parameter passing convention for the program MYTASK is shown in the following figure. In this case, Register 1 points to a parameter list of three parameters. It would contain zero (0) if no parameters were specified. Connect:Direct sets the high-order bit in PARM 3 ADDR to indicate the end of the PARM list.



See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 6

Connect:Direct OS/390 SUBMIT Statement

During execution of a Process, the SUBMIT statement causes another Process to be submitted to either the PNODE (primary node), which is the node with Process control, or to the SNODE, which is the node that interacts with the PNODE during Process execution. The Process to be submitted must reside in a file on the node where the SUBMIT statement will execute. This node is referred to as the SUBNODE.

Note: The SUBMIT statement described in this chapter is not the same as the SUBMIT command. The SUBMIT statement is used within a Process to submit another Connect:Direct Process and parses special character strings differently than the SUBMIT command. See the *Connect:Direct OS/390 User's Guide* for SUBMIT command syntax and parameters. See *Chapter 2, Process Statement Structure and Syntax* in the *Connect:Direct Process Concepts and Examples Guide* for more information on SUBMIT special character strings parsing.

Format

The Connect:Direct OS/390 SUBMIT statement format includes the following parameters. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the SUBMIT statement format.

Label	Statement	Parameters
stepname	**SUBMIT**	**DSN=dsn[(member)]**
		DEBUG=trace bits
		NEWNAME=new-name
		SUBNODE=PNODE | SNODE
		SNODE=secondary-node-name | SNODE=TCPNAME=tcpvalue
		SNODEID=(id [,pswd] [,newpswd])

Label	Statement	Parameters
		SACCT='snode-accounting-data'
		PLEXCLASS=(pnode class, snode class)
		PNODEID=(id [,pswd] [,newpswd])
		PACCT='pnode-accounting-data'
		CASE= Yes | No
		CLASS=n
		HOLD=Yes | No | Call
		PRTY=n
		NOTIFY=%USER | userid
		REQUEUE=Yes | No
		RETAIN=Yes | No | Initial
		STARTT=([date | day][,hh:mm:ssxm])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n
		%JDATE
		%PNODE
		%SUBDATE
		%SUBTIME
		%USER

Field Descriptions

SUBMIT
identifies the statement with all its parameters as the SUBMIT statement.

Required Parameters

DSN = dsn[(member)]
specifies the name of the file that contains the Process. DSN specifies the file name and member name, if the Process resides in a PDS. If the Process is in a SAM file, only the file name should be given.

Optional Parameters

CASE = Yes | No
specifies whether parameters associated with accounting data, userid, password, and data set name in the command and in the Process are to be case sensitive. The default is NO.

CLASS = n
determines the node-to-node session on which a Process can execute. If CLASS is not specified in the Process, it defaults to the class value specified in the ADJACENT.NODE NETMAP record for the destination node (SNODE). Values range from 1-255.

DEBUG= trace settings for this Process
specifies the 8-position trace setting for this Process. This allows you to specify a trace for only this Process. The DEBUG Setting column in the following table lists acceptable trace values.

DEBUG Setting	Trace Type	Output DD
80000000	COPY Routine and RUN TASK trace	RADBDD01
10000000	Full TPCB/SYMBOLICS from DMCBSUBM	DMCBSUBM
08000000	Session manager trace	RADBDD05
04000000	Separate trace per task (Example: “R0000005” to trace TASK 5)	Rnnnnnnn
02000000	API session trace	RADBDD07
01000000	DMGCBSUB trace	RADBDD08
00800000	NETEX task termination disconnect trace	NTXTRACE
00400000	TCQSH from DMCOPYRT	DMCOPYRT
00200000	Make each SVC dump unique	N/A
00040000	GETMAIN/FREEMAIN trace	RADBDD16
00008000	I/O buffer trace	RADBDD21
00004000	WTO all dynamic allocation parameters	RADBDD22
00002000	Connect:Direct/Plex traces	
	ACTION queue manager trace	CDPLXACT

DEBUG Setting	Trace Type	Output DD
	CKPT queue manager trace	CDPLXCKP
	TCQ queue manager trace	CDPLXTCQ
	STATS queue manager trace	CDPLXSTA
	First REQUEST queue manager trace	CDPLXREQ
	Second and subsequent REQUEST queue manager trace. For example, "CDPLXR03" traces the third REQUEST queue manager. The number of REQUEST queue manager traces is based on the maximum number of servers from the asset protection (APKEY) file.	CDPLXRnn
	JOIN queue manager trace	CDPLXJOI
00001000	Workload Balancing trace	CDPLXWLB
00000080	RPL trace - long	RPLOUT
00000040	RPL trace - short	RPLOUT
00000020	Version 2 session trace	RADBDD33
00000008	Logon exit trace	RADBDD35
00000004	Logon processor trace	RADBDD36
00000002	SCIP exit trace	RADBDD37
00000001	Extended dump Information	ESTAE

HOLD = Yes | <u>No</u> | Call

specifies whether the Process is placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process. When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed in the Hold queue even if a start time is specified.

<u>No</u> specifies that the Process executes as soon as possible. HOLD=NO is the default.

Call specifies that Connect:Direct is to place the Process in the hold queue until a session is established with the specified SNODE. This session is established by either another Process starting on the PNODE destined for the same SNODE or the SNODE contacting the PNODE. For example, a Process submitted with HOLD=NO establishes a session and causes execution of any Processes residing on the SNODE destined for this node that are submitted with HOLD=CALL.

Note the following:

- Connect:Direct ignores the HOLD parameter if RETAIN=Y.

- When the SNODE is a Windows operating system, a null or ENABLE Process is required from the Windows operating system to release the held Processes. A normal send or receive of a file does not release them. This functionality enables those who dial in to send or receive files without executing held Processes until they are ready.

NEWNAME = new-name
specifies the new name to be given to the Process. The default value is the label on the PROCESS statement.

NOTIFY = %USER | userid
specifies the user to receive Process completion messages.

%USER specifies that the user on the host Connect:Direct who submitted the Process receives the completion messages. (If the Connect:Direct userid is different from the TSO userid, the user is not notified.)

userid specifies the userid on the host Connect:Direct to receive Process completion messages.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

PLEXCLASS = (pnode class, snode class)
specifies the class that directs the Process to only certain servers in a Connect:Direct/Plex. This parameter is only used in a Connect:Direct/Plex.

Each server in a Connect:Direct/Plex can be designated to support only certain PLEXCLASSes through the CDPLEX.PLEXCLASSES initialization parameter. Processes can then be limited to only those servers by specifying the PLEXCLASS in the Process definition.

The pnode class controls which Connect:Direct/Server runs the process. The snode class controls what other node is used with the Process.

The pnode class and snode class are each 1-8 characters long. An asterisk (*) indicates that the Process will run on any server with an asterisk designated in the CDPLEX.PLEXCLASSES initialization parameter. If no PLEXCLASS is specified, the network map is checked for a default PLEXCLASS. If the network map does not specify a default PLEXCLASS, then an asterisk is used as the default.

If a Process must run on a specific Connect:Direct/Server, specify the Connect:Direct/Server name in this field. The Process will only run on that server.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one used to sign on to the Connect:Direct node.

id specifies the security ID passed to a security exit (1-64 characters).

pswd specifies the current security password. This parameter can be used by the security exit to validate the current security password (1-64 alphanumeric characters). This is optional unless the user has security set to require a password.

newpswd specifies the new security password. It can be used by the security exit to change the current security password to the new security password (1-64 alphanumeric characters).

PRTY = n

specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ is a file that holds all Processes that have been submitted to Connect:Direct. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect VTAM transmission priority. The range is from 0-15. If PRTY is not specified, the default is the priority defined by the PRTYDEF keyword in the Connect:Direct OS/390 initialization parameters.

REQUEUE = Yes | No

specifies whether a COPY step should requeue if an x37 abend occurs during processing. This parameter is valid only if used when checkpointing.

Yes allows the requeued Process to be placed in the Hold queue with a status of HELD IN ERROR (HE). Corrective action can be taken and the Process restarted with the failing step; checkpointing resumes at the last successful checkpoint. Note that the Process must be explicitly released from the Hold queue when the status is HELD IN ERROR (HE).

No causes the Process to run to completion, executing subsequent steps when a COPY step fails with an abend (such as x37). The default is NO.

RETAIN = Yes | No | Initial

keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains in the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is automatically held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process will be executed every time Connect:Direct software is initialized. Processes submitted with RETAIN=INITIAL do not execute when initially submitted. STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

For Connect:Direct OS/400: This parameter is ignored when the SNODE is a Connect:Direct OS/400 node.

SNODE = secondary-node |
SNODE = TCPNAME = tcpvalue

specifies the secondary node to be used in the Process.

Note: This parameter can override the value specified in the PROCESS statement. The default value for SNODE is the value coded in the PROCESS statement. Connect:Direct allows the PNODE and SNODE to specify the same symbolic node name.

---

secondary-node-name is a 1-16 character alphanumeric name that is defined in the network map. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

Use the **TCPNAME=tcpvalue** form of the SNODE parameter to specify TCP/IP connections that are not defined in the Connect:Direct Network Map. **tcpvalue** is defined as the TCP/IP network address, the network name, or an alias for the network name. The tcpvalue can be from 1 to 16 characters with embedded periods. If the network name is longer than 16 characters, you must specify an alias for the network name.

If the TCPNAME keyword is used, the default TCP/IP port number is assumed.

For Interlink TCP/IP support: If tcpvalue specifies the name of the network, Connect:Direct will use this name to get the real address from the network name server on the TCP/IP network.

SNODEID = (id [,pswd] [,newpswd])

specifies security user IDs and passwords at the secondary node (SNODE). For Connect:Direct OS/390 and Connect:Direct UNIX, each value can be 1-64 alphanumeric characters. For other operating environments unless noted, each value can be 1-8 alphanumeric characters.

id specifies the security ID passed to the security system on the SNODE.

For Connect:Direct Tandem: This subparameter specifies the Tandem group number and user number. These numbers can range from 0-255. When specifying a Tandem value, you must use a period character as a separator between the group number and the user number.

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles may be 10 characters long.

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password. This is optional unless the user has security set to require a password.

For Connect:Direct Tandem: The IBM OS/390 node only recognizes passwords specified in uppercase alphanumeric characters. Therefore, a Process cannot be successfully initiated from Connect:Direct OS/390 with Connect:Direct Tandem unless the Connect:Direct Tandem SNODEID password follows the same convention (no lowercase characters and no control characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password.

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

For Connect:Direct OS/400: This subparameter is ignored.

STARTT = ([date | day] [,hh:mm:ssXM])

specifies that the Process will execute at a selected date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

---

date specifies that the Process is to be held until a specific date. You can specify the month (mm), day (dd), and year (yy) in one of the following two formats:

- mm/dd/yy
- mm.dd.yy

Note: You must use periods or backslashes (/) to separate the components of a date value to guarantee transfers between all platforms.

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

---

day specifies the day of the week the Process will be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value can be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday with Monday as the only STARTT parameter, the Process does not run until the following Monday.

You can also specify TODAY, which releases the Process for execution today, or TOMORROW, which releases the Process for execution the next day.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT, which releases the Process for execution at midnight.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. A Process submitted with HOLD=YES is placed in the Hold queue even if a start time has been specified.

SUBNODE = PNODE | SNODE

specifies the node where the Process defined in this SUBMIT statement will execute. Specifying PNODE means that the Process is submitted on the node that has Process control. Specifying SNODE means that the Process is submitted on the node participating in, but not controlling, Process execution. In both cases, the Process must reside on the node where it is being submitted. The default is PNODE.

For Connect:Direct OS/400: SUBNODE=SNODE is not valid if communicating with a node running Connect:Direct OS/400.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden when submitting the Process.

A symbolic parameter containing special characters must be enclosed in single quotation marks.

Note that the symbolic parameter for the SUBMIT statement must begin with a single ampersand. This allows submission of the Process that contains the SUBMIT statement to resolve symbolic parameters correctly.

An ampersand symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

%JDATE

used to specify the date the process was submitted in Julian format. The variable is resolved as the submission date of the process in the format yyyyddd. Among other uses, the value returned is suitable for constructing a file name on the node receiving

the file. When used for this purpose, the %JDATE value must be preceded by an alphabetic character.

Note: The value of the variable is resolved at process submit time. The value will correspond to the date on which the process was submitted, regardless of when or how many times the process is actually executed.

%PNODE = pnode
is an intrinsic symbolic variable that resolves to the PNODE of the Process.

%SUBDATE
used to specify the date the process was submitted in Gregorian format. The variable is resolved as the submission date of the process in the format cyymmdd where c is the century indicator and is set to 0 for 20th-century date values or 1 for 21st-century values. Among other uses, the value returned is suitable for constructing a file name on the node receiving the file. When used for this purpose, the %SUBDATE value must be preceded by an alphabetic character.

Note: The value of the variable is resolved at process submit time. The value will correspond to the date on which the process was submitted, regardless of when or how many times the process is actually executed.

%SUBTIME
used to specify the time the process was submitted. The variable is resolved as the submission time of the process in the format hhmmss. Among other uses, the value returned is suitable for constructing a file name on the node receiving the file. The %SUBTIME value must be preceded by an alphabetic character.

Note: The value of the variable is resolved at process submit time. The value will correspond to the time at which the process was submitted, regardless of when or how many times the process is actually executed.

%USER = userid
is an intrinsic symbolic variable that resolves to the user submitting the Process.

Example

This example shows using SUBMIT with the DSN parameter. Symbolics for PROCESS2 are supplied by the operator submitting PROCESS1.

The operator at the PNODE issues the following Connect:Direct SUBMIT PROCESS command to initiate the file transfer:

```
SUB       PROC=PROCESS1                                                 -
          &DSN1=A345.DATA                                               -
          &DSN2=A345.NEW.DATA
```

PROCESS1 executes:

```
PROCESS1  PROCESS    PNODE=CD.LA  SNODE=CD.DALLAS                    -
                     &DSN1=&DSN1                                     -
                     &DSN2=&DSN2                                     -
                     &PRTY=14                                        -
                     NOTIFY=A345
COPYSTEP  COPY   FROM (DSN=&DSN1 DISP=SHR PNODE)                     -
                  TO   (DSN=&DSN2 DISP=SHR SNODE)
SUBSTEP   SUBMIT     DSN=A345.PROCESS.LIB(PROCESS2)                  -
                     PRTY=&PRTY                                      -
                     SUBNODE=SNODE                                   -
                     &DSN=&DSN2
```

PROCESS1 submits PROCESS2:

```
PROCESS2  PROCESS    PNODE=CD.DALLAS                                 -
                     SNODE=CD.NEWYORK
COPYSTEP  COPY  FROM  (DSN=&DSN2 PNODE)                              -
                TO    (DSN=A345.NEW.DATA1 DISP=SHR)
```

- PROCESS1 copies the file A345.DATA in LA to a file called A345.NEW.DATA in DALLAS. It then submits PROCESS2, which executes on the DALLAS node. PROCESS2 is submitted with a PRTY of 14.
- PROCESS2 copies the file A345.NEW.DATA in Dallas to the file A345.NEW.DATA1 in New York.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 7

Connect:Direct OS/390 SYMBOL Statement

The Connect:Direct OS/390 SYMBOL statement allows you to build symbolic substitution values.

Format

The Connect:Direct OS/390 SYMBOL statement format includes the following parameters. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**SYMBOL**	**&symbolic_name=variable-string**

Field Descriptions

SYMBOL
identifies the statement with all its parameters as the SYMBOL statement.

Required Parameters

&symbolic_name = variable string
specifies the string that is substituted into the Process.

When Connect:Direct encounters an ampersand (&) plus 1-17 alphanumeric characters, it substitutes the variable string assigned to that symbolic.

Symbols in the string are resolved from previously specified values in a PROCESS, SUBMIT, or SYMBOL statement. With the SYMBOL statement, different pieces of a Connect:Direct statement string can be concatenated, allowing the user to move data in a variety of ways. See the following example for more information.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Examples

In the following example Process, PROCSYM, the SYMBOL statement causes the file Z813103.ABC.DEF to be copied into the PDS Z813103.TEST.XXX.

Note that two vertical bars preceded and followed by blank spaces (||) are used to indicate concatenation. Bracketing backslashes ensure that special characters within the string are maintained. The COPY statement is resolved by the symbolic substitution of the &AA symbol to Z813103.ABC.DEF.

```
PROCSYM   PROCESS    HOLD=YES                                          -
                     SNODE=CD.NEWYORK                                  -
                     SNODEID=(RSMITH,ROGER)                            -
                     PRTY=12
          SYMBOL     &A1=ABC
          SYMBOL     &A2=DEF
          SYMBOL     &AA=\Z813103.\ || &A1 \.\ || &A2
STEP1     COPY  FROM (DSN=&AA)                                          -
                TO   (DSN=Z813103.TEST.XXX                              -
                     DISP=OLD)
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 8

Connect:Direct OS/390 Conditional Statements

Conditional statements permit you to alter the sequence of Connect:Direct Process execution based on the completion of the previous step in the Process.

Formats

Formats for Connect:Direct OS/390 conditional statements follow. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**IF**	**(label condition nn) THEN**
		(process steps)
	ELSE	
		(alternative process steps)
	EIF	
optional	GOTO	label
	EXIT	

Statement Descriptions

IF THEN

specifies that Connect:Direct executes a block of statements based on the completion code of a Process step. The EIF statement must be used in conjunction with an IF THEN statement. A return code with the high order bit on is evaluated as a negative return code. See *Field Descriptions* on page 76 for details.

ELSE
designates a block of Connect:Direct statements that execute when the IF THEN condition is not satisfied. No parameters exist.

EIF
is required for specifying the end of the IF THEN or IF THEN ELSE block of statements. No parameters exist.

GOTO
moves to a specific step within a Process. See *Field Descriptions* on page 76 for details.

EXIT
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

label
For the **IF THEN statement**, the label specifies the name of a previous step whose completion code is used for comparison.

For the **GOTO statement**, the label specifies the name of a subsequent step in a Process (required for GOTO only). The name cannot be the label of a preceding step nor can it be the label of the GOTO statement of which it is a part.

Note: User-defined labels must begin in column one. The label consists of a 1-8 character alphanumeric string; the first character must be alphabetic.

condition
specifies the type of comparison to be performed. This condition checking can be based on comparisons for equality, inequality, greater than, less than, greater than or equal to, and less than or equal to.

The completion code from RUN JOB is for the job submission only. It is not the completion code of the job submitted.

Valid symbols, alternate symbols, and conditions follow:

= or EQ specifies that the completion code must be equal to the value *nn* for the condition to be satisfied.

<> or ¬= or NE specifies that the completion code must not equal the value *nn* for the condition to be satisfied.

>= or ¬< or GE specifies that the completion code must be greater than or equal to the value *nn* for the condition to be satisfied.

> or GT specifies that the completion code must be greater than the value *nn* for the condition to be satisfied.

<= or ¬> or LE specifies that the completion code must be less than or equal to the value *nn* for the condition to be satisfied.

< or LT specifies that the completion code must be less than the value *nn* for the condition to be satisfied.

nn

specifies the numeric value to be used for comparison. If coded as X‘nn’, it is a hexadecimal value; any other coding is treated as a decimal.

Typically, if a completion code less than 4 is returned, the Process completed successfully. In most cases, a return code greater than 4 indicates the Process ended in error. A return code of 4 indicates a warning.

THEN

specifies subsequent processing to be performed if the condition specified is true.

Example

The following is an example of a Process containing all of the conditional statements. A description of each step follows the example Process.

```
COPY01   PROCESS    SNODE=CD.CHICAGO
STEP01   COPY  FROM (DSN=ABC.FILEA PNODE)                              -
               TO   (DSN=JKL.FILEA)
STEP02   IF         (STEP01 GT 4) THEN
                         GOTO STEP07
         ELSE
STEP03   RUN JOB    (DSN=USERJOB) SNODE
         EIF
STEP04   IF         (STEP03 >= 8) THEN
                         EXIT
         EIF
STEP05   IF         (STEP03 LT 4) THEN
STEP06   COPY  FROM (DSN=ABC SNODE)                                   -
               TO   (DSN=MNO PNODE)
         EIF
         EXIT
STEP07   RUN TASK   (PGM=DMNOTIFY,                                    -
                    PARM=('FAIL',ABC.FILEA))                          -
                    PNODE
```

COPY01 is the PROCESS statement defining the secondary node as CD.CHICAGO.

STEP01 copies file ABC.FILEA on the PNODE to file JKL.FILEA on the SNODE.

STEP02 checks the completion code of STEP01. If STEP01 fails, STEP07 executes. If STEP01 ended with a completion code of 4 or less, STEP03 executes.

STEP03 submits the job, USERJOB, on the SNODE.

STEP04 checks the completion code of STEP03. If STEP03 fails with a completion code of 8 or greater, the Process terminates. Otherwise, STEP05 executes.

STEP05 checks the completion code from STEP03. If less than 4, indicating the step completed without errors, the COPY statement in STEP06 executes and the Process terminates.

STEP06 copies file ABC on the SNODE to file MNO on the PNODE.

STEP07 only executes if STEP01 fails. The program DMNOTIFY runs, sending an OPERATION FAILED message to the console operator.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 9

Connect:Direct VM/ESA PROCESS Statement

A Connect:Direct VM/ESA PROCESS statement defines the attributes of a Connect:Direct Process and is always the first statement in a Process.

Note: The maximum storage area allowed for a Connect:Direct VM/ESA Process statement is 64K. To accommodate a larger Process, split the Process into two separate Processes. Include a SUBMIT statement in the first Process to run the second Process.

Format

The Connect:Direct VM/ESA PROCESS statement format follows. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the PROCESS statement format.

Label	Statement	Parameters
process name	**PROC**ess	**SNODE=secondary-node-name**
		SACCT='snode-accounting-data'
		PNODE=primary-node-name
		PNODEID=(id [,pswd] [,newpswd])
		SNODEID=(id [,pswd] [,newpswd])
		PACCT='pnode-accounting-data'
		CLASS=n
		MAXDELAY=[UNLIMITED | QUEUED | 0 | hh:mm:ss]
		HOLD=Yes | No | Call
		PRTY=n
		NOTIFY=%USER | userid

Label	Statement	Parameters
		REQUEUE=Yes | No
		RETAIN=Yes | No | Initial
		STARTT=([date | day][,hh:mm:ssXM])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n =variable-string-n

Field Descriptions

process name
specifies the name of the Process. The Process name can be from 1-8 characters long. The first character must be alphabetic. The Process name must start in column one. This label is used to identify the Process in any messages or statistics relating to this Process.

PROCess
identifies the statement with all its parameters as the PROCESS statement. This statement identifier can be abbreviated to PROC.

Required Parameters

SNODE = secondary-node-name
is a 1-16 character alphanumeric name that specifies the secondary node (SNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

This is the logical node name that has been defined in the ADJACENT.NODE entry for that node in the network map.

Note: This parameter is not required if it is specified on the SUBMIT command.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. If CLASS is not specified, the Process uses the class value specified in the ADJACENT.NODE NETMAP record for the destination node (SNODE). Values range from 1-255.

HOLD = Yes | No | Call

specifies whether the Process is to be placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process.

No specifies that the Process is to execute as soon as possible. HOLD=NO is the default.

Call specifies that the Process is placed in the Hold queue until a VTAM session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

MAXDELAY = [UNLIMITED | QUEUED | 0 | hh:mm:ss]

indicates that the **submit** command waits until the submitted Process completes execution or the specified time interval expires.

unlimited specifies the **submit** command to wait for the Process to complete execution. UNLIMITED is the default if MAXDELAY is coded without any parameters.

queued specifies the **submit** command waits until the process completes or 30 minutes, whichever occurs first.

0 specifies the **submit** command will attempt to start a session for the submitted Process to execute on immediately. The submit command waits until the Process completes or until all timer retries have been exhausted.

hh:mm:ss specifies that the **submit** command waits for an interval no longer than the specified hours, minutes, and seconds or until the Process completes, whichever occurs first.

NOTIFY = %USER | userid

specifies the user to receive Process completion messages.

%USER specifies that the user who submitted the Process receives the completion messages if the Connect:Direct userid is the same as the VM id. If the Connect:Direct userid is different from the VM userid, the user is not notified.

userid specifies the VM userid to receive Process completion messages.

PACCT = 'pnode-accounting-data'

specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

PNODE = primary-node-name

is a 1-16 character alphanumeric name that specifies the primary node (PNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

The node to which the Process is submitted is always the PNODE. This parameter defaults to the name of the node submitting the Process and need not be specified. It is used for documentation purposes only.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one you used to sign on to Connect:Direct.

id specifies the security ID passed to the security system at the PNODE (1-8 alphanumeric characters).

pswd specifies the current security password for the specified ID. This parameter can be used by the security system at the PNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

newpswd specifies the new security password. It can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

PRTY = n
specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ holds all Processes that have been submitted to Connect:Direct. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect VTAM transmission priority. The range is from 0-15. If PRTY is not specified, the default is the priority defined by the PRTYDEF keyword in the Connect:Direct initialization parameters. See the *Connect:Direct VM/ESA Administration Guide* for more information.

REQUEUE = Yes | No
specifies whether a COPY step should requeue if an x37 abend occurs during processing. This parameter is valid only if used when checkpointing.

Yes allows the requeued Process to be placed in the Hold queue with a status of HELD IN ERROR (HE). Corrective action can be taken and the Process restarted with the failing step. Checkpointing resumes at the last successful checkpoint. Note that the Process must be explicitly released from the Hold queue when the status is HELD IN ERROR (HE).

No causes the Process to run to completion, executing subsequent steps when a COPY step fails with an abend (such as x37). The default is NO.

RETAIN = Yes | No | Initial
keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains on the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval. However, a date is invalid as a STARTT subparameter when used in conjunction with RETAIN.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time Connect:Direct is initialized. The Process will not execute when initially submitted. Note that STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

For Connect:Direct OS/400: This parameter is ignored when the SNODE is a Connect:Direct OS/400 node.

SNODEID = (id [,pswd] [,newpswd])

specifies security user IDs and passwords at the secondary node (SNODE).

For Connect:Direct OS/400: If an SNODEID and password of the Process submitter is not coded in the PROCESS statement, the user ID and password of the Process submitter will be used for the security ID and password check by Connect:Direct OS/400.

id specifies the security ID passed to the security system on the SNODE (1-8 alphanumeric characters).

For Connect:Direct Tandem: This subparameter specifies the Tandem group number and user number. These numbers can range from 0-255. When specifying a Tandem value, you must use a period character as a separator between the group number and the user number.

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles may be 10 characters long.

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

For Connect:Direct Tandem: The VM node only recognizes passwords specified in uppercase alphanumeric characters. Therefore, a Process cannot be successfully initiated from Connect:Direct VM/ESA with Connect:Direct Tandem unless the Connect:Direct Tandem SNODEID password follows the same convention (no lowercase characters and no control characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

For Connect:Direct OS/400: This subparameter is ignored.

STARTT = ([date | day][,hh:mm:ssXM])

specifies that the Process will be executed at a selected date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

---

date specifies that the Process is to be held until a specific date. You can specify the day (dd), month (mm), and year (yy for 2-digit year and yyyy for 4-digit year).

You can use periods or slashes (/) to separate the components of a date value.

Note: You can omit the separators only for transfers between mainframe nodes. Sterling Commerce recommends the use of separators to guarantee transfers between all platforms.

To specify the order of a Gregorian day, month, and year, you *must* define the DATEFORM initialization parameter. If you do not specify the DATEFORM parameter, Connect:Direct VM/ESA defaults to the DATEFORM=MDY formats described in this section.

After you designate the date order in your initialization parameters, you can use the following date formats:

DATEFORM=MDY specifies the date format as:

- mm/dd/yy *or* mm/dd/yyyy
- mm.dd.yy *or* mm.dd.yyyy

DATEFORM=DMY specifies the date format as:

- dd/mm/yy *or* dd/mm/yyyy
- dd.mm.yy *or* dd.mm.yyyy

DATEFORM=YMD specifies the date format as:

- yy/mm/dd *or* yyyy/mm/dd
- yy.mm.dd *or* yyyy.mm.dd

DATEFORM=YDM specifies the date format as:

- yy/dd/mm *or* yyyy/dd/mm
- yy.dd.mm *or* yyyy.dd.mm

The following Julian date formats are valid:

- yyddd *or* yyyyddd
- yy/ddd *or* yyyy/ddd
- yy.ddd *or* yyyy.ddd

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

day specifies the day of the week the Process is to be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value can be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday, with Monday as the only STARTT parameter, the Process will not run until the following Monday.

You can also specify TODAY, which releases the Process for execution the day and time of Process submission (unless the time of day is specified), or TOMORROW, which releases the Process for execution the next day. If a time of day is not specified with TOMORROW, the Process executes after midnight.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT to release the Process for execution at midnight.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time is specified.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden in the SUBMIT command.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Example

The following is a example PROCESS statement; its description follows:

```
PROC1     PROCESS     SNODE=CD.NODE.A                                  -
                      SNODEID=(JONES,OPENUP)                           -
                      CLASS=4                                          -
                      HOLD=YES                                         -
                      NOTIFY=%USER                                     -
                      PACCT='OPERATIONS, DEPT. 87'                     -
                      RETAIN=NO
```

The Process named PROC1 specifies a secondary node (SNODE) of CD.NODE.A. The corresponding security userids and passwords (SNODEID) are included.

This Process will run in CLASS 4.

The Process is placed in the Hold queue until it is released for execution with a CHANGE PROCESS command. As indicated by the NOTIFY parameter, the VM user who submitted the Process is notified upon completion of the Process.

The PACCT parameter specifies that all accounting information will be attributed to the operations account, department 87, if the node has a program that maintains this information.

Once the Process executes, it is deleted because the RETAIN parameter is set to NO. Note that NO is the default value for the RETAIN parameter.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 10

Connect:Direct VM/ESA COPY Statement

The Connect:Direct VM/ESA COPY statement allows you to copy CMS (Conversational Monitor System) files, a set of CMS files, a Group of files, and VSAM files between nodes.

Format

The COPY statement consists of a FROM parameter that includes the source file name and a TO parameter that includes the destination file name. Additional parameters and subparameters can be specified to further customize the file transfer operation.

The following pages provide the format of the COPY statement. All platform-specific parameters and subparameters are listed along with their possible values. Following the COPY statement format is a definition of each parameter and subparameter.

To copy from one Connect:Direct system environment to another, refer to the appropriate COPY FROM and TO sections for those environments. For example, if the source file is located on Connect:Direct Tandem, refer to the COPY FROM information for Connect:Direct Tandem. If the destination for the file is Connect:Direct VM/ESA, refer to the TO information in this chapter.

Label	Statement	Parameters	
optional	**COPY**	**FROM**	**(**
			DSN='filename filetype' | GROUP='* * ... *'
			LINK=(vmid,pwd,accmode,ccuu)
			VSAMCAT=(dsn,vmid,pwd,accmode,ccuu)
			PNODE | SNODE
			TYPE=typekey

Label	Statement	Parameters	
			DCB=([BLKSIZE=no. bytes ,DEN=[0 | 1 | 2 | 3 | 4] ,DSORG=[PS | VSAM] ,LRECL=no. bytes ,RECFM=record format ,TRTCH=[C | E | T | ET]])
			DISP=([OLD | SHR]
			LABEL=([file sequence number], [SL | NL] ,[RETPD=nnnn | EXPDT=yyddd])
			UNIT=(3480|TAPE)
			VOL=(,,[volume sequence number] ,[volume count] ,SER=volume serial number |(list))
			EXCLUDE=[(] generic | member | (startrange/stoprange)| list[)]
			REPLACE | NOREPLACE
			SELECT=[(]member | generic|(*)| (member,[newname],[NR|R]) | (generic,,[NR |R]) (startrange/stoprange,, [NR|R])| list[)]
			SFSDIR = (’dirid’,[CRE | NOCRE])
			)
		TO	**(**
			DSN=’filename filetype’ | DSN=’!SPOOL vmid fn ft’ | GROUP=’%1% ... %N%’
			LINK=(vmid,pwd,accmode,ccuu)
			VSAMCAT=(dsn,vmid,pwd,accmode,ccuu)
			PNODE | SNODE
			SFSDIR = (’dirid’,[CRE | NOCRE])
			TYPE=typekey
			PROTECT=Yes|No

Label	Statement	Parameters	
			DCB=([BLKSIZE=no. bytes ,DEN=[0 | 1 | 2 | 3 | 4] ,DSORG=[PS | VSAM] ,LRECL=no. bytes ,RECFM=record format ,TRTCH=[C | E | T | ET]])
			DISP=[NEW | OLD | RPL | SHR | MOD]
			LABEL=([file sequence number] ,[SL | NL] ,[RETPD=nnnn | EXPDT=yyddd])
			UNIT=(3480|TAPE)
			VOL=(,,[volume sequence number] ,[volume count] ,SER=volume serial number | (list))
			)
			SYSOPTS='!SPOOL [CLASS x] [DIST distcode | * | OFF] [FORM form | OFF]'
			CKPT=[nK|nM]
			COMPRESS [PRIMEchar=X'40' | X'xx' | C'c' | EXTended]
			OLDDATE

Field Descriptions

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent subparameters define the source file characteristics.

(FROM) DSN = 'filename filetype' | GROUP = '* * ... *'
specifies the source filename and filetype, or VSAM filename when used with the FROM parameter. The filename and filetype are verified based on the VM standard

filename conventions. If the copy is from a Connect:Direct VM/ESA node or if the Connect:Direct OS/390 PDS member name contains special characters, the filename must be enclosed in single quotation marks.

TO
specifies that the subsequent subparameters define the destination file characteristics.

(TO) DSN = 'filename filetype'|DSN = '!SPOOL vmid fn ft' | GROUP = '%1% ... %N%'
specifies the destination filename and filetype, or VSAM filename, when used with the TO parameter. The filename must be enclosed in single quotation marks to allow for special characters if the copy is to a Connect:Direct VM/ESA node.

Connect:Direct VM/ESA can be used to spool files to a specific virtual reader. This is useful for downloading files, because Connect:Direct cannot gain write access to a primary user minidisk. The spooled data is in Netdata format (the same as files sent with the CMS SENDFILE command).

To send a file to the virtual reader, use the following format, where vmid is the machine ID of the virtual reader at the destination, and fn and ft are the filename and filetype of the spooled file:

```
DSN='!SPOOL  vmid  fn  ft'
```

To spool a set of files on a minidisk with the same filetype, enter the following, where vmid is the machine ID of the virtual reader at the destination, and ft is the filetype of the file(s) to be spooled:

```
DSN='!SPOOL  vmid  *  ft'
```

Note: If you are using !SPOOL and your keyboard does not interpret ! as X'5A', use the key that translates to 5A.

LINK = (userid,password,mode,ccuu)
specifies the disk where the CMS file is located. This parameter allows the user to access the CMS file.

Note: You cannot specify the SFSDIR and the LINK parameter for the same file.

userid specifies the owner ID for the CMS minidisk where the file is located. The valid length ranges from 1-8 characters.

password specifies the appropriate password for the CMS minidisk where the file is located. The maximum length is 256 characters. The default password is ALL.

mode specifies the link access mode.

When used with the FROM parameter, the access modes are W (primary read/write access), M (primary multiple access), R (primary read only), RR (primary and secondary read only access), WR (primary read/write access; alternate read only access), MR (primary multiple access; alternate read only access), and MW (primary multiple access; alternate read/write only access).

When used with the TO parameter, the access modes are W, M, MW, WR, and MR.

Caution: MW access to CMS format disks can be destructive. You must be able to guarantee that no other VM user, or Connect:Direct Process, has MW, M, or W access to the minidisk. If multiple users or Processes get write access to the disk at the same time, there is a high probability that the CMS directory on the disk will be destroyed. The most likely result is a message from Group Control System (GCS) or an equivalent message from CMS. The GCS message will indicate that a CSIFNS420T file system error was detected. When GCS issues the CSIFNS420T message, GCS terminates all processing.

ccuu specifies the virtual address of the disk where the CMS file is located. Any four-digit number is valid.

Optional Parameters

CKPT = [nK|nM]

specifies the byte interval for checkpoint support, which allows restart of interrupted transmissions at the last valid transmission point and reduces restart time. K denotes thousands, and M denotes millions. Connect:Direct software converts the value to a block boundary, and a data transmission checkpoint is taken at that position. Sequential files can be checkpointed. A checkpoint value of zero (0) stops automatic processing.

COMPRESS [PRIMEchar=X'40'|X'xx'|C'c'] | [EXTended]

specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at the destination.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X'40').

If compression is specified, Connect:Direct software reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-64) of the primary compression character are compressed to one byte.
- Repetitive occurrences (ranging from 3-64) of any other character are compressed to two bytes.

EXTended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited, CPU is available, and data is repetitive.

The following are valid options for EXTended:

- **CMPrlevel** determines the compression level. The valid value range is 1-9. Level **1** is the fastest compression, but it offers the lowest degree of compression. A higher compression level produces a higher quality of compression, but the higher level has the slowest rate of compression. The default is 1.

- **WINdowsize** determines the size of the compression window or history buffer. This memory is above the line. The valid values are 8-15. Higher windowsize specifications increase the degree of compression and use more virtual memory. Size **8** uses **1** KB of memory where Size 15 requires 128 KB of memory. The default is 13.
- **MEMlevel** identifies how much virtual memory is allocated to maintain the internal compression state. This memory is above the line memory. The valid value range is 1-9. Level **1** requires the least memory (1K), but it reduces the degree of compression. Level **9** provides the fastest speed, but it uses the most memory (256K). The default is 4.

The following example shows one way to specify the various EXTended options in a COPY statement:

```
COMPRESS EXT = ( CMP=4
                 WIN=12
                 MEM=8
                )
```

DCB =([BLKSIZE=no. bytes
,DEN=[0|1|2|3|4]
,DSORG=[PS|VSAM]
,LRECL=no. bytes
,RECFM=record format
,TRTCH=[C|E|T|ET] [COMP | XF | NOCOMP | NF]])

specifies attributes to be used in the allocation of the files for the source and destination files. For destination files, these parameters override the DCB information provided in the source file at open time. For SAM-to-SAM copies where the destination file is new and the DCB parameter is not coded with the TO parameter, the DCB BLKSIZE, DEN, DSORG, LRECL, RECFM, and TRTCH are taken from the source file or from the TYPE defaults file if the TYPE keyword is specified.

BLKSIZE specifies the length in bytes of the block. The minimum length is 18 bytes, and the maximum length is 32,760 bytes. For BLKSIZE greater than 32760, you must use the DMGIOX64 exit described in the *Using Connect:Direct Exits* chapter of the *Connect:Direct VM/ESA Administration Guide*.

DEN specifies the magnetic tape mode setting. The values for the DEN parameter for 7- and 9-track tape are shown in the following table. When coded together, the DEN and TRTCH values are used to select a tape device for allocation by Connect:Direct VM/ESA.

DEN	7-Track Tape	9-Track Tape
0	200 bpi	-
1	556 bpi	-
2	800 bpi	800 bpi (NRZI) NRZI is Non-Return-to-Zero Inverted recording mode

DEN	7-Track Tape	9-Track Tape
3	-	1600 bpi (PE) PE is Phase Encoded recording mode
4	-	6250 bpi (GCR) GCR is Group Coded recording mode

DSORG specifies file organization. Supported file organizations are Physical Sequential (PS) and VSAM.

LRECL specifies the length in bytes of the record.

Note: When RECFM=V or RECFM=VB type files are used, the LRECL value must be at least the size of the largest record in the file plus 4 bytes. If RECFM=V, the BLKSIZE value must be at least the LRECL value plus another 4 bytes. If RECFM=V, the BLKSIZE value does not need to be an even multiple of LRECL.

RECFM specifies the format of the records in the file. Either F (Fixed), FB, V (Variable), or VB can be specified.

TRTCH specifies the magnetic tape mode setting. When coded together, the TRTCH and DEN values are used to select a tape device for allocation by Connect:Direct VM/ESA. Valid options include the following:

- **C** specifies data conversion, odd parity, and no translation.
- **E** specifies no data conversion, even parity, and no translation.
- **T** specifies no data conversion, odd parity, and BCD or EBCDIC translation.
- **ET** specifies no data conversion, even parity, and BCD or EBCDIC translation.
- **COMP | XF** specifies the tape to create in compressed format. You can specify COMP or XF.
- **NOCOMP | NF** specifies the tape to create without compression. You can specify NOCOMP or NF.

(FROM) DISP = ([OLD | SHR])

specifies the status of the file and what is to be done with the file after notification of successful transmission. Subparameters are as follows:

- **OLD** specifies that the source file existed before the Process began executing and the Process is given exclusive control of the file.
- **SHR** specifies that the source file existed before the Process began executing and the file can be used simultaneously by another job or Process. The default is SHR.

(TO) DISP = [NEW|OLD|RPL|SHR|MOD]

specifies the status of the data on the receiving node. Only the OLD and RPL dispositions apply to VSAM files. Options for this subparameter are as follows:

NEW specifies that the Process step will create the destination file. NEW applies to SAM files only. NEW is the default.

OLD specifies that the destination file existed before the Process began executing. If DISP=OLD, the destination file can be a VSAM file or a SAM file.

RPL specifies that the destination file will replace any existing file or allocate a new file. DISP=RPL can be specified for SAM files only.

SHR specifies that the destination file existed before the Process began executing and that the file can be used simultaneously by another job or Process.

MOD specifies that the Process step will modify the SAM file by appending data at the end of an existing file. If a file does not exist, a new file is allocated

EXCLUDE = [(] generic | member | (startrange/stoprange)| list[)]

specifies criteria by which specific files in a set of CMS files will not be copied. EXCLUDE can be specified only with the FROM parameter. EXCLUDE allows you to make exceptions to files specified generically or by range in the SELECT option. See page 99 for the override priority for the SELECT and EXCLUDE parameters.

generic specifies a generic filename. For example, if CDV* is specified, all filenames beginning with CDV and having the specified filetype are excluded.

The only way to override an excluded generic is to specify an individual filename in the SELECT parameter.

member specifies an individual filename. When a file is specified in the EXCLUDE parameter, its exclusion cannot be overridden.

startrange specifies the first name in an alphanumeric range of files. Although range filenames are treated as generics, they cannot be used with an asterisk. A slash (/) separates the first and last (stoprange) filenames. When used with the EXCLUDE statement, the first and last files specified in the range as well as all files between are not copied.

stoprange specifies the last name in an alphanumeric range of files. Although range filenames are treated as generics, they cannot be used with an asterisk. A slash (/) separates the first (startrange) and last filenames. When used with the EXCLUDE statement, the first and last files specified in the range as well as all files between are not copied.

The only way to override an excluded range is to specify an individual filename in the SELECT parameter.

LABEL =([file sequence number]
,[SL|NL]
,[RETPD=nnnn|EXPDT=yyddd])

specifies label information for the tape.

file sequence number specifies the relative file position on the tape.

SL specifies IBM standard labels.

NL specifies no labels.

RETPD specifies the retention period.

EXPDT specifies the expiration date.

NOREPLACE
specifies that files of a sending set of files will not replace existing files of the same name at the receiving set of files.

NOREPLACE takes effect only when the FROM and TO files are sets of files. The default is REPLACE. NOREPLACE applies to an entire set of files as opposed to the NR option of the SELECT parameter, which applies to files within a set of files.

OLDDATE
specifies that the creation or last modified date and the time of the file being transmitted will be used to set the creation date and time of the file received.

If OLDDATE is not specified in the COPY statement, the current date and time are used for the creation date and time of the received file.

The OLDDATE parameter is used for sequential file transfers between two Connect:Direct VM/ESA systems, and transfers between a set of CMS files on Connect:Direct VM/ESA to partitioned data sets (PDSs) on Connect:Direct OS/390 systems.

PNODE
specifies the primary node. When PNODE is coded with the FROM parameter, the file to be copied resides on the primary node. When PNODE is coded with the TO parameter, the file is sent to the primary node. PNODE is the default with the FROM parameter.

PROTECT = Yes|<u>No</u>
specifies whether an IBM RACF profile will be created for a new file. The PROTECT parameter is valid only in the TO clause of the COPY statement.

Yes specifies that a RACF profile will be created for a newly transferred file.

<u>No</u> specifies that a RACF profile will not be created. No is the default.

REPLACE
specifies that the sending set of files replaces files of the same name at the receiving set of files. REPLACE is the default.

**SELECT = [(]member|generic|(*) | (member,[newname]
,[NR|R])|(generic,,[NR|R]) (startrange/stoprange
,, [NR|R])| list [)]**
specifies selection criteria by which files in a set of CMS files are to be copied. SELECT can be specified only with the FROM parameter. See page 99 for the override priority for the SELECT and EXCLUDE parameters.

generic specifies a generic filename. If CDV* is specified as either a parameter or as a subparameter, all filenames beginning with CDV and having the specified filetype are selected to be copied.

(*****) represents a global generic. A global generic indicates that all files in the set of files are to be included. A global generic is valid only with the SELECT keyword.

When a generic is specified in the SELECT parameter, its selection can be overridden with any type of specification in the EXCLUDE parameter.

When using a generic and specifying **NR** or **R**, the second positional parameter (newname) must be null.

member specifies an individual filename. Specifying a filename in the DSN is the same as specifying a SELECT statement with only that file.

The only way to override a selection by filename is to specify that filename in the EXCLUDE parameter.

newname specifies a new name for a file. The newname parameter must be null if a generic name or range is used in the first subparameter position.

NR specifies that a file will not replace an existing file of the same name at the receiving set of files. **NR** overrides the REPLACE option. **R** is the default.

When used with newname, NR applies to the newname and not to the original filename. When used with a generic name or with a range, **NR** applies to all files selected for that criterion. **NR** applies to files within a set of files, as opposed to NOREPLACE, which applies to the set of files as a whole.

R specifies that a file will replace an existing file of the same name at the receiving set of files. **R** overrides the NOREPLACE option.

When used with newname, **R** applies to the newname and not to the original filename. When used with a generic name or with a range, **R** applies to all files selected for that criterion.

startrange specifies the first name in an alphanumeric range of files. Although range filenames are treated as generics, they cannot be used with an asterisk. A slash (/) separates the first and last (stoprange) filenames. When used with the SELECT statement, the first and last files specified in the range as well as all files between are copied.

stoprange specifies the last name in an alphanumeric range of files. Although range filenames are treated as generics, they cannot be used with an asterisk. A slash (/) separates the first (startrange) and last filenames. When used with the SELECT statement, the first and last files specified in the range as well as all files between are copied.

When a range in the SELECT parameter is specified, its selection can be overridden with any type of specification in the EXCLUDE parameter.

The second positional parameter (newname) of SELECT must be null when using a range and specifying NR or R.

SFSDIR=('dirid',[CRE , NOCRE])
specifies the location of the Shared File System (SFS) managed file.

Note: You cannot specify the SFSDIR and the LINK parameter for the same file.

dirid specifies the directory name in which the SFS file resides. The maximum length is 153 alphanumeric characters. The minimum specification for dirid is 'poolid:userid.'

CRE specifies to create a new or extend an existing subdirectory if the user has been granted authority by the system administrator or the definition of the subdirectory. This is the default.

NOCRE specifies to neither create a new subdirectory nor extend an existing subdirectory.

SNODE
specifies the secondary node. When SNODE is coded with the FROM parameter, the file to be copied resides on the secondary node. When SNODE is coded with the TO parameter, the file is sent to the secondary node.

SYSOPTS='!SPOOL[CLASS x]
[DIST distcode | * | OFF]
[FORM form | OFF]'
specifies CLASS, FORM and DISTCODE values for !SPOOL output. You can specify one, two, or all three subparameters. Enclose the !SPOOL string in single or double quotes.

The following is an example:

```
COPY FROM (DSN='SUPPORT.VM1500.TEXT' SNODE )      -
     TO   (DSN='!SPOOL MAINT * TEXT'              -
           SYSOPTS='!SPOOL CLASS B DIST VM1500'   -
           ) COMPRESS PRIMECHAR=X'00'
```

TYPE = typekey
specifies the member name of the type defaults file containing the file attribute defaults used to open the destination file. This typekey is specified only when defaults are requested by the user..

UNIT = (3480 | 3480X |TAPE)
specifies the device type. Acceptable values are 3480, 3480X and TAPE.

3480X specifies the unit type is an IDRC-compatible drive.

VOL =(,,[volume sequence number] ,[volume count]
,SER=volume serial number | (list))
specifies the volume serial number(s) containing the file. If VOL is not specified with the FROM parameter, the file must be cataloged.

volume sequence number (for tape files) is a number ranging from 1-255 and is used to begin processing. The default is **1**.

volume count (for tape files) specifies the maximum number of volumes needed for an output type file. The default is **5**.

ser specifies by serial number the volume(s) on which the file resides or will reside. A volume serial number is 1-6 alphanumeric characters.

VSAMCAT = (dsn,vmid,pwd,accmode,ccuu)
specifies the catalog for the VSAM file to be copied. This parameter is required only if using a catalog other than the master catalog.

dsn specifies the filename of the VSAM catalog containing the file to be copied. The maximum length is 44 characters.

vmid specifies the owner ID for the VSAM minidisk where the file is located. The maximum length is **8** characters.

pwd specifies the appropriate password for the VSAM minidisk where the file is located. The maximum length is **8** characters.

accmode specifies the link access mode. Valid access modes are NULL, **W** (primary read/write access), **M** (primary multiple access), and **MW** (primary multiple access; alternate read/write only access).

Caution: MW access to CMS format disks can be destructive. You must be able to guarantee that no other VM user, or Connect:Direct Process, has MW, M, or W access to the minidisk. If multiple users or Processes get write access to the disk at the same time, there is a high probability that the CMS directory on the disk will be destroyed. The most likely result is a message from Group Control System (GCS) or an equivalent message from CMS. The GCS message will indicate that a CSIFNS420T file system error was detected. When GCS issues the CSIFNS420T message, GCS terminates all processing.

ccuu specifies the virtual address of the disk where the VSAM file is located. Any four-digit number is valid.

Example

The following Process copies a non-VSAM file from a node named Chicago to a node named Minneapolis.

```
COPYSEQ   PROCESS    PNODE=CHICAGO SNODE=MINNEAPOLIS
STEP01    COPY  FROM (DSN='MYFILE TEXT'                              -
                      LINK=(VMID1,PASS1,RR,125))                     -
                TO    (DSN='YOURFILE TEXT'                           -
                      LINK=(VMID2,PASS2,W,126))
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

General Information on Support for a Set of CMS Files

For transmission of a set of CMS files, you can specify whether:

- An entire set of files is to be sent.
- Certain files will be selected for transmission (SELECT parameter).
- Certain files will be excluded from transmission (EXCLUDE parameter).
- Files will overlay existing files of the same name at the receiving node (REPLACE, NOREPLACE, **NR**, and **R** parameters).

A set of CMS files is specified as DSN='* ft'. When the COPY statement involves two sets of files, all files will be sent unless overriding SELECT or EXCLUDE options are specified.

Hierarchy for EXCLUDE and SELECT Parameters

The hierarchy for the EXCLUDE and SELECT parameters follows:

- EXCLUDE or SELECT cannot be used if the FROM DSN contains a member name.
- The hierarchy for the SELECT and EXCLUDE parameters proceeds from top to bottom (highest override priority to lowest) and is as follows:
  - Exclude by member name
  - Select by member name
  - Exclude by generic (or range)
  - Select by generic (or range)
  - Combine various specifications in a list after the EXCLUDE parameter

  If EXCLUDE is specified and SELECT is not specified, all files not excluded are copied. If EXCLUDE is not specified and SELECT is specified, only selected files are copied.

Group File Copy

Group file copy feature of the Connect:Direct VM/ESA enables users to send multiple files to a destination Connect:Direct node with one COPY statement. This feature can be used for backups, archive purposes, and efficient file distribution.

Group file copies are for sequential file transfers from DASD between Connect:Direct VM/ESA systems.

Group Parameter

You can supply a source and destination data set name pattern to a Connect:Direct node in the Connect:Direct COPY statement using the GROUP parameter instead of DSN. All files that match the specified source data set name pattern, called the group name, are sent by Connect:Direct software to the specified destination node.

Source Data Set Name Pattern

You can supply a source data set name in addition to the group name; however, you cannot supply a destination data set name. When a source data set name and group name are specified, Connect:Direct sends members of the group beginning with that source data set name, instead of beginning with the first member of the group. Members of a group are sent in alphabetical order.

If an error occurs during transmission of a group member, the transfer stops. If the same Process is resubmitted, all members that transferred successfully are resent. If the Process is modified so that a source data set name is specified with the name of the last group member successfully sent, only members of the group not sent previously, except for the data set specified, are sent when the Process is submitted.

Special characters to use in creating a source data set name pattern, the group name, are:

- *****, which matches any character(s)
- ?, which matches any single character

For a name to match a group name, each character in that name must correspond to a character, or special character, in the group name.

The asterisk (*****) symbol matches any number, including zero, of characters in a name. For example the group name **B*** matches the names **B** and **B123456**. In the first case, the asterisk corresponds to no characters, and in the second case, the asterisk corresponds to **123456**.

The question mark (?) symbol matches any single character in a string. For example, the group name **B?** does not match the name **B**, because there is no second character to correspond to the question mark. The group name **B?** does not match the name **B123456.** The question mark corresponds with the **1** in the string, but nothing corresponds to the remaining characters **23456**. Use **B?23456** to match **B123456**. The group name **B?** matches the names **B1** or **BB**. In the first case, the question mark corresponds to **1** , and in the second case, the question mark corresponds to the second **B**.

Destination Pattern

You can supply a destination pattern that the sending Connect:Direct node uses to construct a destination data set name for each file sent. Special symbols used to create a destination pattern that builds a destination name are in the form of **%N%**, where N is an integer between 1 and the number of special pattern-matching characters, asterisks (*****), and question marks (?), specified in the group name. Each destination pattern symbol number is determined by the order of the pattern matching symbols in the source group name specification.

For example, the group name * * uses two pattern matching characters. The destination pattern can contain the special symbols **%1%** and **%2%**. When the destination name is built by Connect:Direct, each special symbol is replaced by characters from the name determined to be in the group. The characters used to replace each special symbol are those that correspond to the corresponding pattern-matching characters from the group name. For example, the special symbol %**N**% is replaced by the characters from the name determined to be in a group that corresponds to the Nth special pattern-matching character in that group name.

Examples of Group File Copies

The following examples show various types of group file copies.

Copying All Files In a Group to a Destination with Fn Ft Unchanged

This example illustrates copying all files in a group to a destination with Fn Ft unchanged. Suppose the following files were on a disk:

```
ABC            ASSEMBLE
AAA            ASSEMBLE
ABCD           ASSEMBLE
SOURCE1        FILE
SOURCE2        FILE
```

Also, assume the following:

```
FROM GROUP='* *'
TO   GROUP='%1% %2%'
```

The group name specified, *** *** has two special pattern-matching characters (two asterisks), so the destination pattern supplied has two replacement symbols (**%1%** and **%2**%). Replacement symbol **%1%** is replaced by the characters that correspond to the first asterisk in all names determined to be in the group. Replacement symbol **%2%** is replaced by the characters that correspond to the second asterisk in all names determined to be in the group.

The transfers that occur are as follows:

```
FROM                          TO

ABC       ASSEMBLE    ---->   ABC       ASSEMBLE
AAA       ASSEMBLE    ---->   AAA       ASSEMBLE
ABCD      ASSEMBLE    ---->   ABCD      ASSEMBLE
SOURCE1   FILE        ---->   SOURCE1   FILE
SOURCE2   FILE        ---->   SOURCE2   FILE
```

Note: A more efficient way to copy all the files from one minidisk to another is to specify the FROM group as ***** and the TO group as **%1%**.

Copying Selected Files from a Group to a Destination with Fn Ft Unchanged

This example illustrates copying selected files from a group to a destination with Fn Ft unchanged.

The group name specified **a* *** includes all names that begin with an **a** or an **A**. If the specified destination pattern is **%1% %2%**, then the leading **a** from each file in the group is dropped when the destination name is built. This action occurs because the first asterisk in the group name corresponds to all the characters that follow but do not include the first **a**.

```
For       FROM GROUP='a* *'
          TO   GROUP='a%1% %2%'
```

The transfers that occur are as follows:

```
FROM                TO

ABC       ASSEMBLE    ---->   ABC       ASSEMBLE
AAA       ASSEMBLE    ---->   AAA       ASSEMBLE
ABCD      ASSEMBLE    ---->   ABCD      ASSEMBLE
SOURCE1   FILE
SOURCE2   FILE
```

Copying Selected Files from a Group to a Destination with Added Characters In Fn Ft

This example illustrates copying selected files from a group to a destination with added characters in Fn Ft.

The group name specified **a* *** includes all names that begin with an **a** or an **A**. Because the destination pattern specified includes the leading **a** specified in the group name and one additional **a**, the destination names built begin with **aa**.

```
For       FROM GROUP='a* *'
          TO   GROUP='aa%1% %2%'
```

The transfers that occur are as follows:

```
FROM                          TO

ABC       ASSEMBLE    ---->   AABC      ASSEMBLE
AAA       ASSEMBLE    ---->   AAAA      ASSEMBLE
ABCD      ASSEMBLE    ---->   AABCD     ASSEMBLE
SOURCE1   FILE
SOURCE2   FILE
```

Copying Selected Files from a Group to a Destination with Characters Stripped from Fn Ft

This example illustrates copying selected files from a group to a destination with characters stripped from Fn Ft.

```
For       FROM GROUP='s* *'
          TO   GROUP='%1% %2%'
```

The transfers that occur are as follows:

```
FROM                          TO

ABC       ASSEMBLE
AAA       ASSEMBLE
ABCD      ASSEMBLE
SOURCE1   FILE        ---->   SOURCE1   FILE
SOURCE2   FILE        ---->   SOURCE2   FILE
```

The leading **s** from each name found in the group is dropped from the names built with the supplied destination pattern.

Copying Selected Files with Equal Length Names from a Group to a Destination

This example illustrates copying selected files with equal length names from a group to a destination.

```
For       FROM GROUP='a?? *'
          TO   GROUP='a%1%%2% %3%'
```

The transfers that occur are as follows:

```
FROM                          TO

ABC       ASSEMBLE    ---->   ABC       ASSEMBLE
AAA       ASSEMBLE    ---->   AAA       ASSEMBLE
ABCD      ASSEMBLE
SOURCE1   FILE
SOURCE2   FILE
```

Copying Selected Files from a Group to a Destination with Fn Ft Formatting

This example illustrates copying selected files from a group to a destination with Fn Ft formatting.

```
For       FROM GROUP='* *'
          TO   GROUP='%1%.%2%'
```

The transfers that occur are as follows:

```
FROM                          TO

ABC       ASSEMBLE    ---->   ABC.ASSEMBLE
AAA       ASSEMBLE    ---->   AAA.ASSEMBLE
ABCD      ASSEMBLE    ---->   ABCD.ASSEMBLE
SOURCE1   FILE        ---->   SOURCE1.FILE
SOURCE2   FILE        ---->   SOURCE2.FILE
```

Copying Selected Files from a Group to a Destination with Fn Ft Reversed and Formatted

This example illustrates copying selected files from a group to a destination with Fn Ft reversed and formatted.

```
For       FROM GROUP='* *'
          TO   GROUP='%2%.%1%'
```

The transfers that occur are as follows:

```
FROM                          TO

ABC       ASSEMBLE    ---->   ASSEMBLE.ABC
AAA       ASSEMBLE    ---->   ASSEMBLE.AAA
ABCD      ASSEMBLE    ---->   ASSEMBLE.ABCD
SOURCE1   FILE        ---->   FILE.SOURCE1
SOURCE2   FILE        ---->   FILE.SOURCE2
```

The special symbols used in the destination pattern are specified in reverse order, which causes the destination names that are built to appear reversed.

Caution: Making group file copies from disks linked R/W can cause unpredictable results and is not advised.

Using SYSOPTS for DBCS Examples

Notification that a Process is transferring a DBCS file is done by the SYSOPTS statement. This statement must be included on the host node definition of the COPY statement.

By requiring the Process to notify Connect:Direct of its DBCS capability through the SYSOPTS statement, support for multiple transfers with multiple translation tables is possible. Furthermore, all Processes support compression and checkpointing.

The following example has a tablename of EBCXKSC and the default values x'0E', for so, and x'0F' for si.

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F)"
```

The following example has a tablename of KSCXEBC and the default values x'0E', for so, and x'0F' for si.

```
SYSOPTS="DBCS=(KSCXEBC,0E,0F)"
```

The following example has a tablename of EBCXKSC and the NOSO value x'00' for so and si.

```
SYSOPTS="DBCS=(EBCXKSC,00,00)"
```

The following example has a tablename of EBCXKSC and takes the defaults for so and si.

```
SYSOPTS="DBCS=(EBCXKSC)"
```

The following example has a tablename of USERTAB and takes the defaults for so and si. USERTAB is a user-defined, customized translation table.

```
SYSOPTS="DBCS=USERTAB"
```

Defining a DBCS Capable Process

The following PC-to-host DBCS translation uses the supplied translation table KSCXEBC.

```
/*****************************************************************/
/*          PC to HOST DBCS translation using table KSCXEBC      */
/*****************************************************************/
PCTOHOST  PROCESS SNODE=HOSTNODE                                    -
          HOLD=CALL
STEP01    COPY                                                      -
          TO (PNODE                                                 -
              DSN='hlq.PCFILE'                                      -
              DISP=(RPL,CATLG)                                      -
              UNIT=SYSDA                                            -
              DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS)        -
              SPACE=(254,(1000,100))                                -
              SYSOPTS="DBCS=KSCXEBC"                                -
              )                                                     -
          FROM (SNODE -
                DSN=PCFILE                                          -
                TYPE=ASC2ASC                                        -
                DISP=SHR                                            -
                )
```

The previous sample COPY statement copies a data set from a PC to a host Connect:Direct VM/ESA node; its description follows:

- The copy step is named STEP01.
- The input data set is cataloged after successful completion of the Process.
- The DCB attributes specified in the TO clause of the COPY statement are used for file allocation.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the supplied translation table KSCXEBC.
- UNIT has been specified on the PNODE only.
- The TYPE parameter on the FROM clause of the COPY statement must be set to ASC2ASC.

The following host-to-PC DBCS translation uses the supplied translation table EBCXKSC.

```
/*****************************************************************/
/*          HOST to PC DBCS translation using table EBCXKSC      */
/*****************************************************************/
HOSTTOPC  PROCESS SNODE=PCNODE                                      -
          HOLD=CALL
STEP01    COPY                                                      -
          FROM (PNODE                                               -
                DSN='hlq.HOSTFILE'                                  -
                SYSOPTS="DBCS=EBCXKSC"                              -
                DISP=(SHR)                                          -
                )                                                   -
          TO (SNODE                                                 -
              DSN=PCFILE                                            -
              TYPE=ASC2ASC                                          -
              DISP=RPL                                              -
              )
```

The previous sample COPY statement copies a data set from a host Connect:Direct VM/ESA to a PC node; its description follows:

- The copy step is named STEP01.
- The SYSOPTS attribute is specified in the FROM clause of the COPY statement is used to define the default translation table EBCXKSC.
- The TYPE parameter on the TO clause of the COPY statement must be set to ASC2ASC.

The following UNIX-to-host DBCS translation uses the default translation table EBCXKSC.

```
/*****************************************************************/
/*           UNIX to HOST DBCS translation using table EBCXKSC    */
/*****************************************************************/
STEP01    COPY                                                    -
                  FROM (PNODE                                     -
                        DSN='hlq.UNIXFILE'                        -
                        SYSOPTS="DBCS=EBCXKSC"                    -
                        DISP=(SHR)                                -
                        )                                         -
                  TO   (SNODE                                     -
                        DSN='/unixfile'                           -
                        SYSOPTS=":xlate=no:strip.blanks=no:"      -
                        DISP=RPL                                  -
                        )
```

The previous sample COPY statement copies a data set from a UNIX to a host Connect:Direct VM/ESA node; its description follows:

- The copy step is named STEP01.
- The SYSOPTS attribute is specified in the TO clause of the COPY statement is used to define the default translation table EBCXKSC.
- The SYSOPTS parameter on the FROM clause of the COPY statement is required.

The following host-to-UNIX DBCS translation uses the default translation table KSCXEBC.

```
/*****************************************************************/
/*          HOST to UNIX DBCS translation using table KSCXEBC     */
/*****************************************************************/
STEP02       COPY                                                          -
             TO   (PNODE                                                   -
                   DSN='hlq.HOSTFILE'                                      -
                   SYSOPTS="DBCS=KSCXEBC"                                  -
                   DISP=(RPL,CATLG)                                        -
                   UNIT=SYSDA                                              -
                   DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS)          -
                   SPACE=(254, (1000,100))                                 -
                   )                                                       -
              FROM (SNODE                                                  -
                    DSN='/unixfile'                                       -
                    SYSOPTS=":xlate=no:strip.blanks=no:"                  -
                    DISP=SHR                                              -
                    )
```

The previous sample COPY statement copies a data set from a host Connect:Direct VM/ESA to a UNIX node; its description follows:

- The copy step is named STEP02.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the default translation table KSCXEBC.
- The DCB attributes specified on the TO clause of the COPY statement are used for file allocation.
- Unit is specified on the PNODE.
- The SYSOPTS parameter on the FROM clause of the COPY statement is required.

Chapter 11

Connect:Direct VM/ESA RUN JOB Statement

A capability similar to RUN JOB is provided for Connect:Direct VM/ESA to allow a file to be punched to a RDR of a specified virtual machine. This feature allows sending EXEC-type job streams and data to a service machine running a product like VMBATCH or CMSBATCH in a format that can be interpreted and executed by those products.

The file name to be punched should be specified after the DSN keyword. Link information should be provided for access to the disk containing the file to be sent, following the same syntax rules described for the LINK parameter on the Connect:Direct VM/ESA COPY statement. The target virtual machine ID name should be specified after the BATCHID keyword.

The service machine, and the disk file to be punched, must be located on the same system on the pnode or snode. Otherwise, a copy must be performed prior to the run job to move the disk file to the system where the service machine is located.

The data to be sent must be in a fixed format, 80-byte record file.

Format

The Connect:Direct VM/ESA RUN JOB statement is comprised of the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN JOB**	**(DSN='filename filetype'**
		LINK=(vmid,pwd,accmode,ccuu)
		BATCHID=VM-ID-name
		PNODE | SNODE

Field Descriptions

RUN JOB
identifies the statement with all its parameters as the RUN JOB statement.

Required Parameters

DSN = 'filename filetype'
specifies the destination filename and filetype. The filename must be enclosed in single quotation marks to allow for special characters.

LINK = (userid,password,mode,ccuu)
specifies the disk where the CMS file is located. This parameter allows the user to access the CMS file.

userid specifies the owner ID for the CMS minidisk where the file is located. The valid length ranges from 1-8 characters.

password specifies the appropriate password for the CMS minidisk where the file is located. The valid length ranges from 1-8 characters. The default password is ALL.

mode specifies the link access mode.

When used with the FROM parameter, the access modes are **W** (primary read/write access), **M** (primary multiple access), **R** (primary read only), **RR** (primary and secondary read only access), **WR** (primary read/write access; alternate read only access), **MR** (primary multiple access; alternate read only access), and **MW** (primary multiple access; alternate read/write only access).

When used with the TO parameter, the access modes are **W**, **M**, **MW**, **WR**, and **MR.**

Caution: **MW** access to CMS format disks can be destructive. You must be able to guarantee that no other VM user, or Connect:Direct Process, has **MW**, **M**, or **W** access to the minidisk. If multiple users or Processes get write access to the disk at the same time, there is a high probability that the CMS directory on the disk will be destroyed. The most likely result is a message from Group Control System (GCS) or an equivalent message from CMS. The GCS message will indicate that a CSIFNS420T file system error was detected. When GCS issues the CSIFNS420T message, GCS terminates all processing.

---

ccuu specifies the virtual address of the disk where the CMS file is located. Any four-digit number is valid.

BATCHID = VM-ID-name
specifies the target virtual machine ID name.

Optional Parameters

PNODE
specifies that the job will be submitted on the primary node (PNODE), the node with Process control. PNODE is the default value.

SNODE
specifies that the job will be submitted on the secondary node (SNODE), the node that interacts with the PNODE.

Example

The example RUN JOB statement named STEP001 sends the job, BATJOB, to the reader for USER01.

```
VMBATCH   PROCESS    SNODE=CD.VM.OTHER NOTIFY=USER01
STEP001   RUN  JOB   (PNODE                                          -
                     DSN='EX BATJOB'                                 -
                     LINK=(USER01,RUSER01,RR,191)                    -
                     BATCHID=VMBATCH                                 -
                     )
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 12

Connect:Direct VM/ESA RUN TASK Statement

The RUN TASK statement allows user programs, or subtasks, to be attached during Process execution. It is permitted only within a Process and is structured so that the Process waits until the subtask completes running before the next step in the Process executes.

You can use the RUN TASK statement to attach an application subtask during Process execution. A list of parameters for the subtask can be specified in the RUN TASK statement. The subtask can be attached at either node involved in Process execution.

With the RUN TASK statement, you can pass user parameters to the subtask. The RUN TASK statistics log records the return code of the subtask, program name, parameter list, and dates and times for starting and completing the subtask.

The library that contains the program to be used with the RUN TASK statement must be defined with a Group Control System (GCS) Global Loadlib command.

Format

The Connect:Direct VM/ESA RUN TASK statement contains the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN TASK**	(**PGM=program-name**
		PARM=(parameter [,parameter,...])
		)
		PNODE | SNODE

Field Descriptions

RUN TASK
identifies the statement with all its parameters as the RUN TASK statement.

Required Parameters

PGM = program-name
specifies the name of the program to be attached as a subtask.

Note: The program runs on the node specified and has access to the DD statements allocated on that node only.

Optional Parameters

PARM = (parameter [, parameter,...])
specifies the parameters to be passed to the subtask when that subtask is attached. These parameters are the actual parameters rather than a list of addresses. Null parameters can be specified by adjacent commas.

The actual format of the parameter list passed to the program consists of a two-byte field, indicating the length of the parameter followed by the parameter itself. The valid data types for the PARM parameter follow:

CLn'value' specifies a data type of character with a length of n, where n is the number of bytes. The length is optional. If it is not specified, the actual length of the value is used. If the length specified is less than the real value, the data is truncated. If the length specified is longer than the value, the value is padded with blanks on the right; for example, CL44'FILE.NAME'.

XLn'**value**' specifies a data type of hexadecimal with a length of n, where n is the number of bytes. The length is optional. If it is not specified, the length of the value is used. If the length specified is less than the real value, the data is truncated. If the length specified is longer than the value, the value is padded on the left with binary zeros; for example, XL8'FF00'.

H'**value**' specifies a half-word value. No length can be specified. The value can be specified with a plus (+) or minus (-) sign; for example, H'-32'. If no sign is given, plus is assumed.

F'**value**' specifies a full-word value. No length can be specified. The value can be specified with a plus (+) or minus (-) sign. If no sign is given, plus is assumed; for example, F'4096'.

PLn'**value**' specifies a packed value. The length is optional; if it is not specified, the length of the value is used. If the length specified is longer than the value, the value is padded on the left with zeros. The length specifies the size of the field in bytes and

cannot be longer than 16. The value can be specified with a plus (+) or minus (-) sign; for example, PL10'+512'. If no sign is given, plus is assumed.

If no data type or length is specified, the parameter is assumed to be character type and the length of the parameter is used. For example, if PARM=('FILE.NAME') is specified, the length used is **9**.

The parameter can also be specified as a symbolic value that is resolved when the Process is submitted. If a symbol is used, the parameter must be specified without a data type designation or length; for example, &PARM1.

When using strings comprised of symbolic substitution, the strings must be enclosed in double quotes. If an ampersand (&) will be passed as part of the parameter, the data-type format must be used. For example, CL8'&PARM1' uses no substitution; CL8"&PARM1" indicates that the value is substituted.

PNODE
specifies that the program will be executed on the PNODE, which is the default.

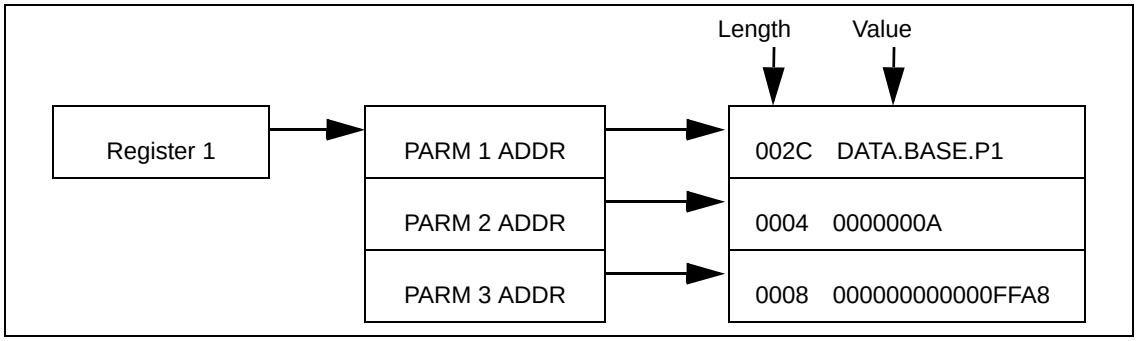
SNODE
specifies that the subtask will be attached on the secondary node (SNODE), which is the destination node.

Example

This example RUN TASK statement runs the program named MYTASK. It is attached to the Process on the secondary node (SNODE) and is passed a list of three parameter addresses.

```
STEP1     RUN TASK    (PGM=MYTASK                                    -
                      PARM=(CL44'DATA.BASE.P1',                      -
                      F'0010', XL8'FFA8'))                           -
                      SNODE
```

The parameter passing convention for the program MYTASK is shown in the following figure. In this case, Register 1 points to a parameter list of three parameters. It would contain zero (0) if no parameters were specified. Connect:Direct sets the high-order bit in PARM 3 ADDR to indicate the end of the PARM list.



See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 13

Connect:Direct VM/ESA SUBMIT Statement

During execution of a Connect:Direct Process, the SUBMIT statement causes another Connect:Direct Process to be submitted to either the PNODE (primary node), the node with Process control, or to the SNODE, the node that interacts with the PNODE during Process execution. The Process to be submitted must reside in a file on the node where the SUBMIT statement will execute. This node is referred to as the SUBNODE.

Note: The SUBMIT statement described in this chapter is not the same as the SUBMIT command. The SUBMIT statement is used within a Process to submit another Process. See the *Connect:Direct VM/ESA User's Guide* for SUBMIT command syntax and parameters.

Format

The Connect:Direct VM/ESA SUBMIT statement format includes the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the SUBMIT statement format.

Label	Statement	Parameters
stepname	**SUBMIT**	**DSN='fn ft [fm]'**
		NEWNAME=new-name
		SUBNODE=PNODE | SNODE
		SNODE=secondary-node-name
		SNODEID=(id [,pswd] [,newpswd])
		SACCT='snode-accounting-data'
		PNODEID=(id [,pswd] [,newpswd])
		PACCT='pnode-accounting-data'
		CLASS=n
		HOLD=Yes | No | Call

Label	Statement	Parameters
		PRTY=n
		NOTIFY=%USER | userid
		REQUEUE=Yes | No
		RETAIN=Yes | No | Initial
		STARTT=([date | day][,hh:mm:ssxm])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Field Descriptions

SUBMIT
identifies the statement with all its parameters as the SUBMIT statement.

Required Parameters

DSN = 'fn ft [fm]'
specifies the filename (fn), filetype (ft), and optional filemode (fm). The optional filemode is the mode of a minidisk accessed by the Connect:Direct Data Transmission Facility (DTF), not by the CMS user.

If a new Process is created that will be submitted from an existing Process, and if the existing Process is submitted immediately, the filemode of the new Process must be specified.

If the Process has been edited, the DTF machine must reaccess the disk where the Process resides.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. If CLASS is not specified in the Process, it defaults to the class value specified in the ADJACENT.NODE NETMAP record for the destination node (SNODE). Values range from 1-255.

HOLD = Yes | No | Call
specifies whether the Process is placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process. When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed in the Hold queue even if a start time is specified.

No specifies that the Process executes as soon as possible. HOLD=NO is the default.

Call specifies that the Process is placed in the Hold queue until a VTAM session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node designated as HOLD=CALL.

NEWNAME = new-name
specifies the new name to be given to the Process. The default value is the label on the PROCESS statement.

NOTIFY = %USER | userid
specifies the user to receive Process completion messages.

%USER specifies that the user submitting the Process receives the completion messages if the Connect:Direct userid is the same as the VM userid. If the Connect:Direct userid is different from the VM userid, the user is not notified.

userid specifies the VM userid that will receive Process completion messages.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one used to sign on to a Connect:Direct node.

id specifies the security ID passed to a security exit (1-8 characters).

pswd specifies the current security password. This parameter can be used by the security exit to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

newpswd specifies the new security password. This parameter can be used by the security exit to change the current security password to the new security password (1-8 alphanumeric characters).

PRTY = n

specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ is a file that holds all Processes that have been submitted to a Connect:Direct node. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect VTAM transmission priority. The range is from 0-15. If PRTY is not specified, the default is the priority defined by the PRTYDEF keyword in the Connect:Direct VM/ESA initialization parameters.

REQUEUE = Yes | No

specifies whether a COPY step should requeue if an x37 abend occurs during processing. This parameter is valid only if used when checkpointing.

Yes allows the requeued Process to be placed in the Hold queue with a status of HELD IN ERROR (HE). Corrective action can then be taken and the Process restarted with the failing step; checkpointing resumes at the last successful checkpoint. Note that the Process must be explicitly released from the Hold queue when the status is HELD IN ERROR (HE).

No causes the Process to run to completion, executing subsequent steps when a COPY step fails with an abend (such as x37). The default is NO.

RETAIN = Yes | No | Initial

keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains on the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is automatically held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time Connect:Direct is initialized. Processes submitted with RETAIN=INITIAL do not execute when initially submitted. Note that STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

SNODE = secondary-node

is a 1-16 alphanumeric character name that specifies the symbolic node name of the secondary node. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

This parameter can override the value specified in the PROCESS statement. The default value for SNODE is the value coded in the PROCESS statement. Connect:Direct allows the PNODE and SNODE to specify the same symbolic node name.

SNODEID = (id [,pswd] [,newpswd])

specifies security user IDs and passwords at the secondary node (SNODE).

id specifies the security ID passed to the security system on the SNODE (1-8 alphanumeric characters).

For Connect:Direct Tandem: This subparameter specifies the Tandem group number and user number. These numbers can range from 0-255. When specifying a Tandem value, you must use a period character as a separator between the group number and the user number.

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

For Connect:Direct Tandem: The VM node only recognizes passwords specified in uppercase alphanumeric characters. Therefore, a Process cannot be successfully initiated from Connect:Direct VM/ESA with Connect:Direct Tandem unless the Connect:Direct Tandem SNODEID password follows the same convention (no lowercase characters and no control characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

STARTT = ([date | day] [,hh:mm:ssXM])

specifies that the Process will be executed at a selected date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

---

date specifies that the Process is to be held until a specific date. You can specify the day (dd), month (mm), and year (yy for 2-digit year and yyyy for 4-digit year).

You can use periods or slashes (/) to separate the components of a date value.

Note: You can omit the separators only for transfers between mainframe nodes. Sterling Commerce recommends the use of separators to guarantee transfers between all platforms.

To specify the order of a Gregorian day, month, and year, you *must* define the DATEFORM initialization parameter. If you do not specify the DATEFORM parameter, Connect:Direct VM/ESA defaults to the DATEFORM=MDY formats described in this section.

After you designate the date order in your initialization parameters, you can use the following date formats:

DATEFORM=MDY specifies the date format as:

- mm/dd/yy *or* mm/dd/yyyy
- mm.dd.yy *or* mm.dd.yyyy

DATEFORM=DMY specifies the date format as:

- dd/mm/yy *or* dd/mm/yyyy
- dd.mm.yy *or* dd.mm.yyyy

DATEFORM=YMD specifies the date format as:

- yy/mm/dd *or* yyyy/mm/dd
- yy.mm.dd *or* yyyy.mm.dd

DATEFORM=YDM specifies the date format as:

- yy/dd/mm *or* yyyy/dd/mm
- yy.dd.mm *or* yyyy.dd.mm

The following Julian date formats are valid:

- yyddd *or* yyyyddd
- yy/ddd *or* yyyy/ddd
- yy.ddd *or* yyyy.ddd

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

---

day specifies the day of the week that the Process is to be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value can be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday with Monday as the only STARTT parameter, the Process does not run until the following Monday.

You can also specify TODAY, which releases the Process for execution today, or TOMORROW, which releases the Process for execution the next day.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT, which releases the Process for execution at midnight.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed in the Hold queue even if a start time is specified.

SUBNODE = PNODE | SNODE

specifies the node on which the Process defined in this SUBMIT statement will execute. Specifying PNODE means that the Process is submitted on the node that has Process control. Specifying SNODE means that the Process is submitted on the node participating in, but not controlling, Process execution. In both cases, the Process must reside on the node on which it is being submitted. The default is PNODE.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden when submitting the Process.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Note that the symbolic parameter for the SUBMIT statement must begin with a single ampersand. This allows submission of the Process containing the SUBMIT statement to resolve symbolic parameters correctly.

An ampersand symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Example

This example shows using SUBMIT with the DSN parameter. Symbolics for PROCESS2 are supplied by the operator submitting PROCESS1.

The operator at the PNODE issues the following Connect:Direct SUBMIT PROCESS command to initiate the file transfer:

```
SUB       PROC=PROCESS1                                                 -
          &DSN1='TEST FILE'                                             -
          &DSN2='TEST FILE2'
```

In the following Process, CD.LA and CD.DALLAS are Connect:Direct VM/ESA nodes.

```
PROCESS1  PROCESS    PNODE=CD.LA  SNODE=CD.DALLAS                       -
                     &DSN1=&DSN1                                        -
                     &DSN2=&DSN2                                        -
                     &PRTY=14                                           -
                     NOTIFY=A345
COPYSTEP  COPY  FROM (DSN=&DSN1 DISP=SHR PNODE                          -
                     LINK=(USER1,READ,RR,191))                          -
                TO   (DSN=&DSN2 DISP=SHR SNODE                          -
                     LINK=(USER2,WRITE,W,191))                          -
SUBSTEP   SUBMIT     DSN=A345.PROCESS.LIB(PROCESS2)                     -
                     PRTY=&PRTY                                         -
                     SUBNODE=SNODE                                      -
                     &DSN=&DSN2
```

PROCESS1 submits PROCESS2, where CD.DALLAS is a Connect:Direct VM/ESA node and CD.NEWYORK is a Connect:Direct OS/390 node.

```
PROCESS2  PROCESS    PNODE=CD.DALLAS                                    -
                     SNODE=CD.NEWYORK
COPYSTEP  COPY  FROM (DSN=&DSN2 PNODE                                   -
                     LINK=(USER2,READ,RR,191)                           -
                TO   (DSN=A345.NEW.DATA1 DISP=SHR)
```

- PROCESS1 copies the file TEST FILE in LA to a file called TEST FILE2 in Dallas. It then submits PROCESS2, which executes on the Dallas node. PROCESS2 is submitted with a PRTY of 14.
- PROCESS2 copies the file TEST FILE2 in Dallas to the file A345.NEW.DATA1 in New York.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 14

Connect:Direct VM/ESA SYMBOL Statement

The Connect:Direct VM/ESA SYMBOL statement allows you to build symbolic substitution values.

Format

The Connect:Direct VM/ESA SYMBOL statement format includes the following parameters. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**SYMBOL**	**&symbolic_name=variable-string**

Field Descriptions

SYMBOL
identifies the statement with all its parameters as the SYMBOL statement.

Required Parameters

&symbolic_name=variable-string
specifies the string that is substituted into the Process.

When the Connect:Direct software encounters an ampersand (&) plus 1-17 alphanumeric characters, Connect:Direct substitutes a string represented by that ampersand and the alphanumeric characters.

Symbols in the string are resolved from previously specified values in a PROCESS, SUBMIT, or SYMBOL statement. With the SYMBOL statement, different pieces of a Connect:Direct statement string can be concatenated, allowing the user to move data in a variety of ways.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Example

In the following Process, PROCSYM, the SYMBOL statement causes the file Z813103.ABC.DEF to be copied into the PDS Z813103.TEST.XXX.

Note that two vertical bars preceded and followed by blanks (||) are used to indicate concatenation. Bracketing backslashes ensure that special characters within the string are maintained. The COPY statement is resolved by the symbolic substitution of the &AA symbol to Z813103.ABC.DEF.

```
/* PROCESS STATEMENT */
PROCSYM   PROCESS    HOLD=YES                                          -
                     SNODE=CD.NEWYORK                                  -
                     SNODEID=(RSMITH,ROGER)                            -
                     PRTY=12
          SYMBOL     &A1=ABC
          SYMBOL     &A2=DEF
          SYMBOL     &AA=\Z813103.\ || &A1 \.\ || &A2
/*  COPY STATEMENT   */
STEP1     COPY  FROM (DSN=&AA)                                         -
                TO   (DSN=Z813103.TEST.XXX                             -
                     DISP=OLD)
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 15

Connect:Direct VM/ESA Conditional Statements

Conditional statements permit you to alter the sequence of Connect:Direct Process execution based on the completion of the previous step in the Process.

Formats

Formats for Connect:Direct VM/ESA conditional statements follow. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**IF**	**(label condition nn) THEN**
		(process steps)
	ELSE	
		(alternative process steps)
	EIF	
optional	GOTO	label
	EXIT	

Statement Descriptions

IF THEN

specifies that the Connect:Direct system executes a block of Connect:Direct statements based on the completion code of a Process step. An EIF statement must be used in conjunction with an IF THEN statement. A return code with the high order bit on is evaluated as a negative return code. See *Field Descriptions* on page 128 for details.

ELSE
designates a block of Connect:Direct statements that execute when the IF THEN condition is not satisfied. No parameters exist.

EIF
is required for specifying the end of the IF THEN or IF THEN ELSE block of statements. No parameters exist.

GOTO
moves to a specific step within a Process. See *Field Descriptions* on page 128 for details.

EXIT
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

label
For the **IF THEN statement**, the label specifies the name of a previous step whose completion code is used for comparison.

For the **GOTO statement**, the label specifies the name of a subsequent step in a Process (required for GOTO only). The name can neither be the label of a preceding step nor the label of the GOTO statement of which it is a part.

Note: User-defined labels must begin in column **1**. The label consists of a 1-8 character alphanumeric string, with the first letter alphabetic only.

condition
specifies the type of comparison to be performed. This condition checking can be based on comparisons for equality, inequality, greater than, less than, greater than or equal to, and less than or equal to.

The completion code from RUN JOB is for the job submission only and not the completion code of the job submitted.

Valid symbols, alternate symbols, and conditions follow:

- **= or EQ** specifies that the completion code must be equal to the value nn for the condition to be satisfied.
- **<> or ¬= or NE** specifies that the completion code must not equal the value nn for the condition to be satisfied.
- **>= or ¬< or GE** specifies that the completion code must be greater than or equal to the value nn for the condition to be satisfied.
- **> or GT** specifies that the completion code must be greater than the value nn for the condition to be satisfied.

- **<= or ¬> or LE** specifies that the completion code must be less than or equal to the value nn for the condition to be satisfied.
- **< or LT** specifies that the completion code must be less than the value nn for the condition to be satisfied.

nn

specifies the numeric value to be used for completion code checking. If coded as X‘nn’, it is a hexadecimal value; any other coding indicates it as decimal.

If a completion code less than **4** is returned, typically the Process completed successfully. In most cases, a return code greater than **4** indicates the Process ended in error. Note that a return code equaling **4** indicates a warning.

THEN

specifies subsequent processing to be performed if the condition specified is true.

Example

The following is an example of a Process that contains all of the conditional statements. A description of each step follows the example Process.

```
COPY01   PROCESS    SNODE=CD.CHICAGO
STEP01   COPY  FROM (DSN=ABC.FILEA PNODE)                              -
               TO   (DSN=JKL.FILEA)
STEP02   IF         (STEP01 GT 4) THEN
                        GOTO STEP07
         ELSE
STEP03   RUN JOB    (DSN=USERJOB) SNODE
         EIF
STEP04   IF         (STEP03 >= 8) THEN
                         EXIT
         EIF
STEP05   IF         (STEP03 LT 4) THEN
STEP06   COPY  FROM (DSN=ABC SNODE)                                   -
               TO   (DSN=MNO PNODE)
         EIF
         EXIT
STEP07   RUN TASK   (PGM=DMNOTIFY,                                    -
                    PARM=('FAIL',ABC.FILEA))                          -
                    PNODE
```

COPY01 is the PROCESS statement defining the secondary node as CD.CHICAGO.

STEP01 copies file ABC.FILEA on the PNODE to file JKL.FILEA on the SNODE.

STEP02 checks the completion code of STEP01. If STEP01 fails, STEP07 executes. If STEP01 ended with a completion code of **4** or less, STEP03 executes.

STEP03 submits the job, USERJOB, on the SNODE.

STEP04 checks the completion code of STEP03. If STEP03 fails with a completion code of **8** or greater, the Process terminates. Otherwise, STEP05 executes.

STEP05 checks the completion code from STEP03. If less than 4, indicating the step completed without errors, the COPY statement in STEP06 executes and the Process terminates.

STEP06 copies file ABC on the SNODE to file MNO on the PNODE.

STEP07 only executes if STEP01 fails. The program DMNOTIFY runs, sending an OPERATION FAILED message to the console operator.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 16

Connect:Direct VSE PROCESS Statement

A Connect:Direct VSE PROCESS statement defines the attributes of a Connect:Direct Process and is always the first statement in a Process.

Note: The maximum storage area allowed for a Connect:Direct VSE Process statement is 64K. To accommodate a larger Process, split the Process into two separate Processes. Include a SUBMIT statement in the first Process to run the second Process.

Format

The Connect:Direct VSE PROCESS statement format includes the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the PROCESS statement format.

Label	Statement	Parameters
process name	**PROC**ess	**SNODE=secondary-node-name**
		SACCT=(snode-accounting-data)
		PNODE=primary-node-name | TCPNAME=primary-node-name
		PNODEID=(id | [,pswd] [,newpswd])
		SNODEID=(id | [,pswd] [,newpswd])
		%NUM1=process-submit-time
		PACCT=(pnode-accounting-data)
		CLASS=n
		HOLD=Yes | No | Call
		PRTY=n
		RETAIN=Yes | No | Initial
		STARTT=([date |day][,hh:mm:ssXM])

Label	Statement	Parameters
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Field Descriptions

process name
specifies the name of the Process. The Process name can be from 1-8 characters long. The first character must be alphabetic. The Process name must start in column one. This label is used to identify the Process in any messages or statistics relating to this Process.

PROCess
identifies the statement with all its parameters as the PROCESS statement. This statement identifier can be abbreviated to PROC.

Required Parameters

SNODE = secondary-node-name
is a 1-16 character alphanumeric name that specifies the secondary node (SNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

This is the logical node name that has been defined in the ADJACENT.NODE entry for that node in the network map.

Note: This parameter is not required if SNODE is specified on the SUBMIT command.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. If CLASS is not specified in the Process, it will default to the class value specified in the ADJACENT.NODE NETMAP record for the destination node (SNODE). Values range from 1-255.

HOLD = Yes | No | Call
specifies whether the Process is placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process.

No specifies that the Process is to execute as soon as possible. HOLD=NO is the default.

Call specifies that the Process is to be placed in the Hold queue until a VTAM session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

%NUM1 = process-submit-time
used to specify unique temporary data set names in the DSN parameter of the COPY TO statement. The variable is resolved as the submission time of the Process in a six-digit numeric-value format, that is, in minutes, seconds, and fraction of seconds. The %NUM1 value must be preceded by an alphabetic character.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in parentheses. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

PNODE = primary-node-name
is a 1-16 character alphanumeric name that specifies the primary node (PNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

The node to which the Process is submitted is always the PNODE. This parameter defaults to the name of the node submitting the Process and need not be specified. It is used for documentation purposes only.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one you used to sign on to the Connect:Direct system.

id specifies the security ID passed to the security system at the PNODE (1-8 alphanumeric characters).

pswd specifies the current security password for the specified ID. This parameter can be used by the security system at the PNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

newpswd specifies the new security password. It can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

PRTY = n

specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ holds all Processes that have been submitted to Connect:Direct. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect VTAM transmission priority. The range is from 0-15. If PRTY is not specified, the default is the priority defined by the PRTYDEF keyword in the Connect:Direct initialization parameters for Connect:Direct VSE. See the *Connect:Direct VSE/ESA Installation Guide* for a discussion of the initialization parameters.

RETAIN = Yes | No | Initial

keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains in the Hold queue after initial execution. The Process must be released manually through the CHANGE PROCESS command to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval. However, a date is invalid as a STARTT subparameter when used in conjunction with RETAIN.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time Connect:Direct is initialized. The Process does not execute when initially submitted. Note that STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in parentheses. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

For Connect:Direct OS/400: This parameter is ignored when the SNODE is a Connect:Direct OS/400 node.

SNODEID = (id [,pswd] [,newpswd])

specifies security user IDs and passwords at the secondary node (SNODE).

For Connect:Direct OS/400: If an SNODEID and password of the Process submitter is not coded in the PROCESS statement, the user ID and password of the Process submitter will be used for the security ID and password check by Connect:Direct OS/400.

id specifies the security ID passed to the security system on the SNODE (1-8 alphanumeric characters).

For Connect:Direct Tandem: This subparameter specifies the Tandem group number and user number. These numbers can range from 0-255. When specifying a Tandem

value, you must use a period character as a separator between the group number and the user number.

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles can be 10 characters long.

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

For Connect:Direct Tandem: The VSE node only recognizes passwords specified in uppercase alphanumeric characters. Therefore, a Process cannot be successfully initiated from a Connect:Direct VSE node with a Connect:Direct Tandem node unless the Connect:Direct Tandem SNODEID password follows the same convention (no lowercase characters and no control characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

For Connect:Direct OS/400: This subparameter is ignored.

STARTT = ([date | day][,hh:mm:ssXM])

specifies that the Process will execute at a selected date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

---

date specifies that the Process is to be held until a specific date. You can specify the day (dd), month (mm), and year (yy for 2-digit year and yyyy for 4-digit year).

You can use periods or backslashes (/) to separate the components of a date value.

Note: You can omit the separators only for transfers between mainframe nodes.

To specify the order of a Gregorian day, month, and year, you *must* define the DATEFORM initialization parameter. If you do not specify the DATEFORM parameter, Connect:Direct VSE defaults to the VSE user setup option STDOPT DATE which is specified at system initial program load (IPL).

After you designate the date order in your initialization parameters, you can use the following date formats:

DATEFORM=MDY specifies the date format as:

- mm/dd/yy or mm/dd/yyyy
- mm.dd.yy or mm.dd.yyyy

DATEFORM=DMY specifies the date format as:

- dd/mm/yy or dd/mm/yyyy
- dd.mm.yy or dd.mm.yyyy

DATEFORM=YMD specifies the date format as:

- yy/mm/dd or yyyy/mm/dd
- yy.mm.dd or yyyy.mm.dd

DATEFORM=YDM specifies the date format as:

- yy/dd/mm or yyyy/dd/mm
- yy.dd.mm or yyyy.dd.mm

The following Julian date formats are valid:

- yyddd or yyyyddd
- yy/ddd or yyyy/ddd
- yy.ddd or yyyy.ddd

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

---

day specifies the day of the week the Process will be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value can be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday, with Monday as the only STARTT parameter, the Process will not run until the following Monday.

You can also specify TODAY, which releases the Process for execution the day and time of Process submission (unless the time of day is specified), or TOMORROW, which releases the Process for execution the next day. If a time of day is not specified with TOMORROW, the Process will execute after midnight.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON or MIDNIGHT.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time is specified.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden in the SUBMIT command.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Example

The following is an example PROCESS statement; its description follows:

```
PROC1    PROCESS    SNODE=CD.NODE.A                                    -
                    SNODEID=(JONES,OPENUP)                             -
                    CLASS=4                                            -
                    HOLD=YES                                           -
                    PACCT='OPERATIONS, DEPT. 87'                       -
                    RETAIN=NO
```

The Process named PROC1 specifies a secondary node (SNODE) of CD.NODE.A. The corresponding security userids and passwords (SNODEID) have been included.

This Process will run in CLASS 4.

The Process will be placed on the Hold queue until it is released for execution with a CHANGE PROCESS command.

The PACCT parameter specifies that all accounting information will be attributed to the operations account, department 87, if the node has a program that maintains this information.

Once the Process executes, it is deleted because the RETAIN parameter is set to NO. Note that NO is the default value for the RETAIN parameter.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 17

Connect:Direct VSE COPY Statement

The Connect:Direct VSE COPY statement allows you to copy files from one node to another.

Supported File Types

The Connect:Direct VSE COPY statement copies the following types of files:

- Sequential Access Method (SAM)
- Virtual Storage Access Method (VSAM)
- ISAM
- VSAM-managed SAM
- VSE/POWER LST and PUN Entries
- VSE/POWER XMT Entries (creation only)
- VSE/POWER RDR Entries (creation only)
- VSE Librarian Members (all types)
- CA-DYNAM or CA-EPIC Controlled Data Sets

Both disk and tape transfers are supported.

Format

The Connect:Direct VSE COPY statement is comprised of a FROM parameter that includes the source file name and a TO parameter that includes the destination file name. Additional parameters can be specified to further customize the file transfer operation.

Note: The length of the entire COPY statement cannot exceed 2040 bytes.

The following pages provide the format of the COPY statement. All platform-specific parameters and subparameters are listed along with their possible values. A description of each parameter and subparameter follows the format.

To copy from one Connect:Direct system environment to another, refer to the appropriate COPY FROM and TO sections for those environments. For example, if the source file is located on a Connect:Direct Tandem node, refer to the COPY FROM information for Connect:Direct Tandem. If the destination for the file is a Connect:Direct VSE node, refer to the TO information in this chapter.

Label	Statement	Parameters	
stepname	**COPY**	**FROM**	**(**
			DSN=data-set-name /password
			PNODE | SNODE
			DCB=([model-file-name] [,BLKSIZE=no.-bytes] [,DEN=0 | 1 | 2 | 3 | 4] [,DSORG=PS | VSAM | MSAM] [,LRECL=no.-bytes] [,OPTCD=W | Q | Z] [,RECFM=record-format] [,TRTCH=C | E | T | ET | COMP | NOCOMP])
			DISP=([OLD | SHR],KEEP | DELETE])
			LABEL=([file-sequence-number] ,[SL | AL | BLP | LTM | NL] ,[PASSWORD | NOPWREAD] ,[IN | OUT] ,[RETPD=nnnn | EXPDT=yyddd | yyyy/ddd])

Label	Statement	Parameters	
			LIBR=([EXCLMEM=(generic|mem| start-range/stop-range|list)] [EXCLSLIB=(generic|sublib| start-range/stop-range|list)] [EXCLTYPE=(generic|type| start-range/stop-range|list] [REPLACE=YES|NO] [SELMEM=member|generic|*| (member, [new-name],[NR|R])| (generic,,[NR|R])| (start-range/stop-range,,[NR|R])| (list)] [SELSLIB=sublibrary|generic|*| (sublibrary, [new-name],[NR|R])| (generic,,[NR|R])| (start-range/stop-range,,[NR|R])| (list)] [SELTYPE=type|generic|*| (type, new-name)| (start-range/stop-range)| (list)] [*])
			LST=([CLASS=class] [DISP=D|K|H|L] [PWD=password])
			PUN=([CLASS=class] [DISP=D|K|H|L] [PWD=password])
			SPACE=(strt-trk | strt-blk,(prim))
			SYSOPTS="DBCS=(tablename,so,si)"
			UNIT=([group name | device-type | unit address | DLBLONLY | TNOASGN])

Label	Statement	Parameters	
			VOL=([PRIVATE],[volume-count] ,[SER=(serial-no [,serial-no,...])])| ([SER=(serial-no , [serial-no,...])
			BUFND=number
			VSAMCAT=(dsn, mode, userid, pswd, cuu)
			IOEXIT=exit-name | (exit-name [,parameter,...])
			SYSOPTS="DBCS=(tablename,so,si,PAD|PAD=pc)"
			)
		TO	**(**
			DSN=data-set-name /password
			PNODE | SNODE
			TYPE=typekey
			DCB=([model-file-name] [,BLKSIZE=no.-bytes] [,DEN=0 | 1 | 2 | 3 | 4] [,DSORG=PS | VSAM | MSAM] [,LRECL=no.-bytes] [,OPTCD=W | Q | Z] [,RECFM=record-format] [,TRTCH=C | E | T | ET | COMP | NOCOMP])
			DISP=([NEW | OLD | MOD | RPL | SHR], [KEEP])
			LABEL=([file-sequence-number] ,[SL | AL | BLP | LTM | NL] ,[PASSWORD | NOPWREAD], [IN | OUT] ,[RETPD = nnnn | EXPDT = yyddd | yyyy/ddd])

Label	Statement	Parameters	
			LIBR=([DATA=NO|YES] [LOCKID=lockid] [MSHP=NO|YES|OVERRIDE] [REPLACE=YES|NO] [RESETLOCK=NO|YES] [REUSE=AUTOMATIC|IMMEDIATE] [SLIBDISP=SHR|OLD|NEW|RPL] [SUBLIB=sublibrary] [TYPE=type] [*])
			LST=([BANNER=(literal1=name1, literal2=name2,...)] [BLDG=building] [BURST=YES|NO] [CC=M|ASA|(NOCC,[TOF=1|char|x'xx'], [LINECT=55|nn])] [CHARS=(tablename,tablename)] [CICSDATA=CICS-data] [CKPTLINE=nnnnn] [CKPTPAGE=nnnnn] [CKPTSEC=nnnnn] [CLASS=class] [COMPACT=compaction-table-name] [CONTROL=program single double triple] [COPIES=nnn] [COPY=nnn] [DEFAULT=YES|NO] [DEPT='department-identification'] [DEST=nodename(nodename,userid)] [DFLT=YES|NO] [DISP=D|K|H|L] [DIST=distribution] [FCB=fcb-name] [FLASH=(overlay-name,count)]

Label	Statement	Parameters	
			[FORMDEF=membername] [FORMS=form-name] [FNO=form-name] [HOLD=YES|NO] [JSEP=nnnnn] [MODIFY=module-name] [PAGEDEF=membername] [PRI=n] [PRMODE=process-mode] [PROGR='programmer-name'] [PRTY=n] [PWD=password] [ROOM='room-identification'] [SUBNAME=submitter's-name [SYSID=n] [THRESHLD=nnnnnnnn] [TRC=YES|NO] [UCS=character-set-name (character-set-name,FC)] [USER='user-data-description'] [USERID=userid] [WRITER=writer-name])

Label	Statement	Parameters	
			PUN=([BLDG=building] [CC=M|ASA|(NOCC)] [CICSDATA=CICS-data] [CKPTLINE=nnnnn] [CKPTPAGE=nnnnn] [CKPTSEC=nnnnn] [CLASS=class] [CONTROL=program single double triple] [COPIES=nnn] [COPY=nnn] [DEPT='department-identification'] [DEST=nodename (nodename,userid)] [DISP=D|K|H|L] [DIST=distribution] [FORMS=form-name] [FNO=form-name] [HOLD=YES|NO] [JSEP=nnnnn] [PRI=n] [PRMODE=process-mode] [PROGR='programmer-name'] [PRTY=n] [PWD=password] [ROOM='room-identification'] [SUBNAME=submitter's-name] [SYSID=n] [THRESHLD=nnnnnnnn] [USER='user-data-description'] [USERID=userid] [WRITER=writer-name])
			SPACE=[(start-track | start-block,(allocation))] [(trigger key,(allocation))] [(record size,(primary allocation,secondary allocation))]

Label	Statement	Parameters	
			UNIT=([group name | device-type | unit address | DLBLONLY | TNOASGN])
			VOL=([PRIVATE][RETAIN] ,[volume-count] ,[SER=(serial-number [,serial-number,...])]) | ([SER=(serial-no ,[serial-no,...])
			VSAMCAT=(dsn, mode, userid, pswd, cuu)
			IOEXIT=exit-name | (exit-name [,parameter,...])
			SYSOPTS="DBCS=(tablename,so,si,PAD|PAD=pc)" "parameter1[,parameter2,...]"
			BUFND=number
			)
			CKPT=nK | nM
			COMPRESS [PRIMEchar=X'40' |X'xx' | C'c' | EXTended]

Field Descriptions

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source file characteristics.

(FROM) DSN =
specifies the source data set name. Data set names are verified based on the standard VSE data set name conventions. If the data set name does not follow VSE naming conventions, the data set name must be enclosed in single quotation marks to allow for special characters. Note that GDG data sets (controlled by Epic or DYNAM) can be copied by either the relative generation number or the absolute generation number.

Note: DSN is optional when used with the IOEXIT parameter.

Connect:Direct VSE does not support multi-extent source data sets.

If the data set being copied from requires a password for read or the data set being copied to requires a password for write, the password can be specified in the COPY statement after the data set name. A slash (/) must follow the data set name and precede the password. This password is used at data set allocation. If it is not correct, VSE issues a WTOR requesting the password when Connect:Direct VSE software opens the data set. For example:

```
COPY FROM DSN=data-set-name/pwd...
```

TO

specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN =

specifies the destination data set name.

Note: DSN is optional when used with the IOEXIT parameter.

If the data set name does not follow standard VSE data set name conventions, the data set name must be enclosed in single quotation marks to allow for special characters.

If the data set being copied from requires a password for read or the data set being copied to requires a password for write, the password can be specified in the COPY statement after the data set name. A slash (/) must follow the data set name and precede the password. This password is used at data set allocation. If it is not correct, VSE issues a WTOR requesting the password when Connect:Direct VSE software opens the data set. For example:

```
COPY TO DSN=data-set-name/pwd...
```

Optional Parameters

BUFND = number

specifies the number of I/O buffers VSAM will use for transmitting data between virtual and auxiliary storage. A buffer is the size of a control interval in the data component. Valid values range from 1-510. The default is **2**. Increasing this number generally improves the I/O performance on the file but requires more memory.

CKPT = nK | nM

specifies the byte interval for checkpoint support, which allows restart of interrupted transmissions at the last valid transmission point, thus avoiding the need to restart transmission from the beginning. **K** denotes thousands; **M** denotes millions. A checkpoint value of zero stops automatic checkpointing.

Connect:Direct software converts the value to a block boundary, and a data transmission checkpoint is taken at that position.

Checkpointing is controlled by the SNODE.

```
CKPT/RESTART of VSAM-managed SAM files is not supported.
```

For Connect:Direct OS/400: Connect:Direct OS/400 does not support checkpointing for versions prior to 1.4.00.

COMPRESS [PRIMEchar = X'40' |X'xx' | C'c' | EXTended]
specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at its destination. The default subparameter for the COMPRESS parameter is PRIMEchar=X'40'.

Note: Compression is CPU-intensive, and its effectiveness is data dependent. It should only be used if its benefits are known.

If compression is specified, Connect:Direct reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character are compressed to **1** byte.
- Repetitive occurrences (ranging from 3-63) of any other character are compressed to **2** bytes.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X'40').

EXTended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited, CPU is available, and data is repetitive.

The following are valid options for EXTended:

- **CMPrlevel** determines the compression level. The valid value range is 1-9. Level **1** is the fastest compression, but it offers the lowest degree of compression. A higher compression level produces a higher quality of compression, but the higher level has the slowest rate of compression. The default is 1.
- **WINdowsize** determines the size of the compression window or history buffer. This memory is above the line. The valid values are 8-15. Higher windowsize specifications increase the degree of compression and use more virtual memory. Size **8** uses **1** KB of memory where Size 15 requires 128 KB of memory. The default is 13.
- **MEMlevel** identifies how much virtual memory is allocated to maintain the internal compression state. This memory is above the line memory. The valid value range is 1-9. Level **1** requires the least memory (1K), but it reduces the degree of compression. Level **9** provides the fastest speed, but it uses the most memory (256K). The default is 4.

The following example shows one way to specify the various EXTended options in a COPY statement:

```
COMPRESS EXT = ( CMP=4
                 WIN=12
                 MEM=8
                )
```

DCB =([model-file-name]
[,BLKSIZE = no.-bytes]
[,DEN = 0 | 1 | 2 | 3 | 4]
[,DSORG = PS | VSAM | MSAM]
[,LRECL = no.-bytes]
[,OPTCD = W | Q | Z]
[,RECFM = record-format]
[,TRTCH = C | E | T | ET | COMP | NOCOMP])

specifies attributes to be used in allocating source and destination files. For existing source and destination files, DCB attributes are determined from the operating system unless specified. For a new destination file, the DCB attributes of the source file are used to allocate the destination file unless DCB information is provided in the Process.

Note: You are not required to specify a DCB parameter when the input data set is VSE/POWER. Connect:Direct sends this information to the "TO" side of the Process. You can, however, specify different attributes to override the default DCB information.

When you transfer data from a VSE Librarian member, in the DCB parameter, specify the DSORG of PS for BSAM libraries or VSAM for VSAM-managed libraries. Connect:Direct will propagate this information.

model-file-name specifies a model data set control block (DSCB).

BLKSIZE specifies the length in bytes of the block. The minimum length is 18 bytes, and the maximum length is 32,760 bytes.

DEN specifies the magnetic tape mode setting. The values for the DEN parameter for 7- and 9-track tape are shown in the following table. When coded together, the DEN and TRTCH values are used to select a tape device for allocation by Connect:Direct VSE.

DEN	7-Track Tape	9-Track Tape
0	200 bpi	-
1	556 bpi	-
2	800 bpi	800 bpi (NRZI). NRZI is Non-Return-to-Zero Inverted recording mode.
3	-	1600 bpi (PE). PE is Phase Encoded recording mode.
4	-	6250 bpi (GCR). GCR is Group Coded recording mode.

DSORG specifies the file organization. File organizations supported are PS, VSAM, and MSAM (VSAM-managed SAM).

When copying TO a VSAM-managed SAM file, in addition to including the DCB=(DSORG=MSAM) parameter, space must be allocated with the SPACE parameter.

LRECL specifies the record length in bytes.

Note: When RECFM=V or RECFM=VB type files are used, the LRECL value must be at least the size of the largest record in the file plus 4 bytes. If RECFM=V, the BLKSIZE value must be at least the LRECL value plus another 4 bytes. If RECFM=V, the BLKSIZE value does not need to be an even multiple of LRECL.

For Connect:Direct Tandem: An entry-sequenced file coming from Tandem to an IBM PS/VB must be specified with an LRECL 4 bytes larger than the Tandem file. This is specified on the TO clause of the COPY statement to account for a 4-byte-length area required on the IBM file but not required on Tandem.

OPTCD specifies optional processing associated with this file. This specification only applies to this file and is not automatically applied to the other files involved in the COPY operation. Valid options include the following:

- **W** performs write validity checks on direct access storage devices.
- **Q** performs ASCII-to-EBCDIC conversion for input files and EBCDIC-to-ASCII conversion for output files. Q is the default and only used for AL-labeled tape files.
- **Z** performs reduced error recovery for tape files.

RECFM specifies the format of the records in the file. Any valid record format, such as F (Fixed), FB (Fixed Block), U (Undefined), V (Variable), VB (Variable Block), VS (Variable Spanned), and VBS (Variable Block Spanned), can be specified.

For Connect:Direct OpenVMS: An OpenVMS file with a record format of undefined (U) cannot be copied to VSE.

TRTCH specifies the magnetic tape mode setting. When coded together, the TRTCH and DEN values are used to select a tape device for allocation by Connect:Direct VSE. Valid options are as follows:

- **C** specifies data conversion, odd parity, and no translation.
- **E** specifies no data conversion, even parity, and no translation.
- **T** specifies no data conversion, odd parity, and BCD or EBCDIC translation.
- **ET** specifies no data conversion, even parity, and BCD or EBCDIC translation.
- **COMP** is a feature for 3480X tape drives only. It enables Improved Data Recording Capability (IDRC), which compresses the data. COMP overrides the system-wide IDRC setting for no compression. If you are specifying COMP, you must also include a UNIT= parameter that specifies either 3480X or a systems-programmer-defined name equivalent to a 3480X tape drive.

- **NOCOMP** overrides the system-wide IDRC setting for compression. It applies to 3480X tape drives only.

(FROM) DISP = ([OLD | SHR] , [KEEP | DELETE])

specifies the status of the file and what is to be done with the file after notification of successful transmission. Subparameters are as follows:

First Subparameter specifies the status of the file prior to execution of the Process. This subparameter applies to all files. Options for this subparameter are as follows:

- **OLD** specifies that the source file existed before the Process began executing and the Process is given exclusive control of the file.
- **SHR** specifies that the source file existed before the Process began executing and that the file can be used simultaneously by another job or Process. The default is SHR.

Second Subparameter specifies the disposition of the file following a normal Process step termination resulting in a zero completion code. This subparameter applies to non-VSAM files. Valid source file dispositions are as follows:

- **KEEP** specifies that the system keeps the file after the Process step completes.
- **DELETE** specifies that the system deletes the file after the Process step completes successfully.

(TO) DISP = ([NEW | OLD | RPL | SHR] [,KEEP|CATLG])

specifies the status of the file on the receiving node. Subparameters are as follows:

First Subparameter specifies the status of the file before the Process executes. Only the OLD and RPL dispositions apply to VSAM files. Valid options for this subparameter are as follows:

- **NEW** specifies that the Process step will create the destination file. NEW is the default.
- **OLD** specifies that the destination file already exists. The Process will have exclusive control of the file. If DISP=OLD, the destination file can be a VSAM or a SAM file.
- **RPL** specifies that the destination file will replace any existing file, or, if none exists, will allocate a new file. DISP=RPL can be specified for SAM or VSAM files. If the file is VSAM, it must be defined with the REUSE attribute, which specifies that the file can be opened and reset to the beginning.
- **SHR** specifies that the destination file already exists. The file can be used simultaneously by another job or Process.

Second Subparameter specifies the normal termination disposition, but does not apply to VSAM files. Valid destination file dispositions are as follows:

- **KEEP** specifies that the system keeps the file after the Process step completes. If DISP=(NEW,KEEP), a volume serial number also must be specified.
- **CATLG** specifies that the system keeps the file after the Process step completes and that an entry gets placed in the catalog. Catalog is the default.

IOEXIT = exit-name | (exit-name[,parameter,...])
indicates that a user-written program is to be called to perform I/O requests for the associated data.

exit-name specifies the name of the user-written program to be given control for I/O-related requests for the associated data.

parameter specifies a parameter, or list of parameters, to be passed to the specified exit. For valid parameter formats, refer to the RUN TASK parameters for the appropriate platform.

LABEL = ([file-sequence-number]
,[SL | AL | BLP | LTM | NL]
,[PASSWORD | NOPWREAD]
,[IN | OUT]
,[RETPD = nnnn | EXPDT = yyddd | yyyyddd])
specifies label information for the tape.

file-sequence-number specifies the relative file position on the tape.

The label type is designated as follows:

- **SL** specifies IBM standard labels.
- **AL** specifies American National Standard labels.
- **BLP** specifies bypass label processing.
- **LTM** specifies bypass leading tape marks.
- **NL** specifies no labels.
- **PASSWORD** specifies that a password must be supplied by the operator or user before the data set can be accessed.
- **NOPWREAD** specifies that a password is not required to read the data set.
- **IN** specifies that a BSAM data set opened for INOUT or a BDAM data set opened for UPDAT is to be read only.
- **OUT** specifies that a BSAM data set opened for OUTIN or OUTINX is to be write only.
- **RETPD** specifies the retention period for the data set in days, where nnnn is 1-4 digits.
- **EXPDT** specifies the expiration date for the data set, where yyddd or yyyy/ddd is a valid Julian date.

(FROM) LIBR=(
EXCLMEM=(generic|mem|start-range/stop-range|list)
EXCLSLIB=(generic|sublib|start-range/stop-range|list)]]
[EXCLTYPE=(generic|type|start-range/stop-range|list)]]
[EXCLMEM=(generic|mem|start-range/stop-range|list)]
[REPLACE=YES|NO]
[SELMEM=member|generic|*|(member, [new-name],[NR|R])|
(generic,,[NR|R])|(start-range/stop-range,,[NR|R])|(list)]
[SELSLIB=sublibrary|generic|*| (sublibrary, [new-name],[NR|R])|
(generic,,[NR|R])|(start-range/stop-range,,[NR|R])|(list)]

[SELTYPE=type|generic|*|
(type, new-name)|(start-range/stop-range)|(list)] [*]
)

specifies criteria that identifies the members, sublibraries or types that are not to be copied. The EXCLxxxx parameter can be specified only in the FROM clause of the COPY statement. EXCLxxxx allows the user to make exceptions to the members, sublibraries or types specified generically, or by range in the SELxxxx option. See page 18 for the override priority for the SELxxxx and EXCLxxxx parameters.

- **generic** specifies a generic member, sublibrary or type name. For example, if EXCLMEM=CDM* is specified, all member names beginning with CDM are excluded. The only way to override an excluded generic is to specify an individual member, sublibrary or type in the SELMEM, SELSLIB, or SELTYPE parameter.
- **member|sublibrary|type** specifies an individual member, sublibrary or type name. When used in the EXCLxxxx parameter, its exclusion cannot be overridden.
- **start-range** specifies the first name in an alphanumeric range of members, sublibraries or types. Although the specification is treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used with the EXCLxxxx parameter, the first and last names, as well as all names between are not copied.
- **stop-range** specifies the last name in an alphanumeric range of members, sublibraries or types. Although the specification is treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used with the EXCLxxxx parameter, the first and last names, as well as all names between are not copied.

  Note: The only way to override an excluded range is to specify an individual member, sublibrary or type name in the SELxxxx parameter.

- **list** specifies a list of member, sublibrary or type names.
- ***** specifies that all Connect:Direct VSE Librarian defaults are to be taken.

EXCLSLIB=(generic|sublib|start-range/stop-range|list) specifies criteria that identifies the members, sublibraries or types that are not to be copied. The EXCLxxxx parameter can be specified only in the FROM clause of the COPY statement. EXCLxxxx allows the user to make exceptions to the members, sublibraries or types specified generically, or by range in the SELxxxx option. See page 18 for the override priority for the SELxxxx and EXCLxxxx parameters.

- **generic** specifies a generic member, sublibrary or type name. For example, if EXCLMEM=CDM* is specified, all member names beginning with CDM are excluded. The only way to override an excluded generic is to specify an individual member, sublibrary or type in the SELMEM, SELSLIB, or SELTYPE parameter.

- **member|sublibrary|type** specifies an individual member, sublibrary or type name. When used in the EXCLxxxx parameter, its exclusion cannot be overridden.
- **start-range** specifies the first name in an alphanumeric range of members, sublibraries or types. Although the specification is treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used with the EXCLxxxx parameter, the first and last names, as well as all names between are not copied.
- **stop-range** specifies the last name in an alphanumeric range of members, sublibraries or types. Although the specification is treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used with the EXCLxxxx parameter, the first and last names, as well as all names between are not copied.

Note: The only way to override an excluded range is to specify an individual member, sublibrary or type name in the SELxxxx parameter.

- **list** specifies a list of member, sublibrary or type names.
- ***** specifies that all Connect:Direct VSE Librarian defaults are to be taken.

EXCLTYPE=(generic|type|start-range/stop-range|list) specifies criteria that identifies the members, sublibraries or types that are not to be copied. The EXCLxxxx parameter can be specified only in the FROM clause of the COPY statement. EXCLxxxx allows the user to make exceptions to the members, sublibraries or types specified generically, or by range in the SELxxxx option. See page 18 for the override priority for the SELxxxx and EXCLxxxx parameters.

- **generic** specifies a generic member, sublibrary or type name. For example, if EXCLMEM=CDM* is specified, all member names beginning with CDM are excluded. The only way to override an excluded generic is to specify an individual member, sublibrary or type in the SELMEM, SELSLIB, or SELTYPE parameter.
- **member|sublibrary|type** specifies an individual member, sublibrary or type name. When used in the EXCLxxxx parameter, its exclusion cannot be overridden.
- **start-range** specifies the first name in an alphanumeric range of members, sublibraries or types. Although the specification is treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used with the EXCLxxxx parameter, the first and last names, as well as all names between are not copied.
- **stop-range** specifies the last name in an alphanumeric range of members, sublibraries or types. Although the specification is treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and

the last (stop-range) names. When used with the EXCLxxxx parameter, the first and last names, as well as all names between are not copied.

Note: The only way to override an excluded range is to specify an individual member, sublibrary or type name in the SELxxxx parameter.

- **list** specifies a list of member, sublibrary or type names.
- ***** specifies that all Connect:Direct VSE Librarian defaults are to be taken.

REPLACE=YES|NO specifies whether or not the member with the same name will be replaced in the target library or PDS. REPLACE= is allowed here because the REPLACE keyword is specified on the FROM side for an OS/390 PDS transfer.

SELMEM=member|generic|*|
(member, [new-name],[NR|R])|
(generic,,[NR|R])|
(start-range/stop-range,,[NR|R]|
(list)

specifies the selection criteria by which VSE Librarian members are to be copied. See page 18 for the override priority for the SELxxxx and EXCLxxxx parameters.

Various specifications can be combined in a list after the SELxxxx parameter.

If SELxxxx is specified and EXCLxxxx for the same MEM, SLIB or TYPE is not specified, all MEM, SLIB or TYPE are selected.

- **generic** specifies a generic member, sublibrary or type. If CDM* is specified as either a parameter or a subparameter, all members, sublibraries or types beginning with CDM are selected for copying.
- **(*)** represents a global generic. A global generic indicates that all members, sublibraries or types of the library are to be included.
- When a generic is specified its selection can be overridden with any type of specification in the appropriate EXCLxxxx parameter.
- When using the generic and specifying NR or R, the second positional parameter (NEWNAME) must be null.
- **member|sublibrary|type** specifies an individual member, sublibrary, or type. The only way to override a selection by member, sublibrary or type is to specify that member, sublibrary or type in the appropriate EXCLxxxx parameter.
- **newname** specifies a new name for the member, sublibrary or type. The NEWNAME parameter must be null if a generic name or range is used in the first subparameter position.
- **NR** specifies that the member or sublibrary does not replace an existing member or sublibrary of the same name at the receiving VSE Library or PDS. NR

overrides the REPLACE or REPLACE= parameter. R is the default. NR may not be specified on SELTYPE.

Note: REPLACE=NO (or NOREPLACE) applies to an entire VSE Library or PDS as opposed to NR, which applies to the member or sublibrary.

- **R** specifies that the member or sublibrary replaces an existing member or sublibrary at the receiving VSE Library or PDS. R overrides the REPLACE=NO or NOREPLACE parameter. R may not be specified on SELTYPE.
- When used with NEWNAME, R applies to the NEWNAME and not to the original name. When used with a generic name or with a range, R applies to all members selected for that criteria.
- **start-range** specifies the first name in an alphanumeric range of members, sublibraries or types. Although member, sublibrary or type names are treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used, the first and last member, sublibrary or type specified in the range as well all members, sublibraries or types between, are copied.
- **stop-range** specifies the last name in an alphanumeric range of members, sublibraries or types. Although member, sublibrary or type names are treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used, the first and last member, sublibrary or type specified in the range as well all members, sublibraries or types between, are copied.
- When a range in the SELxxxx parameter is specified, its selection can be overridden with any type of specification in the appropriate EXCLxxxx parameter.
- The second positional parameter (NEWNAME) of the SELxxxx must be null when using a range and specifying NR or R.
- **list** specifies a list of selected members.

SELSLIB=sublibrary|generic|*|
(sublibrary, [new-name],[NR|R])|
(generic,,[NR|R])|
(start-range/stop-range,,[NR|R]|
(list)

specifies the selection criteria by which VSE Librarian members are to be copied. See page 18 for the override priority for the SELxxxx and EXCLxxxx parameters.

Various specifications can be combined in a list after the SELxxxx parameter.

If SELxxxx is specified and EXCLxxxx for the same MEM, SLIB or TYPE is not specified, all MEM, SLIB or TYPE are selected.

- **generic** specifies a generic member, sublibrary or type. If CDM* is specified as either a parameter or a subparameter, all members, sublibraries or types beginning with CDM are selected for copying.

- **(*)** represents a global generic. A global generic indicates that all members, sublibraries or types of the library are to be included.
- When a generic is specified its selection can be overridden with any type of specification in the appropriate EXCLxxxx parameter.
- When using the generic and specifying NR or R, the second positional parameter (NEWNAME) must be null.
- **member|sublibrary|type** specifies an individual member, sublibrary, or type. The only way to override a selection by member, sublibrary or type is to specify that member, sublibrary or type in the appropriate EXCLxxxx parameter.
- **newname** specifies a new name for the member, sublibrary or type. The NEWNAME parameter must be null if a generic name or range is used in the first subparameter position.
- **NR** specifies that the member or sublibrary does not replace an existing member or sublibrary of the same name at the receiving VSE Library or PDS. NR overrides the REPLACE or REPLACE= parameter. **R** is the default. **NR** may not be specified on SELTYPE.

  Note: REPLACE=NO (or NOREPLACE) applies to an entire VSE Library or PDS as opposed to **NR**, which applies to the member or sublibrary.

- **R** specifies that the member or sublibrary replaces an existing member or sublibrary at the receiving VSE Library or PDS. **R** overrides the REPLACE=NO or NOREPLACE parameter. **R** may not be specified on SELTYPE.
- When used with NEWNAME, **R** applies to the NEWNAME and not to the original name. When used with a generic name or with a range, **R** applies to all members selected for that criteria.
- **start-range** specifies the first name in an alphanumeric range of members, sublibraries or types. Although member, sublibrary or type names are treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used, the first and last member, sublibrary or type specified in the range as well all members, sublibraries or types between, are copied.
- **stop-range** specifies the last name in an alphanumeric range of members, sublibraries or types. Although member, sublibrary or type names are treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used, the first and last member, sublibrary or type specified in the range as well all members, sublibraries or types between, are copied.
- When a range in the SELxxxx parameter is specified, its selection can be overridden with any type of specification in the appropriate EXCLxxxx parameter.
- The second positional parameter (NEWNAME) of the SELxxxx must be null when using a range and specifying **NR** or **R**.
- **list** specifies a list of selected members.

SELTYPE=type|generic|(*)|
(type, new-name)|
(start-range/stop-range)|
(list)

specifies the selection criteria by which VSE Librarian members are to be copied. See page 18 for the override priority for the SELxxxx and EXCLxxxx parameters.

Various specifications can be combined in a list after the SELxxxx parameter.

If SELxxxx is specified and EXCLxxxx for the same MEM, SLIB or TYPE is not specified, all MEM, SLIB or TYPE are selected.

- **generic** specifies a generic member, sublibrary or type. If CDM* is specified as either a parameter or a subparameter, all members, sublibraries or types beginning with CDM are selected for copying.
- **(*)** represents a global generic. A global generic indicates that all members, sublibraries or types of the library are to be included.
- When a generic is specified its selection can be overridden with any type of specification in the appropriate EXCLxxxx parameter.
- When using the generic and specifying NR or R, the second positional parameter (NEWNAME) must be null.
- **member|sublibrary|type** specifies an individual member, sublibrary, or type. The only way to override a selection by member, sublibrary or type is to specify that member, sublibrary or type in the appropriate EXCLxxxx parameter.
- **newname** specifies a new name for the member, sublibrary or type. The NEWNAME parameter must be null if a generic name or range is used in the first subparameter position.
- **start-range** specifies the first name in an alphanumeric range of members, sublibraries or types. Although member, sublibrary or type names are treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used, the first and last member, sublibrary or type specified in the range as well all members, sublibraries or types between, are copied.
- **stop-range** specifies the last name in an alphanumeric range of members, sublibraries or types. Although member, sublibrary or type names are treated as generics, they cannot be used with an asterisk (*). A slash separates the first (start-range) and the last (stop-range) names. When used, the first and last member, sublibrary or type specified in the range as well all members, sublibraries or types between, are copied.
- When a range in the SELxxxx parameter is specified, its selection can be overridden with any type of specification in the appropriate EXCLxxxx parameter.
- **list** specifies a list of selected members.

(TO) LIBR=(
[DATA=NO|YES]
[LOCKID=lockid]
[MSHP=NO|YES|OVERRIDE]
[REPLACE=YES|NO]
[RESETLOCK=NO|YES]
[REUSE=AUTOMATIC|IMMEDIATE]
[SLIBDISP=SHR|OLD|NEW|RPL]
[SUBLIB=sublibrary]
[TYPE=type]
[*]
)

specifies how the members being copied will be catalogued.

DATA=NO|YES specifies whether or not the member(s) being cataloged are to have the DATA=YES flag turned on. This flag is only meaningful for members of the type PROC.

LOCKID=lockid specifies whether or not the member(s) being cataloged are to be locked using the specified lockid. If not specified, the member(s) are not locked unless the source member is also a VSE Library member and it was originally locked.

MSHP=NO|YES|OVERRIDE specifies whether or not the member(s) being cataloged are to be flagged as under MSHP control. MSHP=OVERRIDE specifies that the member(s) are to be flagged as under MSHP control and to have the MSHP Bypass flag turned on. If the source member is from a VSE Library, the flags on the source member are copied to the new member.

REPLACE=YES|NO specifies whether the member should be replaced if it currently exists in the target VSE Library.

RESETLOCK=NO|YES specifies that an existing VSE Library can be replaced even if contains members that are locked.

REUSE=AUTOMATIC|IMMEDIATE specifies how the space attribute is to be set when a sublibrary is defined.

SLIBDISP=SHR|OLD|NEW|RPL|MOD specifies the disposition of the sublibrary for the member being cataloged. Specify SLIBDISP=RPL if you want the sublibrary defined as empty even if it currently exists. Specify SLIBDISP=MOD if you want the sublibrary defined only if it doesn't currently exist and you do not want the current members deleted if the sublibrary does exist.

SUBLIB=sublibrary specifies the target sublibrary for the member(s) being cataloged. This parameter is required when the source is not a VSE Library.

TYPE=type specifies the target member type for the member(s) being cataloged. This parameter is required when the source is not a VSE Library.

***** specifies that the Connect:Direct VSE Library defaults are to be taken from the source VSE Library members. This parameter is only valid when the source is from a VSE Library.

(FROM) LST=(
[CLASS=class]
[DISP=D|K|H|L]
[PWD=password]
)

specifies how the LST queue entry will be handled after processing.

CLASS=class specifies the class (A-Z, 0-9) of the LST or PUN queue entry to be accessed. This parameter is required.

DISP=D|K|H|L specifies how you want the disposition that is to be assumed when the LST or PUN queue entry is processed. Specifying DISP= overrides the actual disposition of the LST or PUN queue entry. When the copy is complete, the queue entry will be deleted if DISP=D is specified. If DISP=K is specified, the disposition will be changed to L. Specifying DISP=H|L causes the disposition to be set to H|L after the queue entry is processed.

PWD=password specifies the password of the queue entry. If PWD= is not specified, Connect:Direct VSE uses binary zeros for the password unless overridden at initialization time by the POWER.MPWD parameter.

(TO) LST=(
[BANNER=(literal1=name1, literal2=name2,...)]
[BLDG=building]
[BURST=YES|NO]
[CC=M|ASA|(NOCC,[TOF=1|char|x'xx'], [LINECT=55|nn])]
[CHARS=(tablename,tablename)]
[CICSDATA=CICS-data]
[CKPTLINE=nnnnn]
[CKPTPAGE=nnnnn]
[CKPTSEC=nnnnn]
[CLASS=class]
[COMPACT=compaction-table-name]
[CONTROL=program
single
double
triple]
[COPIES=nnn]
[COPY=nnn]
[DEFAULT=YES|NO]
[DEPT='department-identification']
[DEST=nodename (nodename,userid)]
[DFLT=YES|NO]
[DISP=D|K|H|L]
[DIST=distribution]
[FCB=fcb-name]
[FLASH=(overlay-name,count)]
[FORMDEF=membername]
[FORMS=form-name]
[FNO=form-name]
[HOLD=YES|NO]
[JSEP=nnnnn]
[MODIFY=module-name]
[PAGEDEF=membername]
[PRI=n]
[PRMODE=process-mode]

[PROGR='programmer-name']
[PRTY=n]
[PWD=password]
[ROOM='room-identification']
[SUBNAME=submitter's-name]
[SYSID=n]
[THRESHLD=nnnnnnnn]
[TRC=YES|NO]
[UCS=character-set-name (character-set-name,FC)]
[USER='user-data-description']
[USERID=userid]
[WRITER=writer-name]
)

specifies how the LST queue entry will be printed. Various combinations can be specified after the LST parameter.

References to the VSE/POWER * $$ JOB, * $$ LST, or * $$ PUN statements refer to the *IBM VSE/POWER Administration and Operation Manual* SC33-6633.

BANNER=(literal1=name1, literal2=name2,...) specifies that a Connect:Direct banner page is to be printed at the beginning of the LST queue entry. Banner values are supplied in pairs of literal=value. One banner line is printed for each supplied pair. The size of the literal can be 1-25 characters and the size of the value can be 1-30 characters. Literals or values that exceed the maximum are truncated to the maximum.

```
literal1/name1,
literal2/name2,...
```

The value of literal can be EITHER a LITERAL or a SYMBOLIC parameter. The value of name can also be EITHER a LITERAL or a SYMBOLIC parameter.

The following example causes a banner page to be printed with three lines of user supplied information:

```
BANNER=(PROGRAMMER=&PROGR,JOBNAME=&JOBNM,
                  &SUBMITTER=JDOE1)
```

The banner page generated would look like this:

```
****************** C O N N E C T : D I R E C T ******************
****************** C O N N E C T : D I R E C T ******************
****************** C O N N E C T : D I R E C T ******************
****************** C O N N E C T : D I R E C T ******************
****   PROGRAMMER                JOHN DOE                      ****
****   JOBNAME                   JOBNAME1                      ****
****   SUBMITTER                 JDOE1                         ****
****************** C O N N E C T : D I R E C T ******************
****************** C O N N E C T : D I R E C T ******************
****************** C O N N E C T : D I R E C T ******************
****************** C O N N E C T : D I R E C T ******************
```

BLDG=building specifies the building identification for this LST or PUN queue entry. See VSE/POWER $$JOB statement of the *IBM VSE/POWER Administration and Operation Manual.*

BURST=YES|NO specifies the 3800 specification for the burster/trimmer/stacker for this LST queue entry. See VSE/POWER LST statement of the *IBM VSE/POWER Administration and Operation Manual*.

CC=M|ASA|(NOCC[,TOF=1|char|x'xx'],[LINECT=55|nn]) specifies what type of control characters are provided with the data. If NOCC is specified, the data is placed in the VSE/POWER LST or PUN queue with ASA control characters, specifying the TOF= character every LINECT= lines of data.

CHARS=(tablename,tablename) specifies the 3800 character arrangement table names. See the VSE/POWER LST statement in *IBM VSE/POWER Administration and Operation Manual*.

CICSDATA=CICS-data specifies the installation CICS data. This is normally used for resource level checking by the Report Controller Feature of CICS/VSE.

CKPTLINE=nnnnn
CKPTPAGE=nnnnn
CKPTSEC=nnnnn specifies the OS/390 checkpoint information. VSE/POWER does not use this data, but if the LST or PUN queue entry will be transmitted to JES2 or JES3 via PNET, then you can specify the values to be passed to JES2 or JES3. The values are documented in the OS/390 JCL manual.

CLASS=class specifies the class of the LST or PUN queue entry. Valid classes are A-Z, 0-9. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

COMPACT=compaction-table-name specifies the name of the compaction table to be used if the LST or PUN queue entry is sent by VSE/POWER RJE to an RJE terminal. See the VSE/POWER or PUN LST statement in *IBM VSE/POWER Administration and Operation Manual*.
CONTROL=program
single
double
triple specifies the OS/390 control information. The specification is not used by VSE/POWER, but can be specified if the LST or PUN queue entry will be transmitted to JES2 or JES3 via PNET. The parameter is documented in the OS/390 JCL manual.

COPIES=nnn|COPY=nnn specifies the number of copies to be printed or punched for the LST or PUN queue entry See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

DEFAULT=YES|NO specifies whether or not the 3800 default values will be used from the installation SETDF. See the VSE/POWER or PUN LST statement in *IBM VSE/POWER Administration and Operation Manual*.

DEPT='department-identification' specifies the department identification for the LST or PUN queue entry. See the VSE/POWER JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

DEST=nodename (nodename,userid) specifies the nodename and/or the userid to be associated with this LST or PUN queue entry. Specifying the nodename causes the

LST or PUN queue entry to be placed in the XMT queue. See the VSE/POWER or PUN LST statement in *IBM VSE/POWER Administration and Operation Manual*.

DFLT=YES|NO specifies whether or not the 3800 default values will be used from the installation SETDF. See the VSE/POWER LST statement in *IBM VSE/POWER Administration and Operation Manual*.

DISP=D|K|H|L specifies the VSE/POWER disposition that will be used for the LST or PUN queue entry.

DIST=distribution specifies distribution code for the LST or PUN queue entry. This information is printed by VSE/POWER on the VSE/POWER separator page.

FCB=fcb-name specifies the name of the FCB phase to be used when the LST queue entry is physically printed. See the VSE/POWER PUN orLST statement in *IBM VSE/POWER Administration and Operation Manual*.

FLASH=(overlay-name,count) specifies the 3800 printer flash overlay name and the number of pages to be flashed. See the VSE/POWER or PUN LST statement in *IBM VSE/POWER Administration and Operation Manual*.

FORMDEF=membername specifies the PSF form definition member name for the LST queue entry.

FORMS=form-name|FNO=form-name specifies the forms name for the LST or PUN queue entry. See the VSE/POWER or PUN LST statement in *IBM VSE/POWER Administration and Operation Manual*.

HOLD=YES|NO specifies the VSE/POWER disposition for the LST or PUN queue entry. This keyword is provided for compatibility with OS/390. See DISP= keyword to specify all VSE/POWER dispositions in *IBM VSE/POWER Administration and Operation Manual*.

JSEP=nnnnn specifies the number of separator pages/cards that should be produced by VSE/POWER when the LST or PUN queue entry is physically output to the printer or card punch.

MODIFY=module-name specifies the set of predefined data to be printed on each page for the 3800 LST queue entry. See the VSE/POWER LST statement in *IBM VSE/POWER Administration and Operation Manual*.

PAGEDEF=membername specifies the PSF page definition for the LST queue entry.

PRI=n specifies the priority for the LST or PUN queue entry. Valid VSE/POWER priorities are 0-9. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

PRMODE=process-mode specifies the printer processing mode. VSE/POWER does not use this value, but it can be specified if the LST queue entry is to be transmitted through PNET to JES2 or JES3.

PROGR='programmer-name' specifies the programmer name for the LST or PUN queue entry. See the VSE/POWER JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

PRTY=n specifies the priority for the LST or PUN queue entry. Valid VSE/POWER priorities
are 0-9. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

PWD=password specifies the password that will be associated with the LST or PUN queue entry. If not specified, the master password (or binary zeros) will be used as the password. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

ROOM='room-identification' specifies the room identification for the LST or PUN queue entry. See the VSE/POWER JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

SUBNAME=submitter's-name specifies the submitter's name for the LST or PUN queue entry. If not specified, then 'SCDIRECT' is used as the submitter's name. See the VSE/POWER JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

SYSID=n specifies the system id for a VSE/POWER Shared Spool environment for this LST or PUN queue entry. See the VSE/POWER PUN or LST statement in *IBM VSE/POWER Administration and Operation Manual*.

THRESHLD=nnnnnnnn specifies the threshold value. This value is not used by VSE/POWER but can be specified if the LST or PUN queue entry it to be transmitted to JES2 or JES3 via PNET.

TRC=YES|NO specifies whether or not translate reference characters are present in the data. This is for 3800 print output only. See the VSE/POWER LST statement in *IBM VSE/POWER Administration and Operation Manual*.

UCS=character-set-name (character-set-name,FC) specifies the name of the UCS buffer and optionally whether FOLD and block data-check should be specified. See the VSE/POWER PUN or LST statement in *IBM VSE/POWER Administration and Operation Manual*.

USER='user-data-description' specifies up to 16-bytes of user data for the LST or PUN queue entry. See the VSE/POWER PUN or LST statement in *IBM VSE/POWER Administration and Operation Manual*.

USERID=userid specifies the userid to be associated with this LST or PUN queue entry. Specifying the nodename causes the LST or PUN queue entry to be placed in the XMT queue. See the VSE/POWER PUN or LST statement in *IBM VSE/POWER Administration and Operation Manual*.

WRITER=writer-name specifies the writer name for the LST or PUN queue entry. VSE/POWER does not use the write name, but it can be specified if the LST or PUN queue entry is to be transmitted to JES or JES3 through PNET.

PNODE
is the primary node, defining the direction of transfer (with SNODE). When PNODE is coded with the FROM parameter, a send takes place. When PNODE is coded with

the TO parameter, a receive takes place. PNODE is the default for the FROM parameter.

(FROM) PUN=(
[CLASS=class]
[DISP=D|K|H|L]
[PWD=password]
)

specifies how the PUN queue entry will be handled after processing.

CLASS=class specifies the class (A-Z, 0-9) of the LST or PUN queue entry to be accessed. This parameter is required.

DISP=D|K|H|L specifies how you want the disposition that is to be assumed when the LST or PUN queue entry is processed. Specifying DISP= overrides the actual disposition of the LST or PUN queue entry. When the copy is complete, the queue entry will be deleted if DISP=D is specified. If DISP=K is specified, the disposition will be changed to L. Specifying DISP=H|L causes the disposition to be set to H|L after the queue entry is processed.

PWD=password specifies the password of the queue entry. If PWD= is not specified, Connect:Direct VSE uses binary zeros for the password unless overridden at initialization time by the POWER.MPWD parameter.

(TO) PUN=(
[BLDG=building]
[CC=M|ASA|NOCC]
[CICSDATA=CICS-data]
[CKPTLINE=nnnnn]
[CKPTPAGE=nnnnn]
[CKPTSEC=nnnnn]
[CLASS=class]
[CONTROL=program
single
double
triple]
[COPIES=nnn]
[COPY=nnn]
[DEPT='department-identification']
[DEST=nodename (nodename,userid)]
[DISP=D|K|H|L]
[DIST=distribution]
[FORMS=form-name]
[FNO=form-name]
[HOLD=YES|NO]
[JSEP=nnnnn]
[PRI=n]
[PROGR='programmer-name']
[PRMODE=process-mode]
[PRTY=n]
[PWD=password]
[ROOM='room-identification']
[SUBNAME=submitter's-name]
[SYSID=n]
[THRESHLD=nnnnnnnn]

[USER='user-data-description']
[USERID=userid]
[WRITER=writer-name]
)

specifies how the PUN queue entry will be processed. Various combinations can be specified after the PUN paramter.

References to the VSE/POWER * $$ JOB, * $$ LST, or * $$ PUN statements refer to the *IBM VSE/POWER Administration and Operation Manual* SC33-6633.

BLDG=building specifies the building identification for this LST or PUN queue entry. See VSE/POWER $$JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

CC=M|ASA|NOCC specifies what type of control characters are provided with the data. If NOCC is specified, the data is placed in the VSE/POWER LST or PUN queue with ASA control characters.

CICSDATA=CICS-data specifies the installation CICS data. This is normally used for resource level checking by the Report Controller Feature of CICS/VSE.

CKPTLINE=nnnnn
CKPTPAGE=nnnnn
CKPTSEC=nnnnn specifies the OS/390 checkpoint information. VSE/POWER does not use this data, but if the LST or PUN queue entry will be transmitted to JES2 or JES3 via PNET, then you may specify the values to be passed to JES2 or JES3. The values are documented in the OS/390 JCL manual.

CLASS=class specifies the class of the LST or PUN queue entry. Valid classes are A-Z, 0-9. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

CONTROL=program
single
double
triple specifies the OS/390 control information. The specification is not used by VSE/POWER, but can be specified if the LST or PUN queue entry will be transmitted to JES2 or JES3 via PNET. The parameter is documented in the OS/390 JCL manual.

COPIES=nnn|COPY=nnn specifies the number of copies to be printed or punched for the LST or PUN queue entry. See the VSE/POWER

DEPT='department-identification' specifies the department identification for the LST or PUN queue entry. See the VSE/POWER $$JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

DEST=nodename (nodename,userid) specifies the nodename and/or the userid to be associated with this LST or PUN queue entry. Specifying the nodename causes the LST or PUN queue entry to be placed in the XMT queue. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

DISP=D|K|H|L specifies the VSE/POWER disposition that will be used for the LST or PUN queue entry.

DIST=distribution specifies distribution code for the LST or PUN queue entry. This information is printed by VSE/POWER on the VSE/POWER separator page.

FORMS=form-name
FNO=form-name specifies the forms name for the LST or PUN queue entry. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

HOLD=YES|NO specifies the VSE/POWER disposition for the LST or PUN queue entry. This keyword is provided for compatibility with OS/390. See DISP= keyword to specify all VSE/POWER dispositions.

JSEP=nnnnn specifies the number of separator pages/cards that should be produced by VSE/POWER when the LST or PUN queue entry is physically output to the printer or card punch.

PRI=n specifies the priority for the LST or PUN queue entry. Valid VSE/POWER priorities are 0-9. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

PRMODE=process-mode specifies the printer processing mode. VSE/POWER does not use this value, but it can be specified if the LST queue entry is to be transmitted via PNET to JES2 or JES3.

PROGR='programmer-name' specifies the programmer name for the LST or PUN queue entry. See the VSE/POWER $$JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

PRTY=n specifies the priority for the LST or PUN queue entry. Valid VSE/POWER priorities are 0-9. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

PWD=password specifies the password that will be associated with the LST or PUN queue entry. If not specified, the master password (or binary zeros) will be used as the password. See the VSE/POWER LST or PUN statement.

ROOM='room-identification' specifies the room identification for the LST or PUN queue entry. See the VSE/POWER $$JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

SUBNAME=submitter's-name specifies the submitter's name for the LST or PUN queue entry. If not specified, then 'SCDIRECT' is used as the submitter's name. See the VSE/POWER $$JOB statement in *IBM VSE/POWER Administration and Operation Manual*.

SYSID=n specifies the system id for a VSE/POWER Shared Spool environment for this LST or PUN queue entry. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

THRESHLD=nnnnnnnn specifies the threshold value. This value is not used by VSE/POWER but can be specified if the LST or PUN queue entry it to be transmitted to JES2 or JES3 via PNET.

USER='user-data-description' specifies up to 16-bytes of user data for the LST or PUN queue entry. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

USERID=userid specifies the userid to be associated with this LST or PUN queue entry. Specifying the nodename causes the LST or PUN queue entry to be placed in the XMT queue. See the VSE/POWER LST or PUN statement in *IBM VSE/POWER Administration and Operation Manual*.

WRITER=writer-name specifies the writer name for the LST or PUN queue entry. VSE/POWER does not use the write name, but it can be specified if the LST or PUN queue entry is to be transmitted to JES or JES3 via PNET.

SNODE
is the secondary node, defining the direction of transfer (with PNODE). When SNODE is coded with the FROM parameter, a receive takes place. When SNODE is coded with the TO parameter, a send takes place. SNODE is the default for the TO parameter.

Either PNODE or SNODE can be coded; the other parameter defaults to the complement of the one coded. If you do not code either parameter, the file is sent from the PNODE to the SNODE.

SPACE=[(start-track | start-block |,(allocation))]
[(trigger-key,(allocation))]
[(record-size,(primary-allocation,secondary-allocation))]
specifies the amount of DASD storage to be allocated for new files on the destination node. SPACE must be specified for all new non-VSAM (ESDS, KSDS, or RRDS) explicitly defined files unless you specified a typekey record that includes a SPACE parameter.

The SPACE parameter has three formats:

- Noncontrolled BSAM files
- START TRACK-1 controlled BSAM data sets
- VSAM-managed SAM (MSAM) files

For noncontrolled BSAM files:

start-track designates the file's starting track number on a CKD or ECKD disk device.

start-block designates the file's starting block number when allocating on a FBA disk device.

allocation specifies the primary allocation of storage, either in tracks or blocks.

For CA-DYNAM/D or CA-EPIC controlled files:

trigger-key designates the start-track number trigger that is defined in your system catalog. In most cases this value is **1**.

allocation specifies the primary allocation of storage in either tracks or blocks.

When you are using a disk management system such as CA-DYNAM/D or CA-EPIC and you specify an allocation trigger value, code the UNIT=DNOASGN in your

Process. See the UNIT parameter on page 170 for additional information about specifying this parameter.

When these parameters are coded, CA-DYNAM/D or CA-EPIC perform both primary and optional secondary data set extent allocation for Connect:Direct and the output data set.

For VSAM Managed SAM (MSAM) files:

record-size specifies the logical record length of the output data set record.

primary-allocation specifies the initial amount of records that are to be initially allocated to the data set.

secondary-allocation specifies the secondary amount of records to be allocated in extents 2 - 15.

Connect:Direct VSE performs this allocation only for VSAM-managed SAM (MSAM) files. VSAM performs this allocation for VSAM-controlled data sets (ESDS, KSDS or RRDS) if the DEFINE CLUSTER specified a secondary extent allocation amount. CA-DYNAM/D or CA-EPIC will perform this allocation if secondary allocation is specified for that data set in the respective system catalog.

SYSOPTS = “DBCS=(tablename,so,si,PAD|PAD=pc)”“parameter1[,parameter2...]”
specifies system operation parameters.

DBCS=(tablename,so,si,PAD|PAD=pc) is used to invoke the double-byte character set translation facility. File transfer with double-byte character set (DBCS) is only supported in record mode. It is not supported in block mode.

Note: Transfers attempted in block mode can produce unpredictable results in the destination file. These results can compromise data integrity. You will not receive an error message.

- **tablename** is the name of the requested DBCS translation table. The tablename is required with DBCS. If you only specify **tablename**, you do not need to enclose the parameters in parentheses.

  The following tables are provided by Connect:Direct

  - **EBCXKSC** translates data from host EBCDIC to ASCII KS5601.
  - **KSCXEBC** translates data from ASCII KS5601 to host EBCDIC.
  - **EBCXKPC** translates data from host EBCDIC to DBCS-PC Korean.
  - **KPCXEBC** translates data from DBCS-PC Korean to host EBCDIC.
  - **NHCXBG5** translates data from Chinese new host code to Chinese Big5.
  - **BG5XNHC** translates data from Chinese Big5 to Chinese new host code.
  - **NHCXC55** translates data from Chinese new host code to Chinese 5550.
  - **C55XNHC** translates data from Chinese 5550 to Chinese new host code.

- **so** is the SHIFT-OUT character denoting a shift from single-byte character set (SBCS) to double-byte character set (DBCS) mode. The default is the IBM standard x’0E’.

- **si** is the SHIFT-IN character denoting a shift from DBCS to SBCS mode. The default is the IBM standard x'0F'.

> **Note:** NOSO indicates no shift-out or shift-in character and is denoted by the use of x'00' for the SO and SI characters. NOSO is used when the data is not in mixed form and is assumed to contain all DBCS characters.

- **PAD|PAD=pc** specifies that padding characters are in use. When DBCS data is translated from EBCDIC to ASCII, **PAD** specifies that the SHIFT-OUT and SHIFT-IN characters will be replaced by a pad character. This allows the displacement of fields within a record to remain unchanged during translation.
- When DBCS data is translated from ASCII to EBCDIC, **PAD** specifies that the input ASCII DBCS file is in a padded format. The character immediately preceding a DBCS character or string will be overlaid by the SHIFT-OUT character. The character immediately following a DBCS character or string will be overlaid with the SHIFT-IN character. This allows the displacement of fields within a record to remain unchanged during translation.
- **pc** is the pad character to be used during EBCDIC to ASCII translation. **pc** is ignored for ASCII to EBCDIC translations. The default value for **pc** is x'00'.

parameter1[**,parameter2**...] is used in conjunction with the IOEXIT parameter. It specifies the parameters to be passed to the I/O exit for copies from a non-370 node to a Connect:Direct VSE node.

TYPE = typekey

specifies the entry name of the type defaults file containing the default file attributes used to allocate the destination file. This typekey is specified only when defaults are requested by the user.

For VSE to OpenVMS copies: If the typekey exceeds eight characters, the typekey must be entered in the SYSOPTS parameter on the TO clause of the Connect:Direct OpenVMS COPY statement.

For OpenVMS to VSE copies: The typekey must not be greater than eight characters.

UNIT=([group-name | device-type | unit-address | DLBLONLY | DNOASGN | TNOASGN])

specifies the group-name, device-type, or unit-address where the file resides or will reside. For BSAM-to-BSAM copies where the destination file is NEW and the UNIT parameter is not coded on the TO parameter, the device type from the source (FROM) is used.

Without Disk or Tape Management System

If you are not using a supported disk or tape management system, you can code the UNIT parameter to meet the amount of flexibility and device independence that you require.

UNIT=group-name specifies the input or output device for your Processes.

Use the following table to select the supported Connect:Direct group-name for either the input or output device on your Processes.

For device independence and greatest flexibility, code your Processes with the UNIT=group-name.

UNIT=device-type specifies the type of device for your Processes.

Use the following table to select an appropriate VSE device type. If you do not find your specific device-type, replace the device-type with the appropriate group-name. For example, if you are creating output files on a 93xx FBA device, you would code UNIT=FBA on your Process statement.

UNIT=unit-address specifies the address of the specific device.

If you require that a data set be allocated on a specific device every time the Process is executed, code your Process with UNIT=CUU and replace the CUU with the Channel/Unit address of the device.

Group-Name	Device-Type
DISK	2311 2314 3310 3330 3340 3350 3375 3380 3390 CKD ECKD FBA
CKD	2311 2314 3310 3330 3340 3350 3375 3380 3390
ECKD	3380 3390
FBA	3370
CART	3480

Group-Name	Device-Type
TAPE	3410 3411 3420 8809
MSAM	VSAM-managed SAM files

With Disk or Tape Management Systems

If you use a supported disk or tape management system, CA-DYNAM/D or CA-EPIC, use the following values for the UNIT parameter on your Process statements:

UNIT=DLBLONLY allows for either CA-DYNAM/D or CA-EPIC to determine the data set characteristics such as primary and secondary allocation amounts and unit allocation and perform the actual allocation of the disk data set.

When you specify UNIT=DLBLONLY, CA-DYNAM/D or CA-EPIC actually controls the data set based upon the characteristics in its system catalog. For example, the pool name, allocation unit (records, blocks, tracks, or cylinders) are kept in the catalog. When you specify this parameter, the data set name *must* be predefined to the system catalog.

Using the UNIT=DLBLONLY parameter is the preferred method of allocation when you are using a disk management system. This method allows DYNAM or EPIC to control the data set at all times.

UNIT=DNOASGN allows either DYNAM or EPIC to perform file allocation for Connect:Direct without your having to predefine the file to the system catalog. The UNIT=DNOASGN parameter can also be used to override the catalog pool specification. When you use this parameter, you must specify the VOL=SER and SPACE parameters.

UNIT=TNOASGN informs Connect:Direct that your tape management system will control the tape data set and the tape drive allocation. When you use TNOASGN, DYNAM or EPIC will perform the following functions for Connect:Direct:

- Control tape drive
- Request the operator to mount the tape volume
- Perform the tape drive LUB/PUB assignments
- Allocate the tape drive to Connect:Direct.

VOL=([PRIVATE],[RETAIN],[volume-count]
,[SER=(serial-number[,serial-number.,,,])]) |
SER=(serial-number,[serial-number])

specifies the volume serial number(s) containing the file and optional processing associated with the file. If VOL is not specified with the FROM parameter, the file must be cataloged.

For Disk Management Systems such as CA-DYNAM/D or CA-EPIC, the VOL=SER parameter serves the following purposes:

- If you are allocating a “START TRACK-1" data set and using UNIT=DNOASGN, VOL=SER specifies the DYNAM or EPIC dynamic space pool name, where the data set will be allocated, for example: POOL01.
- VOL=SER can be used to force a disk data set allocation to a specific volume whether the data set is controlled or uncontrolled.
- VOL=SER can be used to override the dynamic space pool name and force a data set to be allocated on a specific device.

PRIVATE specifies allocation of an output file only if the volume is specifically requested and is used for direct access devices only.

Note: Connect:Direct VSE does not use this parameter but it is supported on other Connect:Direct platforms and is listed here for compatibility with those platforms.

RETAIN has no significance because Connect:Direct does dynamic deallocation of data sets. However, if RETAIN is omitted, a comma must be coded. If RETAIN is specified, the volume is not retained as it would be in a regular batch. The parameter is ignored by Connect:Direct.

volume-count specifies the maximum number of volumes required by an output file.

SER identifies by serial number the volumes on which the output file resides or will reside.

Note: When you allocate an output scratch tape data set for a VSE node, do not code the VOL=SER parameter. Connect:Direct will prompt the operator to mount a scratch tape at file open time. The volume serial number of SCRTCH is also reserved and should not be used.

If you need a VOL=SER for MSAM files, update the default model for MSAM. Change the VSAM default model, called DEFAULT.MODEL.ESDS.SAM, to specify the volumes needed.

For DASD Manager POOL allocation when UNIT=DNOASGN, you must also specify VOL=SER=poolname.

For DASD Manager specific volume allocation, when UNIT=DNOASGN, you must specify VOL=SER=*xxxxxx*.

VSAMCAT = (dsn, mode, userid, pswd, cuu)

specifies the VSAM catalog in which the VSAM file resides.

dsn is the name of the catalog in which the file resides. This subparameter is required.

mode specifies the VSE file mode of the catalog. This subparameter is included for compatibility with Connect:Direct VM. Any 2-character alphanumeric value is accepted.

userid specifies the VSE userid that owns the catalog. This subparameter is included for compatibility with Connect:Direct VM. Any 8-character alphanumeric value is accepted.

pswd specifies the user password of the VSE user that owns the catalog. This subparameter is included for compatibility with Connect:Direct VM. The value can be left blank.

cuu specifies the device address of the VSAM catalog. This subparameter is included for compatibility with Connect:Direct VSE. Any valid unit number is accepted.

Note: Only the dsn subparameter of the VSAMCAT parameter is required. Specify VSAMCAT=(dsn,1,1,,111) as a standard.

Example

This example COPY statement copies a data set from one Connect:Direct VSE node to another; its description follows:

```
STEP1    COPY  FROM  (DSN=SRCDATA.SET                                  -
                     DISP=(SHR)                                       -
                     PNODE                                            -
                     DCB=(DSORG=PS,LRECL=80,                          -
                          RECFM=FB,BLKSIZE=3120)                      -
                     UNIT=3380                                        -
                     VOL=SER=VOL003                                   -
                     )                                                -
               COMPRESS                                               -
               TO    (                                                -
                     DSN=DESTDATA.SET                                 -
                     DCB=(DSORG=PS,LRECL=80,                          -
                          RECFM=FB,BLKSIZE=3120)                      -
                     DISP=(NEW)                                       -
                     UNIT=242                                         -
                     SPACE=(600,(300))                                -
                     SNODE                                            -
                     )
```

- The COPY step is named STEP1.
- The DCB attributes specified in the TO clause of the COPY statement are used for file allocation.
- Unit and volume serial number are specified on the PNODE; however, only unit is specified on the SNODE.
- Specifying COMPRESS without a subparameter indicates that blanks will be compressed during transmission and converted back to the original string during decompression.

- Space parameters for the new TO data set are explicitly specified. This will allocate the new file on track 600 of unit 242 for a length of 300 tracks.

Note: See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Using SYSOPTS for DBCS Examples

Notification that a Process is transferring a DBCS file is done by means of the SYSOPTS statement. This statement must be included on the host node definition of the COPY statement.

By requiring the Process to notify Connect:Direct of its DBCS capability through the SYSOPTS statement, support for multiple transfers with multiple translation tables is possible. Furthermore, all Processes support compression and checkpointing.

File transfer with double-byte character set (DBCS) is only supported in record mode. It is not supported in block mode. The following example has a **tablename** of EBCXKSC and

Note: Transfers attempted in block mode can produce unpredictable results in the destination file. These results can compromise data integrity. You will not receive an error message.

the default values x'0E' for **so**, and x'0F' for **si**.

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F)"
```

The following example has a **tablename** of KSCXEBC and the default values x'0E' for **so**, and x'0F' for **si**.

```
SYSOPTS="DBCS=(KSCXEBC,0E,0F)"
```

The following example has a **tablename** of EBCXKSC and the NOSO value x'00' for **so** and **si**.

```
SYSOPTS="DBCS=(EBCXKSC,00,00)"
```

The following example has a **tablename** of EBCXKSC and takes the defaults for **so** and **si**.

```
SYSOPTS="DBCS=(EBCXKSC)"
```

The following example has a **tablename** of USERTAB and takes the defaults for **so** and **si**. USERTAB is a user-defined, customized translation table.

```
SYSOPTS="DBCS=USERTAB"
```

Defining a DBCS Capable Process

The following PC-to-host DBCS translation uses the supplied translation table KSCXEBC. Required parameters for this translation are in bold print.

```
/*****************************************************************/
/*           PC to HOST DBCS translation using table KSCXEBC      */
/*****************************************************************/
PCTOHOST  PROCESS SNODE=HOSTNODE                                    -
                  HOLD=CALL
STEP01  COPY                                                        -
              TO (PNODE                                             -
                  DSN='hlq.PCFILE'                                  -
                  DISP=(RPL,CATLG)                                  -
                  UNIT=SYSDA                                        -
                  DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS)    -
                  SPACE=(254,(1000,100))                            -
                  SYSOPTS="DBCS=KSCXEBC"                            -
                  )                                                 -
            FROM (SNODE                                             -
                  DSN=PCFILE                                        -
                  TYPE=ASC2ASC                                      -
                  DISP=SHR                                          -
                  )
```

The previous sample COPY statement copies a data set from a PC to a host Connect:Direct VSE node; its description follows:

- The copy step is named STEP01.
- The input data set is cataloged after successful completion of the Process.
- The DCB attributes specified in the TO clause of the COPY statement are used for file allocation.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the supplied translation table KSCXEBC.
- UNIT has been specified on the PNODE only.
- The TYPE parameter on the FROM clause of the COPY statement must be set to ASC2ASC.

The following host-to-PC DBCS translation uses the supplied translation table EBCXKSC. Required parameters for this translation are in bold print.

```
/*****************************************************************/
/*           HOST to PC DBCS translation using table EBCXKSC     */
/*****************************************************************/
HOSTTOPC  PROCESS SNODE=PCNODE-
                     HOLD=CALL
STEP01         COPY                                                -
                     FROM (PNODE                                   -
                           DSN='hlq.HOSTFILE'                      -
                           SYSOPTS="DBCS=EBCXKSC"                  -
                           DISP=(SHR)                              -
                           )                                       -
                     TO   (SNODE                                   -
                           DSN=PCFILE                              -
                           TYPE=ASC2ASC                            -
                           DISP=RPL                                -
                           )
```

The previous sample COPY statement copies a data set from a host Connect:Direct VSE to a PC node; its description follows:

- The copy step is named STEP01.
- The SYSOPTS attribute is specified in the FROM clause of the COPY statement is used to define the default translation table EBCXKSC.
- The TYPE parameter on the TO clause of the COPY statement must be set to ASC2ASC.

The following UNIX-to-host DBCS translation uses the default translation table EBCXKSC. Required parameters for this translation are in bold print.

```
/*****************************************************************/
/*           UNIX to HOST DBCS translation using table EBCXKSC    */
/*****************************************************************/

STEP01   COPY                                                     -
               FROM (PNODE                                        -
                     DSN='hlq.UNIXFILE'                           -
                     SYSOPTS="DBCS=EBCXKSC"                       -
                     DISP=(SHR)                                   -
                     )                                            -

               TO   (SNODE                                        -
                     DSN='/unixfile'                              -
                     SYSOPTS=":xlate=no:strip.blanks=no:"         -
                     DISP=RPL                                     -
                     )
```

The previous sample COPY statement copies a data set from a UNIX to a host Connect:Direct VSE node; its description follows:

- The copy step is named STEP01.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the default translation table EBCXKSC.
- The SYSOPTS parameter on the FROM clause of the COPY statement is required.

The following host-to-UNIX DBCS translation uses the default translation table KSCXEBC. Required parameters for this translation are in bold print.

```
/*****************************************************************/
/*          HOST to UNIX DBCS translation using table KSCXEBC     */
/*****************************************************************/

STEP02   COPY                                                     -
         TO    (PNODE                                             -
               DSN='hlq.HOSTFILE'                                 -
               SYSOPTS="DBCS=KSCXEBC"                             -
               DISP=(RPL,CATLG)                                   -
               UNIT=SYSDA                                         -
               DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS)     -
               SPACE=(254, (1000,100))                            -
               )                                                  -
         FROM   (SNODE                                            -
               DSN='/unixfile'                                    -
               SYSOPTS=":xlate=no:strip.blanks=no:"               -
               DISP=SHR                                           -
               )
```

The previous sample COPY statement copies a data set from a host Connect:Direct VSE to a UNIX node; its description follows:

- The copy step is named STEP02.
- The SYSOPTS attribute specified in the TO clause of the COPY statement is used to define the default translation table KSCXEBC.
- The DCB attributes specified on the TO clause of the COPY statement are used for file allocation.
- Unit is specified on the PNODE.
- The SYSOPTS parameter on the FROM clause of the COPY statement is required.

Chapter 18

Connect:Direct VSE RUN JOB Statement

The Connect:Direct VSE RUN JOB statement allows a job to be submitted through the VSE virtual reader, which is a facility that transfers jobs to VSE/POWER. Connect:Direct supports job submission only from a file existing on the node that executes the RUN JOB statement. Connect:Direct software does not verify job statements. To determine the completion status of a RUN JOB statement, check Connect:Direct statistics records.

Format

The Connect:Direct VSE RUN JOB statement is comprised of the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN JOB**	**(DSN = member-type(member) | member)**
		PNODE | SNODE

Field Descriptions

RUN JOB
identifies the statement with all its parameters as the RUN JOB statement.

Required Parameters

DSN = member-type(member) | member
specifies the name of a member in the LIBDEF source chain containing the job to be submitted. If the member-type is not specified, this job must be catalogued with a member type of J. All POWER JECL and JCL statements should begin in column one. For com-

patibility with previous releases of Connect:Direct VSE, the JCL/JECL can begin in column two.

Optional Parameters

PNODE
specifies that the job is to be submitted on the primary node (PNODE), which is the node with Process control. PNODE is the default value.

SNODE
specifies that the job is to be submitted on the secondary node (SNODE), which is the node that interacts with the PNODE.

Example

The example RUN JOB statement named STEP1 executes job TESTJOB.J from the source library on the SNODE. TESTJOB.J must exist as a member in a source library on the SNODE system. It must have been cataloged previously as a J member.

```
STEP1    RUN JOB    (DSN=J(TESTJOB))                                    -
                    SNODE
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 19

Connect:Direct VSE RUN TASK Statement

The Connect:Direct VSE RUN TASK statement allows user programs, or subtasks, to be attached during Process execution. It is permitted only within a Process and is structured so that the Process will wait until the subtask has finished running before the next step in the Process executes.

The RUN TASK statement can be used to attach an application subtask during Process execution. A list of parameters for the subtask can be specified in the RUN TASK statement. The subtask can be attached at either node involved in Process execution.

The program must be placed in a load library that can be accessed by the Connect:Direct DTF.

You can use the RUN TASK statement to pass user parameters to the subtask. The RUN TASK statistics log records the return code, program name, and parameter list for the subtask. The log also records the dates and times for starting and completing the subtask.

Writing the Program for the RUN TASK Statement

ASMTASK is a sample program in the SAMPLIB file supplied on the distribution tape. It can be used as a guideline when writing a program to be attached to a Process with a RUN TASK statement.

If the program requires libraries, load modules, or other files for execution, ensure that the library that contains the modules is part of the LIBDEF PHASE chain when Connect:Direct VSE is started.

The programs DMNOTIFY and DMRTAMS can also be invoked by the RUN TASK statement. See the *Connect:Direct VSE/ESA User's Guide* for further details concerning the use of these programs.

Format

The Connect:Direct VSE RUN TASK statement includes the following parameters. The required parameters and keywords are in bold. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN TASK**	**(PGM=program-name**
		PARM=(parameter [,parameter,...])
		)
		PNODE | SNODE

Field Descriptions

RUN TASK
identifies the statement with all its parameters as the RUN TASK statement.

Required Parameters

PGM = program-name
specifies the name of the program to be attached as a subtask. The program runs on the node specified and has access to the DLBL cards allocated on that node only.

Optional Parameters

PARM = (parameter [, parameter,...])
specifies the parameters that are to be passed to the subtask when that subtask is attached. These parameters are the actual parameters rather than a list of addresses. Null parameters can be specified by adjacent commas.

The actual format of the parameter list that is passed to the program consists of a 2-byte field, indicating the length of the parameter followed by the parameter itself. The valid data types for the PARM parameter are as follows:

CLn'value' specifies a data type of character with a length of **n**, where **n** is the number of bytes. The length is optional. If it is not specified, the actual length of the value is used. If the length specified is less than the real value, the data is truncated. If the length specified is longer than the value, the value is padded with blanks on the right. For example, CL44'FILE.NAME'

XLn'**value**' specifies a data type of hexadecimal with a length of **n**, where **n** is the number of bytes. The length is optional. If it is not specified, the length of the value is used. If the length specified is less than the real value, the data is truncated. If the length specified is longer than the value, the value is padded on the left with binary zeros. For example, XL8'FF00'

H'**value**' specifies a half-word value. No length can be specified. The value can be specified with a plus (+) or minus (-) sign. If no sign is given, plus is assumed. For example, H'-32'

F'**value**' specifies a full-word value. No length can be specified. The value can be specified with a plus (+) or minus (-) sign. If no sign is given, plus is assumed. For example, F'4096'

PLn'**value**' specifies a packed value. The length is optional; if it is not specified, the length of the value is used. If the length specified is longer than the value, the value is padded on the left with zeros. The length specifies the size of the field in bytes and cannot be longer than 16. The value can be specified with a plus (+) or minus (-) sign. If no sign is given, plus is assumed. For example, PL10'+512'

If no data type or length is specified, the parameter is assumed to be the character type and the length of the parameter is used. For example, if PARM=('FILE.NAME') is specified, the length used is **9**.

The parameter can also be specified as a symbolic value that is resolved when the Process is submitted. If a symbol is used, the parameter must be specified without a data type designation or length. For example, &PARM1

When using strings comprised of symbolic substitution, the strings must be enclosed in double quotes. If an ampersand (&) is to be passed as part of the parameter, then the data-type format must be used. For example, CL8'&PARM1' uses no substitution; CL8"&PARM1" indicates that the value is substituted.

The following example shows how Connect:Direct VSE passes parameters.

PNODE
specifies that the program is to be executed on the PNODE, which is the default.

SNODE
specifies that the subtask is to be attached on the secondary node (SNODE), which is the destination node.

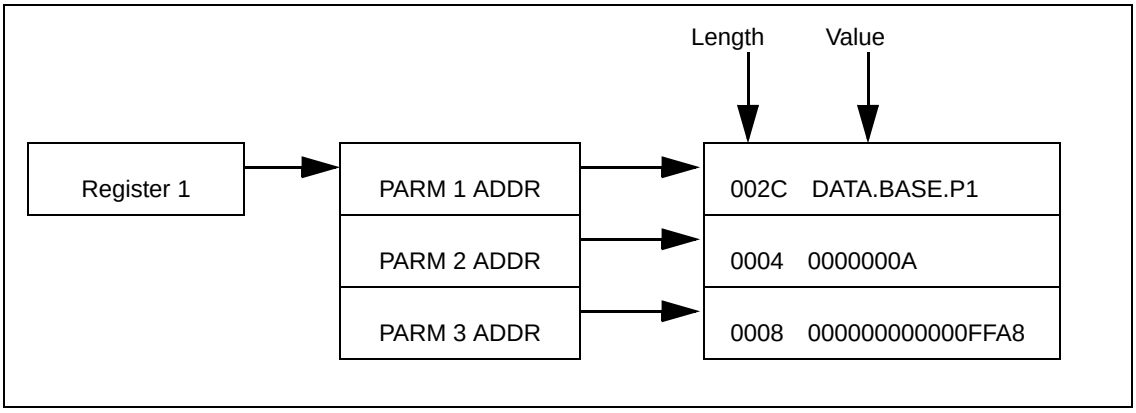
Note: The program must exist as an executable module in the LIBDEF PHASE chain on the specified node.

Example

This example RUN TASK statement runs the program named MYTASK. It is attached to the Process on the secondary node (SNODE) and is passed a list of three parameter addresses.

```
STEP1     RUN TASK    (PGM=MYTASK                                          -
                      PARM=(CL44'DATA.BASE.P1',                            -
                      F'0010', XL8'FFA8'))                                 -
                      SNODE
```

The parameter passing convention for the program MYTASK is shown in the following figure. In this case, Register 1 points to a parameter list of three parameters. It would contain 0 if no parameters were specified. Connect:Direct sets the high-order bit in PARM 3 ADDR to indicate the end of the PARM list.



See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 20

Connect:Direct VSE SUBMIT Statement

During execution of a Connect:Direct Process, the SUBMIT statement causes another Connect:Direct Process to be submitted to either the PNODE (primary node), which is the node with Process control, or to the SNODE, which is the node that interacts with the PNODE during Process execution. The Process to be submitted must reside in a file on the node where the SUBMIT statement will execute. This node is referred to as the SUBNODE.

Note: The SUBMIT statement described in this chapter is not the same as the SUBMIT command. The SUBMIT statement is used within a Connect:Direct Process to submit another Connect:Direct Process. See the *Connect:Direct VSE/ESA User's Guide* for SUBMIT command syntax and parameters.

Format

The Connect:Direct VSE SUBMIT statement format includes the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the SUBMIT statement format.

Label	Statement	Parameters
stepname	**SUBMIT**	**DSN=mt(membername)**
		NEWNAME=new-name
		SUBNODE=PNODE | SNODE
		SNODE=secondary-node
		SNODEID=(id [,pswd] [,newpswd])
		SACCT=(snode-accounting-data)
		PNODEID=(id [,pswd] [,newpswd])
		PACCT=(pnode-accounting-data)
		CASE=Yes | No

Label	Statement	Parameters
		CLASS=n
		HOLD=Yes | No | Call
		PRTY=n
		REQUEUE=Yes | No
		RETAIN=Yes | No | Initial
		STARTT=([date | day] [,hh:mm:ssxm])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Field Descriptions

SUBMIT
identifies the statement with all its parameters as the SUBMIT statement.

Required Parameters

DSN = mt (membername)
specifies the member type and member of the Process that resides in the Process library specified by the LIBDEF statement in the Connect:Direct JCL.

mt must be specified as a single character A-Z, 0-9, $, #, or @.

membername is the name of the member in the Process library.

Optional Parameters

CASE = Yes | No
specifies whether parameters associated with accounting data, userid, password, and data set name in the command and in the Process are to be case sensitive. The default is NO.

CLASS = n
determines the node-to-node session on which a Connect:Direct Process can execute. If CLASS is not specified in the Connect:Direct Process, it will default to the class value specified in the ADJACENT.NODE NETMAP record for the destination node (SNODE). Values range from 1-255.

HOLD = Yes | No | Call
specifies whether the Connect:Direct Process is placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process. When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed in the Hold queue even if a start time is specified.

No specifies that the Process executes as soon as possible. HOLD=NO is the default.

Call specifies that the Process is placed in the Hold queue until a VTAM session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

NEWNAME = new-name
specifies the new name to be given to the Process. The default value is the label on the PROCESS statement.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in parentheses. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one used to sign on to the Connect:Direct system.

id specifies the security ID passed to a security exit (1-8 characters).

pswd specifies the current security password. This parameter can be used by the security exit to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

newpswd specifies the new security password. It can be used by the security exit to change the current security password to the new security password (1-8 alphanumeric characters).

PRTY = n
specifies the Connect:Direct Process priority in the Transmission Control Queue (TCQ). The TCQ is a file that holds all Processes that have been submitted to a Connect:Direct node. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect VTAM transmission priority. The range is from 0-15. If PRTY is not specified, the default is the priority defined by the PRTYDEF keyword in the Connect:Direct VSE initialization parameters.

REQUEUE = Yes | No
specifies whether a COPY step should requeue if an x37 abend occurs during processing. This parameter is valid only if used when checkpointing.

Yes allows the requeued Process to be placed in the Hold queue with a status of HELD IN ERROR (HE). Corrective action can be taken and the Process restarted with the failing step; checkpointing resumes at the last successful checkpoint. Note that the Process must be explicitly released from the Hold queue when the status is HELD IN ERROR (HE).

No causes the Process to run to completion, executing subsequent steps when a COPY step fails with an abend (such as x37). The default is NO.

RETAIN = Yes | No | Initial
keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains on the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is automatically held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval.

When a Connect:Direct Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time Connect:Direct is initialized. Processes submitted with RETAIN=INITIAL do not execute when initially submitted.

Note: STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'
specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in parentheses. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

For Connect:Direct OS/400: This parameter is ignored when the SNODE is a Connect:Direct OS/400 node.

SNODE = secondary-node
SNODE = TCPNAME = tcpvalue
is a 1-16 alphanumeric character name that specifies the symbolic node name of the secondary node. The name can be expressed in alphanumerics or nationals (@ # $) with embedded periods. The first character must be alphabetic.

This parameter can override the value specified in the PROCESS statement. The default value for SNODE is the value coded in the PROCESS statement. Connect:Direct allows the PNODE and SNODE to specify the same symbolic node name.

Use the **TCPNAME=tcpvalue** form of the SNODE parameter to specify TCP/IP connections that are not defined in the Connect:Direct Network Map. **tcpvalue** is defined as the TCP/IP network address, the network name, or an alias for the network name. The tcpvalue can be from 1 to 16 characters with embedded periods. If the network name is longer than 16 characters, you must specify an alias for the network name.

If the TCPNAME keyword is used, the default TCP/IP port number is assumed.

SNODEID = (id [,pswd] [,newpswd])

specifies security user IDs and passwords at the secondary node (SNODE).

id specifies the security ID passed to the security system on the SNODE (1-8 alphanumeric characters).

For Connect:Direct Tandem: This subparameter specifies the Tandem group number and user number. These numbers can range from 0-255. When specifying a Tandem value, you must use a period character as a separator between the group number and the user number.

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles may be 10 characters long.

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

For Connect:Direct Tandem: The IBM VSE node only recognizes passwords specified in uppercase alphanumeric characters. Therefore, a Process cannot be successfully initiated from Connect:Direct VSE with Connect:Direct Tandem unless the Connect:Direct Tandem SNODEID password follows the same convention (no lowercase characters and no control characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

For Connect:Direct OS/400: This subparameter is ignored.

STARTT = ([date | day] [,hh:mm:ssXM])

specifies that the Process will execute at a selected date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

---

date specifies that the Process is to be held until a specific date. You can specify the month (mm), day (dd), and year (yy) in one of the following two formats:

- mm/dd/yy
- mm.dd.yy

Note: You must use periods or backslashes (/) to separate the components of a date value to guarantee transfers between all platforms.

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

day specifies the day of the week that the Process is to be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value can be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday with Monday as the only STARTT parameter, the Process does not run until the following Monday.

You can also specify TODAY, which releases the Process for execution today, or TOMORROW, which releases the Process for execution the next day.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON or MIDNIGHT.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed in the Hold queue even if a start time has been specified.

SUBNODE = PNODE | SNODE

specifies the node on which the Process defined in this SUBMIT statement will execute. Specifying PNODE means that the Process is submitted on the node that has Process control. Specifying SNODE means that the Process is submitted on the node participating in, but not controlling, Process execution. In both cases, the Process must reside on the node on which it is being submitted. The default is PNODE.

For Connect:Direct OS/400: SUBNODE=SNODE is not valid if communicating with a node running Connect:Direct OS/400.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden when submitting the Process.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Note that the symbolic parameter for the SUBMIT statement must begin with a single ampersand. This allows submission of the Process that contains the SUBMIT statement to resolve symbolic parameters correctly.

An ampersand symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Example

This example shows using SUBMIT with the DSN parameter. Symbolics for PROCESS2 are supplied by the operator submitting PROCESS1.

The operator at the PNODE issues the following Connect:Direct SUBMIT PROCESS command to initiate the file transfer:

```
SUB       PROC=PROCESS1                                                 -
          &DSN1=A345.DATA                                               -
          &DSN2=A345.NEW.DATA
```

PROCESS1 executes:

```
PROCESS1  PROCESS    PNODE=CD.LA  SNODE=CD.DALLAS                       -
                     &DSN1=&DSN1                                        -
                     &DSN2=&DSN2                                        -
                     &PRTY=14                                           -
COPYSTEP  COPY  FROM (DSN=&DSN1 DISP=SHR PNODE)                         -
                TO   (DSN=&DSN2 DISP=SHR SNODE)                         -
SUBSTEP   SUBMIT     DSN=N(PROCESS2)                                    -
                     PRTY=&PRTY                                         -
                     SUBNODE=SNODE                                      -
                     &DSN=&DSN2
```

PROCESS1 submits PROCESS2:

```
PROCESS2  PROCESS    PNODE=CD.DALLAS                                    -
                     SNODE=CD.NEWYORK
COPYSTEP  COPY  FROM (DSN=&DSN2 PNODE)                                  -
                TO   (DSN=A345.NEW.DATA1 DISP=SHR)
```

- PROCESS1 copies the file A345.DATA in LA to a file called A345.NEW.DATA in Dallas. It then submits PROCESS2, which executes on the Dallas node. PROCESS2 is submitted with a PRTY of 14.
- PROCESS2 copies the file A345.NEW.DATA in Dallas to the file A345.NEW.DATA1 in New York.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 21

Connect:Direct VSE SYMBOL Statement

The Connect:Direct VSE SYMBOL statement allows you to build symbolic substitution values.

Format

The Connect:Direct VSE SYMBOL statement format follows. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**SYMBOL**	**&symbolic_name=variable-string**

Field Descriptions

SYMBOL
identifies the statement with all its parameters as the SYMBOL statement.

Required Parameters

&symbolic_name = variable string
specifies the string that is substituted into the Process.

When Connect:Direct software encounters an ampersand (&) plus 1-17 alphanumeric characters, Connect:Direct substitutes a string represented by that ampersand and the alphanumeric characters.

Symbols in the string are resolved from previously specified values in a PROCESS, SUBMIT, or SYMBOL statement. With the SYMBOL statement, different pieces of a Connect:Direct statement string can be concatenated, allowing you to move data in many ways.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Example

In the Process, PROCSYM, the SYMBOL statement causes the file Z813103.ABC.DEF to be copied into file Z813103.TEST.XXX.

Note that two vertical bars preceded and followed by blanks (||) are used to indicate concatenation. Bracketing backslashes ensure that special characters within the string are maintained. The COPY statement is resolved by the symbolic substitution of the &AA symbol to Z813103.ABC.DEF.

```
PROCSYM   PROCESS    HOLD=YES                                          -
                     SNODE=CD.NEWYORK                                  -
                     SNODEID=(RSMITH,ROGER)                            -
                     PRTY=12
          SYMBOL     &A1=ABC
          SYMBOL     &A2=DEF
          SYMBOL     &AA=\Z813103.\ || &A1 \.\ || &A2
STEP1     COPY  FROM (DSN=&AA)                                          -
                     DCB=(DSORG=VSAM) PNODE)                           -
                TO   (DSN=Z813103.TEST.XXX                             -
                     DCB=(DSORG=VSAM) DISP=NEW SNODE)
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 22

Connect:Direct VSE Conditional Statements

Conditional statements permit you to alter the sequence of Connect:Direct Process execution based on the completion of the previous step in the Process.

Formats

Formats for Connect:Direct conditional statements follow. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**IF**	**(label condition nn) THEN**
		(process steps)
	ELSE	
		(alternative process steps)
	EIF	
optional	GOTO	label
	EXIT	

Statement Descriptions

IF THEN

specifies that the Connect:Direct system executes a block of Connect:Direct statements based on the completion code of a Process step. An EIF statement must be used in conjunction with an IF THEN statement. A return code with the high order bit on is evaluated as a negative return code. See *Field Descriptions* on page 196 for details.

ELSE
designates a block of Connect:Direct statements that execute when the IF THEN condition is not satisfied. No parameters exist.

EIF
is required for specifying the end of the IF THEN or IF THEN ELSE block of statements. No parameters exist.

GOTO
moves to a specific step within a Process. *Field Descriptions* on page 196 for details.

EXIT
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

label
For the IF THEN statement, the label specifies the name of a previous step whose completion code is used for comparison.

For the GOTO statement**,** the label specifies the name of a subsequent step in a Process (required for GOTO only). The name can neither be the label of a preceding step nor the label of the GOTO statement of which it is a part.

Note: User-defined labels must begin in column one. The label consists of a 1-8 character alphanumeric string, with the first letter alphabetic only.

condition
specifies the type of comparison to be performed. This condition checking can be based on comparisons for equality, inequality, greater than, less than, greater than or equal to, and less than or equal to.

The completion code from RUN JOB is for the job submission only. It is not the completion code of the job submitted.

Valid symbols, alternate symbols, and conditions follow:

= or EQ specifies that the completion code must be equal to the value nn for the condition to be satisfied.

<> or ¬= or NE specifies that the completion code must not equal the value nn for the condition to be satisfied.

>= or ¬< or GE specifies that the completion code must be greater than or equal to the value nn for the condition to be satisfied.

> or GT specifies that the completion code must be greater than the value nn for the condition to be satisfied.

<= or ¬> or LE specifies that the completion code must be less than or equal to the value nn for the condition to be satisfied.

< or LT specifies that the completion code must be less than the value nn for the condition to be satisfied.

nn
specifies the numeric value used for completion code checking. If coded as X‘nn’, it is a hexadecimal value; any other coding indicates it as decimal.

If a completion code less than 4 is returned, typically the Process completed successfully. In most cases, a return code greater than 4 indicates the Process ended in error. A return code equaling 4 indicates a warning.

THEN
specifies subsequent processing to be performed if the condition specified is true.

Example

The following Connect:Direct Process contains all of the conditional statements. A description of each step follows.

```
COPY01   PROCESS    SNODE=CD.CHICAGO
STEP01   COPY  FROM (DSN=ABC.FILEA                                    -
                    DCB=(DSORG=PS, BLKSIZE=800, LRECL=80, RECFM=FB    -
                    VOL=SER=PACK01 DISP=(SHR) PNODE)                  -
               TO   (DSN=JKL.FILEA                                    -
                    VOL=SER=PACK02                                    -
                    SPACE=(1993, (15)) DISP=(NEW) SNODE
STEP02   IF         (STEP01 GT 4) THEN
                         GOTO STEP07
         ELSE
STEP03   RUN JOB    (DSN=USERJOB) SNODE
         EIF
STEP04   IF         (STEP03 >= 8) THEN
                         EXIT
         EIF
STEP05   IF         (STEP03 LT 4) THEN
STEP06   COPY  FROM (DSN=ABC                                          -
                    DCB=(DSORG=PS, BLKSIZE=800, LRECL=80, RECFM=FB    -
                    VOL=SER=PACK01 DISP=(SHR) PNODE)                  -
               TO   (DSN=MNO                                          -
                    VOL=SER=PACK02                                    -
                    SPACE=(2008, (15)) DISP=(NEW) SNODE
         EIF
         EXIT
STEP07   RUN TASK   (PGM=DMNOTIFY,                                    -
                    PARM=(‘FAIL’,ABC.FILEA))                          -
                    PNODE
```

COPY01 is the PROCESS statement defining the secondary node as CD.CHICAGO.

STEP01 copies file ABC.FILEA on the PNODE to file JKL.FILEA on the SNODE.

STEP02 checks the completion code of STEP01. If STEP01 fails, STEP07 executes. If STEP01 ended with a completion code of 4 or less, STEP03 executes.

STEP03 submits the job, USERJOB, on the SNODE.

STEP04 checks the completion code of STEP03. If STEP03 fails with a completion code of 8 or greater, the Process terminates. Otherwise, STEP05 executes.

STEP05 checks the completion code from STEP03. If less than 4, indicating the step completed without errors, the COPY statement in STEP06 executes and the Process terminates.

STEP06 copies file ABC on the SNODE to file MNO on the PNODE.

STEP07 only executes if STEP01 fails. The program DMNOTIFY runs, sending a message indicating the operation failed to the console operator.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples containing conditional statements.

Chapter 23

Connect:Direct Tandem PROCESS Statement

A Connect:Direct Tandem NonStop Kernel PROCESS statement defines the attributes of a Connect:Direct Process and is always the first statement in a Process.

Format

The Connect:Direct Tandem PROCESS statement format includes the following parameters. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the PROCESS statement format.

Label	Statement	Parameters
process name	**PROC**ess	**SNODE=secondary-node-name**
		SACCT='snode-accounting-data'
		PNODE=primary-node-name
		PNODEID=(id | [,pswd])
		SNODEID=(id | [,pswd] [,newpswd])
		PACCT='pnode-accounting-data'
		CLASS=n
		HOLD=Yes | No | Call
		PRTY=n
		RETAIN=Yes | No | Initial
		STARTT=([date |day][,hh:mm:ssXM])

Label	Statement	Parameters
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Field Descriptions

process name
specifies the name of the Process. The Process name may be from 1-8 characters long. The first character must be alphabetic. The Process name must start in column one. This label is used to identify the Process in any messages or statistics relating to this Process.

PROCess
identifies the statement with all its parameters as the PROCESS statement. This statement identifier can be abbreviated to PROC.

Required Parameters

SNODE = secondary-node-name
is a 1-16 character alphanumeric name that specifies the secondary node (SNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

This is the logical node name that has been defined as the adjacent node in the network map.

Note: This parameter is not required if SNODE is specified on the SUBMIT command.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. Each logical unit (LU) has an assigned default class value, which allows a Process to execute on an LU having a matching class value or on LUs with higher class values. Class numbers are assigned in the order in which LUs appear in the network map. If a class value of 1 is specified, a Process will run on the first available LU.

If CLASS is not specified in the Connect:Direct Process or SUBMIT command, CLASS defaults to the default class specified in the PARSESS parameter of the adjacent node network map record.

HOLD = Yes | No | Call
specifies whether the Process is to be placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process.

No specifies that the Process is to execute as soon as possible. HOLD=NO is the default.

Call specifies that the Process is to be placed in the Hold queue until a session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks.

Note: PACCT is used only for documentation for the Connect:Direct Tandem software.

PNODE = primary-node-name
is a 1-16 character alphanumeric name that specifies the primary node (PNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

The node to which the Process is submitted is always the PNODE. This parameter defaults to the name of the node submitting the Process and need not be specified. It is used for documentation purposes only.

PNODEID = (id [,pswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one you used to sign on to a Connect:Direct node.

id specifies the Tandem group number and user number. These numbers can range from 0-255 and are separated by a period (.).

pswd specifies the current security password for the specified ID. This parameter can be used by the security system at the PNODE to validate the current security password (1-8 alphanumeric characters).

PRTY = n
specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ holds all Processes that have been submitted to a node. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect transmission priority. The range is from 0-15. If PRTY is not specified, the default is

the priority defined during Connect:Direct installation. See the *Connect:Direct Tandem Installation and Administration Guide* for more information.

RETAIN = Yes | No | Initial
keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains on the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval. However, a date is invalid as a STARTT subparameter when used in conjunction with RETAIN.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time the Connect:Direct system is initialized. The Process will not execute when initially submitted. Note that STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'
specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks.

For Connect:Direct OS/400: This parameter is ignored when the SNODE is a Connect:Direct OS/400 node.

SNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the secondary node (SNODE).

For Connect:Direct OS/400: If an SNODEID and password of the Process submitter is not coded in the PROCESS statement, the user ID and password of the Process submitter will be used for the security ID and password check by Connect:Direct OS/400.

For Connect:Direct UNIX: The security user ID and passwords are case sensitive.

id specifies the Tandem group number and user number. These numbers can range 0-255 and are separated by a period (.). Other operating environments limit the security ID to 1-8 alphanumeric characters.

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles may be 10 characters long.

pswd specifies the current security password. This parameter can be used by the security system on the SNODE to validate the current security password and can be 1-8

alphanumeric characters. This is optional unless the user has security set to require a password.

For OS/390, VM, and VSE nodes: The IBM OS/390, VM, and VSE nodes only recognize passwords specified in uppercase alphanumeric characters. For an alternative solution for mixed-case passwords, refer to the *Connect:Direct Tandem Installation and Administration Guide* for details on automatic security resolution.

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD™ must be running on Tandem.

For Connect:Direct OS/400: This subparameter is ignored.

STARTT = ([date | day][,hh:mm:ssXM])

specifies that the Process is not to be executed until a specified date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=I.

date specifies that the Process is to be held until the desired date. The day (dd), month (mm), and year (yy) can be specified in one of the following formats:

yymmdd	mm/dd/yy	yy/mm/dd	mm.dd.yy
yy.mm.dd	mmddyy	yyddd (Julian date)	yy/ddd (Julian date)

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

day specifies the day of the week that the Process is to be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value may be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday, with Monday as the only STARTT parameter, the Process will not run until the following Monday.

You can also specify TODAY, which releases the Process for execution the day and time of Process submission (unless the time of day is specified), or TOMORROW, which releases the Process for execution the next day. If a time of day is not specified with TOMORROW, the Process will execute after midnight.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT to release the Process for execution at midnight.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time is specified.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden in the SUBMIT command.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters can be enclosed in single quotation marks.

Example

The following is an example PROCESS statement; its description follows:

```
PROC1   PROCESS         SNODE=CD.NODE
                        SNODEID=(JONES,OPENUP)
                        CLASS=4
                        HOLD=YES
                        PACCT='OPERATIONS, DEPT. 87'
                        RETAIN=NO
```

The Process named PROC1 specifies a secondary node (SNODE) of CD.NODE. The corresponding security userids and passwords (SNODEID) have been included.

This Process will run in CLASS 4.

The Process will be placed on the Hold queue until it is released for execution with a CHANGE PROCESS command.

The PACCT parameter specifies that all accounting information will be attributed to the operations account, department 87, if the node has a program that maintains this information.

Once the Process executes, it will be deleted because the RETAIN parameter is set to NO. Note that NO is the default value for the RETAIN parameter.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 24

Connect:Direct Tandem COPY Statement

The Connect:Direct Tandem COPY statement allows you to copy files from one node to another.

General Considerations for Tapes

If a time-out occurs before a tape is loaded on an adjacent node, the operator must cancel the tape mount. This action cancels the Process on the adjacent node.

General Format Information

The COPY statement is comprised of a FROM parameter that includes the source file name and a TO parameter that includes the destination file name. Additional parameters and subparameters can be specified to further customize the file transfer operation.

The following pages provide the format of the COPY statement, separated into disk and tape files, and spooler jobs. All platform-specific parameters and subparameters are listed along with their possible values. Following these pages is a description of each parameter and subparameter.

To copy from one Connect:Direct system environment to another, refer to the appropriate COPY FROM and TO sections for those environments. For example, if the source file is located on a Connect:Direct OS/390 node, refer to the COPY FROM information for Connect:Direct OS/390. If the destination for the file is a Connect:Direct Tandem node, refer to the TO information in this chapter.

Format-Disk and Tape Files

The following formats are used to copy disk files from and to a Connect:Direct Tandem node. When the operation is copying to or from OS/390, VM, or VSE, tape files are supported.

Label	Statement	Parameters	
stepname	**COPY**	**FROM**	**(**
			DSN=filename |FILE=filename
			PNODE | SNODE
			TYPE=typekey
			DCB=([RECFM=record-format] [,BLKSIZE=no.-bytes] [,LRECL=no.-bytes])
			DISP=([OLD | SHR] , [KEEP | DELETE])
			IOEXIT=[exit-name | (exit-name [,parameter,...])] | [$process-name]
			SYSOPTS=["SET XLATE ON | YES | OFF | NO | table-name"]
			)
		TO	**(**
			DSN=filename | FILE=filename
			PNODE | SNODE
			TYPE=typekey
			DCB=([BLKSIZE=no.-bytes] [,DSORG=[U | 0] | [R | 1] | [E | 2] | [K | 3] [,KEYLEN=no.-bytes] [,LRECL=no.-bytes] [,RECFM=record-format])
			DISP=([NEW | OLD | MOD | RPL | SHR],, [KEEP | DELETE])
			IOEXIT=[exit-name | (exit-name [,parameter,...])] | [$process-name]

Label	Statement	Parameters
		SYSOPTS=([“SET TYPE [U | 0] | [R | 1] | [E | 2] | [K | 3]"] [“SET CODE file-code”] [“SET EXT (extent.size) | (pri.ext.size , sec.ext.size)"] [“SET REC record-length”] [“SET BLOCK data-block-length”] [“SET [NO] COMPRESS”] [“SET FORMAT 0|1|2”] ["SET PART ([sec.partition.num] [\system.name.$volume] [pri.ext.size] [sec.ext.size] [partial.key.value])"] [“SET [NO] AUDIT”] [“SET [NO] DCOMPRESS”] [“SET [NO] ICOMPRESS”] [“SET KEYLEN key-length”] [“SET KEYOFF key-offset”] [“SET ALTKEY ([key-specifier] [“SET [NO] PARTONLY”] [“SET ODDUNSTR”] [“SET [NO] REFRESH”] [“SET [NO] AUDIT”] [“SET MAXEXTENTS maximum-extents”] [“SET BUFFERSIZE unstructured-buffer-size"] [“SET [NO] BUFFERED”] [“SET [NO] AUDITCOMPRESS”] [“SET [NO] VERIFIEDWRITES”] [“SET [NO] SERIALWRITES”] [“SET [NO] BLOCKIO”] [“SET [NO] LARGEIO”] ["SET FAST.LOAD Y"] [FILE key-file-number] [KEYLEN key-length] [KEYOFF key-offset] [NULL | NO NULL] [[NO] UNIQUE] [[NO] UPDATE"])"]

Label	Statement	Parameters
		["SET ALTFILE key-file-number ,filename"] ["SET [NO] ALTCREATE"] ["SET PART ([sec.partition.num] [\system.name.$volume] [pri.ext.size] [sec.ext.size] [partial.key.value])"] ["SET FAST.LOAD.PRI priority"] ["SET FAST.LOAD.CPU cpu-number"] ["SET FAST.LOAD.SORTED Y"] ["SET XLATE ON | YES | OFF | NO | table-name"] ["SET FORMAT 0|1|2"]) For transfers from Windows to Tandem, use single quotation marks around each SET statement. Omit the parentheses and enclose the entire SYSOPTS statement in double quotation marks.
		)
		CKPT=nK | nM
		COMPRESS [[PRIMEchar=X'xx' | X'20' | C'c'] | EXTended]

Field Descriptions-Disk and Tape Files

COPY
identifies the statement with all its parameters and subparameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source file characteristics.

(FROM) DSN | FILE
specifies the source file name. File names are verified based on Tandem standard file name conventions. DSN or FILE is invalid when coding the IOEXIT parameter.

TO
specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN | FILE
specifies the destination file name. DSN or FILE is invalid when coding the IOEXIT parameter.

Optional Parameters

CKPT = nK | nM
specifies the byte interval for checkpoint support. Checkpointing allows restarting interrupted transmissions at the last valid transmission point, thus avoiding the need to restart transmission from the beginning; restart time is therefore reduced. Connect:Direct converts the specified value to a block boundary, and a data transmission checkpoint is taken at that position. K denotes thousands; M denotes millions.

Connect:Direct Tandem supports checkpointing for entry sequenced files, key sequenced files, unstructured files (but not unstructured code 101), and relative files. Connect:Direct Tandem does not support checkpointing for edit files, spool files, or file copies using the Fastload option.

Coding CKPT=0K in the COPY statement will override any checkpointing values specified in the initialization parameters; checkpointing will not occur.

COMPRESS [[PRIMEchar = X‘xx’ | X‘20’ | C‘c’] | EXTended]
specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at the destination. X‘xx’ is the hexadecimal representation of the compression character. C‘cc’ is the character representation of the compression character. The default subparameter for the COMPRESS parameter is PRIMEchar=X‘20’ (blank).

If compression is specified, Connect:Direct software will reduce the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character will be compressed to 1 byte.
- Repetitive occurrences (ranging from 3-63) of any other character will be compressed to 2 bytes.

Note: Compression is CPU-intensive, and its effectiveness is data dependent. It should only be used if its benefits are known.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X‘20’).

EXTended searches for repetitive strings of characters in data and compresses them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited and data is repetitive.

(FROM) DCB =([RECFM = record-format]
[,BLKSIZE = no.-bytes]
[,LRECL = no.-bytes])

specifies attributes to be used in allocating source files. If reading in an unstructured file by record count, BLKSIZE and LRECL must be specified.

RECFM specifies the format of the records in the file. Connect:Direct Tandem automatically determines the RECFM of the ENSCRIBE file; however, you can override this value to allow an unstructured file to be handled differently. For example, to copy an unstructured file containing 4096-byte records to an adjacent node in 132-byte records, code the following in the FROM clause of the Connect:Direct Tandem COPY statement:

```
DCB=(RECFM=U,BLKSIZE=4096,LRECL=132)
```

BLKSIZE specifies the length in bytes of the block. The minimum length is 512 bytes, and the maximum length is 4,096 bytes. All valid sizes are supported. If the value specified is not a standard value, it will be rounded up to the next largest standard size or to the maximum length.

LRECL specifies the length in bytes of the record. The LRECL values for each supported Tandem file type for the DP1 and DP2 operating systems are listed in the following table.

File Type	DP1 Limit	DP2 Limit
Relative Record	4072	4072
Unstructured	4096	4096
Entry Sequenced	4072	4072
Key Sequenced	2035	4062
Unstructured Code 101	2048	2048

(TO) DCB =([BLKSIZE = no.-bytes]
[,DSORG = dsorg]
[,KEYLEN = key-length]
[,LRECL = no.-bytes]
[,RECFM = record-format])

specifies attributes to be used in allocating destination. For destination files, these parameters override the DCB information provided in the source file at open time.

If you are transferring a text file without the DCB parameter specified, the file type on the Tandem node defaults to an unstructured file, code 101.

If you are transferring a binary file without the DCB parameter specified, the file type on the Tandem node defaults to an unstructured file, code 0.

For UNIX to Tandem copies: When copying files from UNIX to Tandem, use the DCB parameter to allocate destination files. Code any additional options using the SYSOPTS parameter.

BLKSIZE specifies the length in bytes of the block. The minimum length is 512 bytes, and the maximum length is 4,096 bytes. All valid sizes are supported. If the value specified is not a standard value, it is rounded up to the next largest standard size or to the maximum length.

DSORG specifies the file organization. File organizations supported are U, R, E, and K. If no value is supplied, DSORG defaults to the file organization of the sending file. Also valid are the numerical representations of 0, 1, 2, and 3, respectively.

KEYLEN specifies the length of the keys used in a file. The maximum length in bytes is 255.

LRECL specifies the length in bytes of the record. The LRECL values for each supported Tandem file type for the DP1 and DP2 operating systems are listed in the previous table.

RECFM specifies the format of the records in the file. Any valid record format, such as F (Fixed), FB (Fixed Block), FBA (Fixed Block ANSI carriage control), U (Undefined), V (Variable), VB (Variable Block), VBA (Variable Block ASA printer control), VBM (Variable Block Machine code control character), and VBS (Variable Block Spanned), can be specified.

(FROM) DISP = ([OLD | SHR], [KEEP | DELETE])

specifies the status of the file and its disposition after notification of successful transmission. Subparameters are as follows:

First Subparameter specifies the status of the file. This subparameter applies to all files. Options for this subparameter are as follows:

- **OLD** specifies that the source file will be opened with exclusive access.
- **SHR** specifies that the source file will be opened with shared access. SHR is the default.

Second Subparameter specifies the disposition of a normal Process step termination. Valid source file dispositions are as follows:

- **KEEP** specifies that the system will keep the file after the Process step has been completed.

- **DELETE** specifies that the system will delete the file after the Process step has been successfully completed.

(TO) DISP = ([NEW | OLD | MOD | RPL | SHR], [KEEP | DELETE])

specifies the status of the data on the receiving node. Subparameters are as follows:

First Subparameter specifies the status of the file. Options for this subparameter are as follows:

- **NEW** specifies that the Process step will create the destination file. NEW is the default.
- **OLD** specifies that the destination file existed before the Process began executing and that the Process will have exclusive control of the file. If DISP=OLD, the destination file may be any Tandem disk file.
- **MOD** specifies that the Process step will modify the file by adding data to the end of the file. DISP=MOD is valid for file types K, R, E, and EDIT files. DISP=MOD is not valid for other unstructured files. When setting DISP=MOD for file types E and R, specify SYSOPTS="SET NO BLOCKIO".
- **RPL** specifies that the destination file will replace any existing file or create a new file. If DISP=RPL and the file exists, the existing file attributes will be retained. Connect:Direct will not change existing file attributes.
- **SHR** specifies that the destination file existed before the Process began executing and that the file can be used simultaneously by another Process. The file is opened in shared access mode.

Second Subparameter is ignored and is indicated with a comma.

Third Subparameter specifies abnormal termination disposition. The KEEP/DELETE option is deactivated if checkpointing is specified. Connect:Direct must always KEEP the file to allow checkpointing to restart transmission from the last valid transmission point.

Valid destination file dispositions are as follows:

- **KEEP** specifies that the system will keep the file after the Process step is terminated abnormally.
- **DELETE** specifies the system will delete the file when the Process step is terminated abnormally.

IOEXIT = [exit-name | (exit-name [,parameter,...])] | [$process-name]

indicates that a user-written program is to be called to perform I/O requests for the associated data.

exit-name specifies the name of the user-written program to be given control for I/O-related requests for the associated data.

parameter specifies a parameter, or list of parameters, to be passed to the specified exit. For valid parameter formats, refer to the Connect:Direct Tandem RUN TASK parameters described in the *RUN TASK Statement* chapter beginning on 4-1.

For Connect:Direct Tandem nodes: Parameters are only valid if the SNODE is a Connect:Direct Tandem node.

$process-name specifies the name of the I/O exit Process currently running This Process is given control for I/O-related requests for the associated data.

PNODE

specifies the primary node and defines the direction of transfer. When PNODE is coded with the FROM parameter, the file to be copied resides on the primary node.

When PNODE is coded with the TO parameter, the file is sent to the primary node. PNODE is the default with the FROM parameter.

SNODE

specifies the secondary node and defines the direction of transfer. When SNODE is coded with the FROM parameter, the file to be copied resides on the secondary node.

When SNODE is coded with the TO parameter, the file is sent to the secondary node. SNODE is the default with the TO parameter.

(FROM) SYSOPTS = ["SET XLATE ON | YES | OFF | NO | table-name"]

allows you to specify whether the file being transferred will be converted from either ASCII to EBCDIC or EBCDIC to ASCII.

If XLATE ON or XLATE YES is specified or the file being copied is a spooler file or an edit file (unstructured file, code 101), Connect:Direct Tandem will check the XLFILE for a table named DEFAULT. If the DEFAULT table is not in XLFILE, then Connect:Direct Tandem uses the standard English language ASCII/EBCDIC table as defined by Tandem.

If the file being copied is a spooler file or an edit file (unstructured file, code 101), XLATE does not need to be specified because these files are automatically translated. Specifying SET XLATE OFF or XLATE NO disables translation.

SET XLATE indicates whether file conversion should be set on, off, or to a specified table.

- **ON | YES** specifies that text will be converted from either ASCII to EBCDIC or EBCDIC to ASCII, depending upon the copy direction.
- **OFF | NO** ensures that text conversion does not occur during file transfer.
- **table-name** is a 1-8 character name of a user-defined translation table in XLFILE. See the *Connect:Direct Tandem Installation and Administration Guide* for further details on user-defined translation tables.

(TO) SYSOPTS =(
["SET TYPE [U | 0] | [R | 1] | [E | 2] | [K | 3]"]
["SET CODE file-code"]
["SET EXT (extent.size) | (pri.ext.size,sec.ext.size)"]
["SET REC record-length"]
["SET BLOCK data-block-length"]
["SET [NO] COMPRESS"]
["SET [NO] DCOMPRESS"]
["SET [NO] ICOMPRESS"]
["SET KEYLEN key-length"]

["SET KEYOFF key-offset"]
["SET ALTKEY
([key-specifier]
[FILE key-file-number]
[KEYLEN key-length]
[KEYOFF key-offset]
[[NO] NULL]
[[NO] UNIQUE]
[[NO] UPDATE]
)"]
["SET ALTFILE key-file-number , filename"]
["SET [NO] ALTCREATE"]
["SET PART
([sec.partition.num]
[\system.name.$volume]
[pri.ext.size]
[sec.ext.size]
[partial.key.value]
)"]
["SET [NO] PARTONLY"]
["SET ODDUNSTR"]
["SET [NO] REFRESH"]
["SET [NO] AUDIT"]
["SET MAXEXTENTS maximum-extents"]
["SET BUFFERSIZE unstructured-buffer-size"]
["SET [NO] BUFFERED"]
["SET [NO] AUDITCOMPRESS"]
["SET [NO] VERIFIEDWRITES"]
["SET [NO] SERIALWRITES"]
["SET [NO] BLOCKIO"]
["SET [NO] LARGEIO"]
["SET FAST.LOAD Y"]
["SET FAST.LOAD.PRI priority"]
["SET FAST.LOAD.CPU cpu-number"]
["SET FAST.LOAD.SORTED Y"]
["SET XLATE ON | YES | OFF | NO | table-name"]
["SET FORMAT 0|1|2"]
)

allows you to specify system operation parameters on the COPY statement. It is an alternative way of specifying file creation attributes. Tandem File Utility Program (FUP™) syntax may be specified for creating Tandem-specific file options that are not otherwise allowed with the default Connect:Direct syntax. FUP-specific SET commands, unlike Connect:Direct specific SET commands, are passed directly to FUP.

Each system operation is indicated as a Tandem SET command. You can specify multiple SET command parameters with SET preceding each parameter.

For example:

```
SET parameter
SET parameter
```

You can also specify multiple SET command parameters with SET preceding the first parameter and commas separating each parameter. For example:

```
SET parameter, parameter, parameter
```

Enclose each system command in double quotation marks except when copying from Windows to Tandem.

```
SYSOPTS=("SET parameter" "SET parameter" "SET parameter")
```

Do not use continuation marks in the sysopts parameter. Type the text in a continuous string, with blanks separating each subparameter.

For transfers from Windows to Tandem only: Use single quotation marks to enclose each system operation and place the entire SYSOPTS statement in double quotation marks.

```
SYSOPTS="'SET parameter' 'SET parameter' 'SET parameter'"
```

For details on listed FUP SET commands, refer to the appropriate Tandem manual. The following commands are Connect:Direct Tandem SET commands and are described only in the documentation set for Connect:Direct Tandem: SET [NO] BLOCKIO, SET [NO] LARGEIO, SET XLATE, SET SPOOLER, SET FAST.LOAD Y, SET FAST.LOAD.PRI, SET FAST.LOAD CPU, SET FAST.LOAD.SORTED Y, and SET SPOOLNUM.

SET parameter commands define Tandem file attributes or special processing instructions for Connect:Direct Tandem. The SET commands follow in alphabetical order.

[NO] ALTCREATE specifies whether automatic alternate-key files will be created. The default is ALTCREATE.

ALTFILE specifies the file number and file name of an alternate-key file. When using the SET ALTFILE or SET ALTKEY commands, the first key file number must be equal to zero (0).

- **key-file-number** is an integer from 0-255, inclusive.
- **filename** is the name of the alternate-key file for the key-file-number.

ALTKEY specifies an alternate key. When using the SET ALTFILE or SET ALTKEY commands, the first key file number must be equal to zero (0). Valid values are as follows:

- **key-specifier** is a 2-byte value that uniquely identifies the alternate-key field.
- **FILE** specifies the key file number. Valid entries range from 0-255. The default is 0.
- **KEYLEN** specifies the length of the key. This parameter is required for creating a key-sequenced file.
- **KEYOFF** specifies the offset for the key. The default is 0.

- **[NO] NULL** specifies the null value set for the key. Valid entries are an ASCII character in quotation marks or an integer ranging from 0-255. The default is NO NULL.
- **[NO] UNIQUE** specifies whether the key is unique. The default is NO UNIQUE.
- **[NO] UPDATE** specifies whether automatic updating is set for the alternate-key file. The default is UPDATE.

[NO] AUDIT specifies whether the file will be audited by the Transaction Monitoring Facility (TMF). The default is NO AUDIT.

[NO] AUDITCOMPRESS specifies whether auditing mode is to compress or generate entire before/after messages. The default is NO AUDITCOMPRESS.

BLOCK specifies the data block length. Values range from 1-4096. The default is 1024.

[NO] BLOCKIO specifies that Connect:Direct will perform its own block I/O for high performance. With BLOCKIO specified, data is transferred in blocks determined by the file block size, with a 4K-maximum block size. Specifying BLOCKIO or NO BLOCKIO in the SYSOPTS parameters overrides the setting in the initialization parameters file.

Improving I/O performance for entry-sequenced and relative files with alternate keys must be handled differently. The alternate key is not updated if BLOCKIO is activated unless a RUN TASK statement to run the FUP LOADALTFILE utility is added to the Process. FUP is then instructed to update the alternate keys for any alternate key files.

[NO] BUFFERED specifies the mode of handling write requests. To buffer write requests into the disk-process cache, specify BUFFERED. The default for audited files is BUFFERED. The default for nonaudited files is NO BUFFERED.

BUFFERSIZE specifies the size in bytes of the internal buffer used when accessing an unstructured file. Values range from 1-4096. The default is 4096.

CODE specifies the file code. Values range from 0-65,535. Codes 100-999 are used exclusively by the system. The default is 0.

[NO] COMPRESS specifies whether keys will be compressed in both index and data blocks. In data blocks, the key offset must be 0, and the maximum record size will be reduced by 1 byte. The default is NO COMPRESS.

[NO] DCOMPRESS specifies whether keys will be compressed in data blocks. The key offset must be 0, and the maximum record size will be reduced by 1 byte. The default is NO DCOMPRESS.

[NO] ICOMPRESS specifies whether keys will be compressed in index blocks. The default is NO ICOMPRESS.

EXT specifies the size of the extents. Valid values are as follows:

- **extent.size** specifies the extent size. The default is 10.
- **pri.ext.size , sec.ext.size** specifies the sizes of the primary and secondary extents. The default is 10.

FAST.LOAD Y indicates that the FASTLOAD facility be used. FASTLOAD is a function that can reduce disk I/O overhead and is used when the Tandem node is the destination. With FASTLOAD, the Connect:Direct Tandem system passes data through the SPI interface to FUP to load into a destination data file. The feature is particularly useful for key-sequenced files, although FASTLOAD is also supported for entry-sequenced and relative record files. Because edit files are unstructured, they cannot be loaded with the FASTLOAD feature.

FAST.LOAD.PRI <priority> sets FASTLOAD and specifies the priority to run FUP. Valid values for priority range from 1-199. The default priority is the priority of the session manager (NDMSMGR). It is recommended to set this priority higher than that for NDMSMGR.

FAST.LOAD.CPU <cpu number> sets FASTLOAD and specifies the CPU to use to run FUP. Valid values for the CPU number range from 0-15. The default CPU is the CPU of the session manager (NDMSMGR).

FAST.LOAD.SORTED Y sets FASTLOAD and indicates to FUP that the data is sorted. This option (valid only for key-sequenced files) bypasses invocation of FASTSORT by FUP. The default is NO; that is, the data is not assumed to be sorted and FASTSORT is called.

SET FORMAT indicates what file format is used when the file is created. The FORMAT parameter applies only to Tandem files, when receiving, and the file is being created NEW. It is not used in a non-Tandem data set definition, when sending a file from Tandem, or when overwriting an existing file on a Tandem computer.

0 directs the Enscribe File System to set the format based on the space requested when the file is created. This is the default value.

1 requests that a Format 1 file be created. If the space requested is greater than the space supported by a Format 1 file, a file creation error occurs.

2 requests that a Format 2 file be created.

KEYLEN specifies the primary-key length. Values range from 1-255. KEYLEN must be specified to create key-sequenced files.

KEYOFF specifies the primary-key offset. Values range from 0-2034. The default is 0.

[NO] LARGEIO specifies the transfer of data in blocks of 28K. Specifying LARGEIO or NO LARGEIO in the SYSOPTS parameters overrides the setting in the initialization parameters file.

Improving I/O performance for entry-sequenced and relative files with alternate keys must be handled differently. The alternate keys are not updated if LARGEIO is activated unless a RUN TASK statement to run the FUP LOADALTFILE utility is added to the Process. FUP would then be instructed to update the alternate keys for any alternate key files.

MAXEXTENTS specifies the maximum number of extents for the file. Values range from 16-n, where n is the maximum value determined by the amount of free space

remaining in the file label. The default is 16, and the maximum value allowed is 978. For partitioned files, this value is always 16.

ODDUNSTR specifies that no upward rounding of an odd byte count will occur.

PART specifies secondary partition specifications for partitioned files. Valid values are as follows:

- **sec.partition.num** specifies the name of the volume where this secondary partition will reside. Values range from 1-15.
- **\system.name.$volume** specifies the names of the system and volume to contain the partition.
- **pri.ext.size** specifies the primary extent size. The default is 1.
- **sec.ext.size** specifies the secondary extent size. The default is 1.
- **partial.key.value** specifies the lowest key value that can reside in this partition. This value is only for key-sequenced files. Valid entries include a string of characters enclosed in double quotation marks; a list of single characters, each enclosed in double quotation marks and separated by commas; and integers representing byte values, ranging from 0-255, and separated by commas.

[NO] PARTONLY specifies whether subsequent file creations will create all partitions of a partitioned file or only a single partition. The default is NO PARTONLY.

REC specifies the length of the records. For relative and entry-sequenced files, values range from 1-4072. For DP1 key-sequenced files, values range from 1-2035. For DP2 key-sequenced files, values range from 1-4062. The default is 80. REC is not valid if the destination file is unstructured.

[NO] REFRESH specifies whether the file label will be automatically copied to disk each time the file control block is marked as *dirty.* The default is NO REFRESH.

[NO] SERIALWRITES specifies whether serial or parallel mirror writes will occur at file open. The default is NO SERIALWRITES, which will result in parallel mirror writes at file open.

TYPE specifies the file type. Values include:

- Use **U** or **0** for an unstructured file.
- Use **R** or 1 for a relative record file.
- Use **E** or **2** for an entry-sequenced file.
- Use **K** or **3** for a key-sequenced file.

[NO] VERIFIEDWRITES specifies whether disk writes will be verified. The default is NO VERIFIEDWRITES.

XLATE indicates whether the file being transferred will be converted from either ASCII to EBCDIC or EBCDIC to ASCII.

If XLATE ON or XLATE YES is specified or the file being copied is a spooler file or an edit file (unstructured file, code 101), Connect:Direct Tandem will check the XLFILE for a table named DEFAULT. If the DEFAULT table is not in XLFILE, then

Connect:Direct Tandem will use the standard English language ASCII/EBCDIC table as defined by Tandem.

For spooler or edit files: If the file being copied is a spooler file or an edit file (unstructured file, code 101), XLATE does not need to be specified because these files are automatically translated.

- **ON | YES** specifies whether text will be converted from either ASCII to EBCDIC or EBCDIC to ASCII, depending upon the copy direction.
- **OFF | NO** ensures that text conversion does not occur during file transfer.
- **table-name** is a 1-8 character name of a user-defined translation table. See the *Connect:Direct Tandem Installation and Administration Guide* for further details on user-defined translation tables.

TYPE=typekey
specifies the entry in the type defaults file containing the file attribute defaults used to open the destination file. This type key is specified only when defaults are requested by the user.

Format-Spooler Jobs

Connect:Direct Tandem supports the use of ISO/ANSI printer control characters between files on an IBM 370 node and the Tandem spooler system. These characters provide carriage control instructions to the printer. A description of these characters follows:

‘ ’ skips one line before printing a record (single-spacing).

‘0’ skips two lines before printing a record (double-spacing).

‘-’ skips three lines before printing a record (triple-spacing).

‘+’ suppresses spacing before printing a line (used for overstriking).

‘1’ begins a new page.

When transferring jobs defined as including ANSI carriage control between an IBM 370 node and the Tandem spooler system, the Connect:Direct system converts Tandem carriage control to ANSI or ANSI to Tandem carriage control as appropriate. A RECFM that includes the A characteristic must be specified on the clause of the COPY statement pertaining to the IBM 370 node.

The following formats are used to copy from and to spooler jobs on a Connect:Direct Tandem node.

Label	Statement	Parameters	
stepname	**COPY**	**FROM**	**(**
			DSN=filename | FILE=filename

Label	Statement	Parameters	
			PNODE | SNODE
			DISP=([OLD | SHR] , [KEEP | DELETE])
			SYSOPTS=(["SET SPOOLER $spooler-name"] ["SET SPOOLNUM job-number"])
			)
		TO	**(**
			DSN=filename | FILE=filename
			PNODE | SNODE
			SYSOPTS=["SET SPOOLER $spooler-name"]
			)

Field Descriptions-Spooler Jobs

COPY
identifies the statement with all its parameters and subparameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source file characteristics.

(FROM) DSN | FILE
specifies the source file name. File names are verified based on Tandem standard file name conventions.

TO
specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN | FILE
specifies the destination file name.

Optional Parameters

(FROM) DISP = ([OLD | SHR] , [KEEP | DELETE])
specifies the status of the file and its disposition after notification of successful transmission. Subparameters are as follows:

First Subparameter specifies the status of the file. This subparameter applies to all files. Options for this subparameter are as follows:

- **OLD** specifies that the source file will be opened with exclusive access.
- **SHR** specifies that the source file will be opened with shared access. SHR is the default.

Second Subparameter specifies the disposition of a normal Process step termination. Valid source file dispositions are as follows:

- **KEEP** specifies that the system will keep the file after the Process step is completed.
- **DELETE** specifies that the system will delete the file after the Process step is successfully completed.

PNODE
specifies the primary node and defines the direction of transfer (with SNODE). When PNODE is coded with the FROM parameter, the file to be copied resides on the primary node.

When PNODE is coded with the TO parameter, the file is sent to the primary node. PNODE is the default with the FROM parameter.

SNODE
specifies the secondary node and defines the direction of transfer (with PNODE). When SNODE is coded with the FROM parameter, the file to be copied resides on the secondary node.

When SNODE is coded with the TO parameter, the file is sent to the secondary node. SNODE is the default with the TO parameter.

(FROM) SYSOPTS =(
["SET SPOOLER $spooler-name"]
["SET SPOOLNUM job-number"]
)
allows you to specify system operation parameters on the Connect:Direct Tandem COPY statement. It is an alternative way of specifying file creation attributes.

Each system operation is indicated as a Tandem SET command and is enclosed in double quotation marks. You can specify multiple SET command parameters with SET preceding each parameter.

For example:

```
SET parameter
SET parameter
```

You can also specify multiple SET command parameters with SET preceding the first parameter and commas separating each parameter. For example:

```
SET parameter, parameter, parameter
```

Do not use continuation marks in the sysopts parameter. Type the text in a continuous string, with blanks separating each subparameter.

SET parameter commands define Tandem file attributes or special processing instructions for Connect:Direct Tandem. The SET commands follow in alphabetical order.

SPOOLER specifies the spooler supervisor process name used to transfer a file between the spooler and another node. The default is $SPLS. The symbol $ must precede specified spooler supervisor names.

If only the spooler supervisor name is specified and multiple instances of a spooler file name are in the spooler, the Connect:Direct system accesses the job with the highest job number corresponding to the given filename in the READY state.

SPOOLNUM specifies the job number of the spooler file. This SYSOPTS SET parameter can be used in conjunction with the spooler file name to clarify selection criteria.

Users can only access jobs that they own in the spooler.

If the Tandem system is operating with PERUSE, version T9101C20^16MAR90^IPM^T9101AAR, any user in the SUPER group can access all jobs in the spooler. If you are logged on as a user in the SUPER group and multiple jobs with the same name are in the spooler but none are owned by users in the SUPER group, the Connect:Direct system accesses the job with the highest number. If one or more of the jobs are owned by users in the SUPER group, the Connect:Direct system accesses the job with the highest job number that is owned by any user in the SUPER group.

If the Tandem system is operating with PERUSE, version T9101C20^15MAY90^IPM^T9101ABJ, you must be logged on as SUPER.SUPER or bring up the server to access all jobs in the spooler. If you are logged on as SUPER.SUPER and multiple jobs with the same name are in the spooler but none are owned by SUPER.SUPER, the Connect:Direct system accesses the job with the highest number. If one or more of the jobs are owned by SUPER.SUPER, the Connect:Direct system accesses the job with the highest job number that is owned by SUPER.SUPER.

(TO) SYSOPTS = ["SET SPOOLER $spooler-name"]

allows you to specify system operation parameters on the COPY statement. It is an alternative way of specifying file creation attributes.

SET parameter commands define Tandem file attributes or special processing instructions for Connect:Direct Tandem.

SPOOLER specifies the spooler supervisor process name used to transfer a file between the spooler and another node. The default is $SPLS. The symbol $ must precede specified spooler supervisor names.

If only the spooler supervisor name is specified and multiple instances of a spooler file name are in the spooler, the Connect:Direct system accesses the job with the highest job number corresponding to the given filename in the READY state.

Example

This example COPY statement copies a data set from a spooler file on a Connect:Direct Tandem node to a Connect:Direct OS/390 node. In this example, the specified spooler supervisor name overrides the default of $SPLS. Job=253 specifies the spooler file uniquely.

```
STEP1     COPY  FROM    (DSN=$S.#FILE2
                         DISP=SHR PNODE
                         SYSOPTS=(“SET SPOOLER=$SPLA”
                                  “SET SPOOLNUM=253"))
                TO      (DSN=MVSSITE.DESTFILE
                         DISP=NEW SNODE)
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 25

Connect:Direct Tandem RUN JOB Statement

While Connect:Direct Tandem cannot execute the RUN JOB statement, it supports a RUN JOB statement executed on the Connect:Direct OS/390 and Connect:Direct VSE nodes (RUN JOB SNODE). For RUN JOB syntax, refer to the *RUN JOB Statement* chapters for Connect:Direct OS/390 or Connect:Direct VSE.

Chapter 26

Connect:Direct Tandem RUN TASK Statement

The Connect:Direct Tandem RUN TASK statement allows user-written or system programs to be executed during a Process. RUN TASK is only valid within a Process. RUN TASK is structured so that the Process waits until the program is finished running before the next step in the Process executes.

Connect:Direct Tandem issues a return code of zero if the program initiates successfully. If the program does not start successfully, Connect:Direct generates a failure return code. See 7-2 for a list of return codes. Once the executed program begins, Connect:Direct has no control of the program and no knowledge of the outcome.

A list of parameters for the program may be specified in the RUN TASK statement. The program can run at either node involved in Process execution.

The RUN TASK statistics log records the program name, Tandem process id (PID), and the date and time the program began.

Format

The Connect:Direct Tandem RUN TASK statement is comprised of the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN TASK**	**(PGM=program-name)**
		SYSOPTS=("[/run-option parameters/] [program-parameter] [program-parameter...]")
		TIMEOUT=n
		PNODE | SNODE

Field Descriptions

RUN TASK
identifies the statement with all its parameters as the RUN TASK statement.

Required Parameters

PGM = program-name
specifies the name of the object file to be executed.

Optional Parameters

SYSOPTS = ("[/run-option parameters/]
[program-parameter]
[program-parameter...]")
specifies the parameters that are to be passed to the new Tandem process when it is created. The SYSOPTS string must be enclosed in single or double quotation marks. Use single quotation marks to allow the parsing of parameters as entered. Use double quotation marks to allow the resolution of &values in a quoted string.

Tandem run-options for the SYSOPTS parameter supported by Connect:Direct Tandem follow. For more information on these run-options, refer to the appropriate Tandem manual. Program parameters are defined by your program.

Note: There is a limit of 256 characters to the SYSOPTS string. This includes the opening and closing quotation marks and the resolved values of any symbolics. SYSOPTS strings greater than 256 characters may cause an ABEND.

CPU n specifies the CPU where the newly created Tandem process will execute. Values range from 0-15 inclusive.

IN file specifies the name of the file containing the Tandem program parameters.

INSPECT [OFF | ON | SAVEABEND] specifies the debugging environment for the Tandem process being created. OFF selects the NonStop Kernel DEBUG debugging facility. ON and SAVEABEND select INSPECT as the debugger. SAVEABEND and ON function the same except that SAVEABEND creates a dump file if the program abends. The default is OFF.

LIB filename specifies a user library file of object routines to be searched prior to the system library file to satisfy external references in the executing program.

MEM n specifies the maximum number of virtual pages to allocate for the new Tandem process. Values range from 1-64 inclusive.

NAME specifies the name of the new Tandem process, where name is a 1-5 character alphanumeric string with the first character alphabetic.

If the new Tandem process name will be used across a network, the name must be in the form $name, where name is a 1-4 character alphanumeric string.

OUT file specifies the name of the file where the output will be directed.

PRI n specifies the priority of the new Tandem process. Values range from 1-199 inclusive.

If this parameter is not specified, a priority is assigned by the program that creates the new Tandem process.

SWAP filename specifies the name of the file that will hold the virtual data for the new Tandem process. This parameter is for debugging purposes only.

TERM $termname specifies the home terminal for the new Tandem process, where termname is an alphanumeric string. The first character is alphabetic.

VOL specifies the volume and subvolume where the PGM value can be found.

PNODE
specifies that the program will be executed on the PNODE. PNODE is the default.

SNODE
specifies that the program will be executed on the secondary node (SNODE), which is the destination node.

TIMEOUT=n
specifies the number of minutes Connect:Direct allows the Process to attempt execution before the session cancels. The range is 1-32768 minutes, which is 22 days and 18 hours.

Example

This Process allows you to put a job in the spooler on hold status using symbolics. Output is sent to $S.#SYMB. The job number will be resolved at Process submission. Double quotation marks allow the resolution of the &value in a quoted string.

```
PROC1   PROCESS       PNODE=CD.TANDEM1
                      SNODE=CD.TANDEM1
STEP1   RUN TASK      (PGM=SPOOLCOM)
                      SYSOPTS=(“/OUT $S.#SYMB/JOB &JNUM,HOLD”)
                      PNODE
```

The command to submit the Process is as follows:

```
SUBMIT PROC PROC1 &JNUM=24
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 27

Connect:Direct Tandem SUBMIT Statement

During execution of a Process, the SUBMIT statement causes another Process to be submitted to either the PNODE (primary node), which is the node with Process control, or to the SNODE, which is the node that interacts with the PNODE during Process execution. The submitted Process must reside in a file on the node where the SUBMIT statement will execute. This node is referred to as the SUBNODE.

Format

The SUBMIT statement format includes the following parameters. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the SUBMIT statement format.

Label	Statement	Parameters
stepname	**SUBMIT**	**DSN=filename | FILE=filename**
		NEWNAME=new-name
		SUBNODE=PNODE | SNODE
		SNODE=secondary-node
		SNODEID=(id [,pswd] [,newpswd])
		SACCT='snode-accounting-data'
		PNODEID=(id [,pswd])
		PACCT='pnode-accounting-data'
		CLASS=n
		HOLD=Yes | No | Call
		PRTY=n
		RETAIN=Yes | No | Initial

Label	Statement	Parameters
		STARTT=([date | day][,hh:mm:ssxm])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Field Descriptions

SUBMIT
identifies the statement with all its parameters as the SUBMIT statement.

Required Parameters

DSN = filename | FILE = filename
specifies the name of the file containing the Process.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. Each logical unit (LU) has an assigned default class value, which allows a Process to execute on an LU having a matching class value or on LUs with higher class values. Class numbers are assigned in the order in which LUs appear in the network map. If a class value of 1 is specified, a Process will run on the first available LU.

If CLASS is not specified in the Connect:Direct Process or SUBMIT command, CLASS defaults to the default class specified in the PARSESS parameter of the adjacent node network map record.

HOLD = Yes | No | Call
specifies whether the Process is placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process. When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time is specified.

No specifies that the Process executes as soon as possible. HOLD=NO is the default.

Call specifies that the Process is placed in the Hold queue until a session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

NEWNAME = new-name
specifies the new name to be given to the Process. The default value is the label on the PROCESS statement.

PACCT = 'pnode-accounting-data'
specifies accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks. PACCT is used for documentation only.

PNODEID = (id [,pswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one used to sign on to a Connect:Direct node.

id specifies the Tandem group name and user name.

pswd specifies the current security password. This parameter can be used by the security exit to validate the current security password and can be 1-8 alphanumeric characters.

PRTY = n
specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ is a file that holds all Processes submitted to a Connect:Direct system. High numbers indicate high priorities; low numbers indicate low priorities.

The range is from 0-15. If PRTY is not specified, the default is the priority defined by the PRTYDEF keyword in the Connect:Direct Tandem initialization parameters. The default for the PRTYDEF keyword is 10. See the *Connect:Direct Tandem Installation and Administration Guide* for a description of the initialization parameters.

RETAIN = Yes | No | Initial
keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies that the Process remains on the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is automatically held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval.

When a Connect:Direct Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time the Connect:Direct system is initialized. Processes submitted with RETAIN=INITIAL do not execute when initially submitted. Note that STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'
specifies accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks.

SNODE = secondary-node
is a 1-16 alphanumeric character name that specifies the symbolic node name of the secondary node. The name can be expressed in alphanumerics or nationals (@ # $) with embedded periods. The first character must be alphabetic.

This parameter can override the value specified in the PROCESS statement. The default value for SNODE is the value coded in the PROCESS statement. Connect:Direct allows the PNODE and SNODE to specify the same symbolic node name.

SNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the secondary node (SNODE).

For Connect:Direct UNIX: The security user ID and passwords are case sensitive.

id specifies the Tandem group number and user number passed to the security system on the SNODE. Values range from 0-255 and are separated by a period (.).

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles may be 10 characters long.

pswd specifies the current security password. This parameter can be used by the security system on the SNODE to validate the current security password and can be 1-8 alphanumeric characters.

For OS/390, VM, and VSE nodes: The OS/390, VM, and VSE nodes only recognize passwords specified in uppercase alphanumeric characters. Therefore, a Process cannot be successfully initiated from Connect:Direct OS/390, VM, or VSE nodes with Connect:Direct Tandem unless the Connect:Direct Tandem SNODEID password follows the same convention (no lowercase characters and no control characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

For Connect:Direct OS/400: This subparameter is ignored.

STARTT = ([date | day] [,hh:mm:ssXM])

specifies that the Process will execute at a specified date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

date specifies that the Process is to be held until the desired date. The day (dd), month (mm), and year (yy) can be specified in one of the following formats:

yymmdd	mm/dd/yy	yy/mm/dd	mm.dd.yy
yy.mm.dd	mmddyy	yyddd (Julian date)	yy/ddd (Julian date)

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

day specifies the day of the week that the Process will be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value may be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week.

If day is specified with no time of day, the time defaults to 00:00. If a Process is submitted on Monday with Monday as the only STARTT parameter, the Process will not run until the following Monday the time is 00:00:00.

You can also specify TODAY, which releases the Process for execution the day of Process submission, or TOMORROW, which releases the Process for execution the next day.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

You can also specify TODAY, which releases the Process for execution the day and time of Process submission (unless the time of day is specified), or TOMORROW, which releases the Process for execution the next day. If a time of day is not specified with TOMORROW, the Process executes after midnight.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT, which releases the Process for execution at midnight.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time is specified.

SUBNODE = PNODE | SNODE

specifies the node where the Process defined in this SUBMIT statement will execute. Specifying PNODE means that the Process is submitted on the node that has Process control. Specifying SNODE means that the Process is submitted on the node participating in, but not controlling, Process execution. In both cases, the Process must reside on the node on which it is being submitted. The default is PNODE.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. You can override this default when submitting the Process.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters can be enclosed in single quotation marks.

Note that the symbolic parameter for the SUBMIT statement must begin with a single ampersand. This allows submission of the Process that contains the SUBMIT statement to resolve symbolic parameters correctly.

An ampersand symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Example

This Process copies USER1T.TEXT to $B.TANDEM.FILE at the SNODE (CD.TAN.NODE). If $B.TANDEM.FILE already exists, the file will be replaced. Upon completion of STEP01, $B.TANDEM.FILE will run at the SUBNODE (CD.TAN.NODE).

```
PROCESS1     PROCESS       PNODE=CD.OS390.NODE SNODE=CD.TAN.NODE
                           PRTY=1
STEP01       COPY  FROM    (DSN=USER1T.TEXT DISP=SHR PNODE)
                   TO      (DSN=$B.TANDEM.FILE DISP=RPL SNODE)
STEP02       SUBMIT        DSN=$B.TANDEM.FILE
                           PRTY=5
                           SUBNODE=SNODE
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 28

Connect:Direct Tandem SYMBOL Statement

The Connect:Direct Tandem SYMBOL statement allows you to build symbolic substitution values.

Format

The Connect:Direct Tandem SYMBOL statement format includes the following parameters. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**SYMBOL**	**&symbolic_name=variable-string**

Field Descriptions

SYMBOL
identifies the statement with all its parameters as the SYMBOL statement.

Required Parameters

&symbolic_name=variable-string
specifies the string that is substituted into the Process.

When Connect:Direct software encounters an ampersand (&) plus 1-17 alphanumeric characters, Connect:Direct substitutes a string represented by that ampersand and the alphanumeric characters.

Symbols in the string are resolved from previously specified values in a PROCESS, SUBMIT, or SYMBOL statement. With the SYMBOL statement, different pieces of a Connect:Direct statement string can be concatenated, allowing the user to move data in a variety of ways.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters can be enclosed in single quotation marks.

Example

This example demonstrates using the SYMBOL statement in a Connect:Direct Process to build symbolic substitution values that are resolved at Process submission.

The example also incorporates the use of symbols that are stored in memory in the Connect:Direct Tandem environment. These symbols are also resolved at Process submission. See the discussion of SYMBOL commands in the *Connect:Direct Tandem NonStop Kernel User Guide and Reference* for the command syntax for inserting, deleting, and displaying symbols in the Connect:Direct Tandem environment.

You can use this generic Process for many different types of file transfers. The Process is constructed with commonly used parameters that can be overridden when the Process is submitted. It eliminates the need for maintaining separate Processes that perform similar functions.

Assume that the following symbols are loaded in Connect:Direct Tandem memory with the INSERT SYMBOL environment command.

```
SYMBOL          PARAMETERS
----------------------------------------
SNODE           CD.TANDEM2
SDISP           SHR
DDISP           RPL
SVOL            $B
SSV             SUBVOLA
DVOL            $C
DSV             SUBVOLB
FROM            &svol..&ssv..&id
TO              &dvol..&dsv..&id
```

Assume that the file, $C.PROC.P1, contains the following Process:

```
&PROC    PROCESS       PNODE=CD.TANDEM1
                       SNODE=&SNODE
         SYMBOL        &COPY=&PROC || -C1
&COPY    COPY  FROM    (DSN=&FROM DISP=&SDISP)
               TO      (DSN=&TO DISP=&DDISP)
```

To submit the Process, issue the following SUBMIT PROCESS command:

```
SUBMIT PROCESS $C.PROC.P1 &PROC=ABCD &ID=FILE1 &DVOL=$D
```

A symbol entered as part of the SUBMIT PROCESS command overrides the value of the same symbol stored in memory.

After resolution of the symbolics, the Process is as follows:

```
ABCD     PROCESS      PNODE=CD.TANDEM1
                      SNODE=CD.TANDEM2
         SYMBOL       &COPY=ABCD-C1
ABCD-C1  COPY  FROM   (DSN=$B.SUBVOLA.FILE1 DISP=SHR)
               TO     (DSN=$D.SUBVOLB.FILE1 DISP=RPL)
```

Symbolics are resolved as follows:

- The label for the PROCESS statement (&PROC) resolves to ABCD.
- The secondary node (&SNODE) resolves to CD.TANDEM2.
- The label for the COPY statement (&COPY) resolves to ABCD-C1.
- The source file (&FROM) resolves to $B.SUBVOLA.FILE1.
- The source disposition (&SDISP) resolves to SHR.
- The destination file (&TO) resolves to $D.SUBVOLB.FILE1.
- The destination disposition (&DDISP) resolves to RPL.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 29

Connect:Direct Tandem Conditional Statements

Conditional statements permit you to alter the sequence of Process execution based on the completion of the previous step in the Process.

Formats

Formats for Connect:Direct conditional statements follow. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	IF	(label condition nn) THEN
		(process steps)
	ELSE	
		(alternative process steps)
	EIF	
optional	GOTO	label
	EXIT	

Statement Descriptions

IF THEN

specifies that the Connect:Direct system executes a block of Connect:Direct statements based on the completion code of a Process step. An EIF statement must be used in conjunction with an IF THEN statement. See the *Field Descriptions* section for details.

ELSE
designates a block of Connect:Direct statements that execute when the IF THEN condition is not satisfied. No parameters exist.

EIF
is required for specifying the end of the IF THEN or IF THEN ELSE block of statements. No parameters exist.

GOTO
moves to a specific step within a Process. See the following *Field Descriptions* section for details on using the step label.

EXIT
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

label
For the **IF THEN** statement, the label specifies the name of a previous step whose completion code is used for comparison.

For the **GOTO** statement, the label specifies the name of a subsequent step in a Process (required for GOTO only). The name can neither be the label of a preceding step nor the label of the GOTO statement of which it is a part.

Note: User-defined labels must begin in column one. The label consists of a 1-8 character alphanumeric string, with the first letter alphabetic only.

condition
specifies the type of comparison to be performed. This condition checking is based on comparisons for equality, inequality, greater than, less than, greater than or equal to, and less than or equal to.

Valid symbols, alternate symbols, and conditions follow:

= or EQ specifies that the completion code must be equal to the value nn for the condition to be satisfied.

<> or ¬= or NE specifies that the completion code must not equal the value nn for the condition to be satisfied.

>= or ¬< or GE specifies that the completion code must be greater than or equal to the value nn for the condition to be satisfied.

> or GT specifies that the completion code must be greater than the value nn for the condition to be satisfied.

<= or ¬> or LE specifies that the completion code must be less than or equal to the value nn for the condition to be satisfied.

< or LT specifies that the completion code must be less than the value nn for the condition to be satisfied.

nn

specifies the numeric value to be used for completion code checking. If coded as X‘nn’, it is a hexadecimal value; any other coding indicates it as decimal.

If a completion code less than 4 is returned, the Process completed successfully. A return code greater than 4 indicates the Process ended in error. Note that a return code equaling 4 indicates a warning.

THEN

specifies subsequent processing to be performed if the condition specified is true.

Example

This example Process copies a file from Tandem to VM. It uses conditional statements to check the completion code from STEP01. If the completion code equals 0, then a message is issued to the operator that indicates that the transfer was successful. If the completion code is any value other than 0, then a message is issued to the operator that indicates that the transfer failed.

```
PROC01     PROCESS     SNODE=CD.VM
                       PNODE=CD.TANDEM
STEP01     COPY  FROM  (FILE=$B.ABC.FILEA PNODE
                       SYSOPTS='SET XLATE ON')
                 TO    DSN=JKL.FILEA
                       DISP=(RPL), LINK=(CDVM,MULT,MW,551))
STEP02     IF          (STEP01 EQ 0) THEN
NOTIFY1    RUN TASK    (PGM=DMNOTFY2
                      SYSOPTS=("GOOD,'TRANS FOR COMPANY A COMPLETED DISK 502',
                       OPERATOR"))
                       SNODE
           EXIT
           ELSE
NOTIFY2    RUN TASK    (PGM=DMNOTFY2
                       SYSOPTS=("FAIL,'TRANS FOR COMPANY A FAILED DISK 502',
                       OPERATOR"))
                       SNODE
           EIF
           EXIT
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 30

Considerations for Connect:Direct OS/400 Users

Connect:Direct OS/400 uses CL commands instead of the Connect:Direct Process language to initiate processes using LU6.2 or TCP/IP sessions to remote Connect:Direct nodes. The Connect:Direct OS/400 statements described in the following chapters are used in Processes where Connect:Direct OS/400 is the remote node.

Connect:Direct OS/390, VSE, VM, and Tandem NonStop Kernel users can submit Processes to connect to Connect:Direct OS/400 using SNA LU0 sessions. Connect:Direct OS/390, VM, UNIX, VSE, and Windows users can submit Connect:Direct Processes to initiate SNA LU6.2 independent or dependent LU sessions. Connect:Direct OS/390, VSE, OpenVMS, Tandem, UNIX, and Windows users can submit Processes to connect to Connect:Direct OS/400 using TCP/IP.

When writing a Process for a platform, use the PROCESS, SYMBOL, and conditional statements for that operating environment. Use the COPY FROM and COPY TO formats from the appropriate FROM and TO platforms. Use the RUN JOB and RUN TASK format of the platform on which the job will execute. For example, to run a program on OS/390, you will use the RUN JOB statement for Connect:Direct OS/390.

Connect:Direct OS/400 CL commands are used to initiate connections to remote nodes running Connect:Direct using LU6.2 sessions or TCP/IP. See the *Connect:Direct OS/400 User's Guide* for a description of CL commands, their syntax and parameters.

Chapter 31

Connect:Direct OS/400 COPY Statement

The Connect:Direct OS/400 COPY statement is used in a Process initiated on a non-OS/400 node to copy files, members, objects, and spooled files to or from an OS/400 node.

General Format Information

The Connect:Direct OS/400 COPY statement consists of a COPY FROM clause that includes the source object name and a COPY TO clause that includes the destination object name. Additional parameters can be specified to further customize the file transfer operation.

The following pages provide the format of the COPY statement for files, members, objects, and spooled files. All allowed parameters are listed, along with their possible values. Following each format is a definition of each parameter and its values.

Format-Files

Use the file format to copy a physical database file to and from a Connect:Direct OS/400 node.

Label	Statement	Parameters	
stepname	**COPY**	**FROM**	**(**
			DSN='library-name/file-name' | '/directory/file-name' | '/QLANSrv/file-name' | '/QDLS/folder-name' | '/QOpenSys/file-name'

Label	Statement	Parameters	
			SYSOPTS="TYPE(FILE) **PRECMPR (*YES | *NO)** **EXITCMD(valid OS/400 command)** **FAILCMD(valid OS/400 command)** **SNDFFD(*YES | *NO)** **TEXTFILE(*YES | *NO)** **XTRAN (table-name)** **[XTRANLSO (so-code) | XTRANLSI (si-code) |** **XTRANLDATA (MIXED | DBCS)]"**
			DISP=([SHR | OLD] ,[KEEP | DELETE] ,[KEEP | DELETE])
			EXCLUDE=(generic | member | (start-range/stop-range) | list)
			REPLACE | NOREPLACE
			SELECT=(member | generic | (*)) | (member, [new-name], [NR | R]) | (generic,, [NR | R]) (start-range/stop-range,,[NR | R]) | (list)
			)
		TO	**(**
			DSN='library-name/file-name' | '/directory/file-name' | **'/QLANSrv/file-name' | '/QDLS/folder-name' |** **'/QOpenSys/file-name'**
			SNODE (remote-node-name)
			SYSOPTS="TYPE(FILE) **DECMPR (*YES | *NO)** **EXITCMD(valid OS/400 command)** **FAILCMD(valid OS/400 command)** **TEXTFILE(*YES | *NO)** **RCDLEN(record-length)** **FILETYPE(*SRC | *DATA)** **TEXT('text-description')** **EXPDATE(expiration-date)** **MAXMBRS(number | *NOMAX)** **SIZE(#-of-recs |** **incr-value#-of-incrs|*NOMAX)** **AUT(*CHANGE | *ALL | *USE | *EXCLUDE)** **IGCDTA(*YES | *NO)"**
			DISP=([NEW | OLD | MOD | RPL | SHR])
			UNIT=(unit-identifier)

Label	Statement	Parameters
		)
		CKPTINV=[nnnnnnn | nnnnnnK | nnnnnnM]
		COMPRESS [[PRIMEchar = X‘xx’ | X‘40’ | C‘cc’] | EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n)

General Information on File Support

A file is an object that contains a set of related records grouped as members. A file must have one or more members.

When copying a file to or from a Connect:Direct OS/400 node, the user can specify whether:

- An entire file is to be sent.
- One specific file member is to be sent.
- Certain members are selected for transmission (SELECT parameter).
- Certain members are excluded from transmission (EXCLUDE parameter).
- A member is renamed (NEWNAME parameter).
- Members replace existing members of the same name at the receiving node (REPLACE and NOREPLACE parameters, R and NR subparameters of the SELECT parameter).

File Guidelines

Refer to the following guidelines when copying a file:

- EXCLUDE and SELECT cannot be used if the FROM DSN or TO DSN contains a member name.
- The hierarchy for the SELECT and EXCLUDE parameters follows, and proceeds from top to bottom (highest override priority to lowest):
  - Exclude by member name
  - Select by member name
  - Exclude by generic (or range)
  - Select by generic (or range)
  - Combine various specifications in a list after the EXCLUDE parameter
- If EXCLUDE is specified and SELECT is not specified, all members not excluded are copied. If EXCLUDE is not specified and SELECT is specified, only selected members are copied.
- When copying to a file, all members are sent unless one of the following conditions exists:

- The SELECT option is specified.
- The EXCLUDE option is specified.
- A member name is specified.

Additional Considerations

When copying a file from a Connect:Direct OS/400 node to a PDS at a Connect:Direct OS/390 node, OS/400 allows member names to be 10 characters. Because an OS/390 member name can only be eight characters long, the ninth and tenth characters are truncated.

This factor affects a file with members having names with the same first eight characters. For example:

Name of Member	Creation Date
membrnamea	08/11/91
membrnameb	08/01/91

Both member names will be truncated to **membrnam**. The member name **membernam** did not previously exist on OS/390. The last **membrnam** overwrites the previous **membrnam** unless NO REPLACE (NR) is specified.

Field Descriptions-Files

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source object characteristics.

(FROM) DSN='library-name/file-name' |'directory/file-name' | '/QLANSrv/file-name' | '/QDLS/folder-name' | '/QOpenSys/file-name'
specifies the source name. File names are verified by the OS/400 standard file name conventions. The entire DSN must be in single quotation marks.

'library-name/file-name' specifies the library and file name to be copied using the native file system.

‘/directory/file-name’|‘/QLANSrv/file-name’|‘/QDLS/folder-name’|‘/QOpenSys/file-name’ specifies the directory and file names to be copied using the integrated file system.

Directories like /QOpenSys and /root are case-sensitive.

TO

specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN=‘library-name/file-name’ |‘directory/file-name’ | ‘/QLANSrv/file-name’ | ‘/QDLS/folder-name’ | ‘/QOpenSys/file-name’

specifies the destination file name. File names are verified based on the OS/400 standard file name conventions. The entire DSN must be in single quotation marks.

‘library-name/file-name’ specifies the library and file name to be copied. The name of the source file is used for the file name in the destination unless the SELECT parameter is specified otherwise.

‘/directory/file-name’|‘/QLANSrv/file-name’|‘/QDLS/folder-name’|‘/QOpenSys/file-name’ specifies the directory and file names to be copied using the integrated file system.

Directories like /QOpenSys and /root are case-sensitive.

(FROM) SYSOPTS = “TYPE(FILE)
PRECMPR (*YES | *NO)
EXITCMD(valid os/400 command)
FAILCMD(valid os/400 command)
SNDFFD(*YES | *NO)
TEXTFILE(*YES | *NO)
XTRAN (table-name)
[XTRANLSO (so-code) | XTRANLSI (si-code) |
XTRANLDATA (MIXED | DBCS)]"

specifies system operation parameters on the Connect:Direct OS/400 COPY statement. This parameter is required. The maximum number of characters for SYSOPTS is 256.

All SYSOPTS keyword values must be enclosed in parentheses. The entire SYSOPTS string must be enclosed in double quotation marks. For example: “TYPE(FILE)”

When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in parentheses. The subparameters must be separated by blanks. For example:

```
SYSOPTS="TYPE(FILE) PRECMPR(*YES) XTRAN(EBCXKSC) XTRANLDATA(MIXED)"
```

TYPE(FILE) specifies that the file being copied is a physical database file.

PRECMPR(*YES | *NO) specifies whether the file being sent has previously been compressed using the CDCOMP command. To send a file that has been compressed, you must specify the PRECMPR(*YES) parameter with the CDSND command. The default is PRECMPR(*NO).

EXITCMD(valid os/400 command) specifies a command to be executed only if the copy process is successful.

Note: The EXITCMD subparameter for (FROM) SYSOPTS is only valid when the sender is running Connect:Direct OS/400 Version 3.3 or higher.

FAILCMD(valid os/400 command) specifies a command to be executed only if the copy process is not successful.

Note: The FAILCMD subparameter for (FROM) SYSOPTS is only valid when the sender is running Connect:Direct OS/400 Version 3.3 or higher.

SNDFFD(*YES | *NO) specifies whether the sending node will transfer file field descriptions.

Use the following formula to determine whether a file can be sent with the file field descriptions. If the bytes required value is greater than 8100, you cannot transfer the file field descriptions.

```
(24 x number of keys) + (92 x number of fields) + 256 = bytes required
```

Note: You cannot use this parameter with the integrated file system.

TEXTFILE(*YES | *NO) specifies that the file being sent is a text file. This keyword is for the OS/400 only.

XTRAN(table-name) specifies the extended translation table to use. The named table object must exist in a library that is in the library list of the session manager job. If the library name is not in the list, the COPY step fails.

If the XTRAN keyword is present, then the following related optional keywords may also be used:

XTRANLSO (so-code) specifies extended translate shift out code. This keyword specifies the hex code to use for the shift out character and overrides the default value of 0E. You can specify any two valid hex digits.

XTRANLSI (si-code) specifies extended translate shift in code. This keyword specifies the hex code to use for the) shift in character and overrides the default value of 0F. You can specify any two valid hex digits.

XTRANLDATA (MIXED | DBCS) specifies extended translate local data format.

- **MIXED** indicates that the data may contain both DBCS and SBCS characters and that SO/SI characters are used. MIXED is the default.
- **DBCS** indicates that the data is pure DBCS characters and that no SO/SI characters are used.

The following rules apply to the use of the XTRAN keyword:

- You must specify the XTRAN keyword to use extended translation; all other keywords are optional.

- The default for local shift-out is the IBM standard x0E.
- The default for local shift-in is the IBM standard x0F.
- The default local data format is MIXED. With SO/SI in use, XTRAN is *not* allowed with PRECMPR(*YES) or DECMPR(*NO)

When sending a file from the AS/400, you are required to specify a binary transfer on the receiving node. See the COPY statement for the appropriate receiving node for instructions on specifying a binary transfer.

(TO) SYSOPTS = "TYPE(FILE)
DECMPR (*YES | *NO)
EXITCMD(valid os/400 command)
FAILCMD(valid os/400 command)
TEXTFILE(*YES | *NO)
RCDLEN(record-length)
FILETYPE(*SRC | *DATA)
TEXT('text-description')
EXPDATE(expiration-date)
MAXMBRS(number | *NOMAX)
SIZE(#-of-recs incr-value #-of-incrs | *NOMAX)
AUT(*CHANGE | *ALL | *USE | *EXCLUDE)
IGCDTA(*YES | *NO)"

specifies system operation parameters on the Connect:Direct OS/400 COPY statement. The maximum number of characters permitted for SYSOPTS is 256.

When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in parentheses. The subparameters must be separated by blanks. For example:

```
SYSOPTS="TYPE(FILE) DECMPR(*YES)"
```

TYPE(FILE) specifies that the data is to be copied to the Connect:Direct OS/400 node as a physical database file. This parameter is required.

DECMPR(*YES | *NO) specifies whether the receiving Connect:Direct node is to decompress the received file. This parameter is valid only when the receiving system is a Connect:Direct OS/400 node.

- ***NO** instructs the receiving system to place the receiving data in a database file without decompressing it.

EXITCMD(valid os/400 command) specifies a command to be executed only if the copy process is successful.

Note: The EXITCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

FAILCMD(valid os/400 command) specifies a command to be executed only if the copy process is not successful.

Note: The FAILCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

TEXTFILE(*YES | *NO) specifies that the file being received is a text file. This keyword is for the OS/400 only.

RCDLEN(record-length) specifies the data length for each record in the file. If FILETYPE(*SRC), valid values range from 1-32754; if FILETYPE(*DATA), valid values range from 1-32766. This parameter is used if a physical database file is created to hold the data received.

When transmitting stream data in text mode, always specify this parameter to avoid possible space allocation problems. If this is a new file and RCDLEN has not been specified, the attributes of the source data are used to determine an acceptable record length.

Note: If the Connect:Direct system creates a physical source file, it uses a record length 12 bytes longer than the value specified for RCDLEN. These 12 bytes allow for the 6-byte sequence number field and 6-byte data field that precede the data in each record of the member.

FILETYPE(*SRC | *DATA) specifies the type of file to be created. This parameter is used whenever a file is created.

- ***SRC** indicates that a physical source database file is to be created.
- ***DATA** indicates that a physical database file is to be created.

TEXT('text description') specifies a text description to be associated with this member (and file, if created). This description cannot exceed 50 characters and must be enclosed in single quotation marks.

EXPDATE(expiration-date) specifies the date after which the new or replaced member cannot be used. If EXPDATE is not specified, then the file does not have an expiration date. The format you must use is dependent upon the system value QDATFMT. For example, if QDATFMT=MDY, you must input the expiration date in MMDDYY form. To display the system value QDATFMT, type 'DSPSYSVAL QDATFMT' on the OS/400 command line.

MAXMBRS(number | *NOMAX) specifies the maximum number of members a physical file can contain. This parameter is used when a file is being created with this COPY statement. If *NOMAX is specified, then the system maximum of 32,767 members per file is used. The default is *NOMAX.

SIZE(#-of-recs incr-value #-of-incrs | *NOMAX) is used when a new file is created for the member received.

- **#-of-recs** indicates the initial number of records for a member. Valid values range from 1-16777215. The default is 10000.
- **incr-value** indicates the number of records in each increment to be added to a member size if the initial space allocated is depleted. Valid values range from 1-32767, the default is 1000. If 0 is specified, the member is not allowed extensions.
- **#-of-incrs** indicates the number of times the increment is automatically applied. Valid values range from 0-32767. The default is 10.

- ***NOMAX** indicates the number of records for a member is limited by the system, not the user.

AUT(<u>*CHANGE</u> | *ALL | *USE | *EXCLUDE) specifies the authority to be given to a user who does not have specific authority to the file or member, is not on the authorization list, and whose user group does not have specific authority to the file or member.

- **<u>*CHANGE</u>** is the default. It grants a user object operational and all data authorities.
- ***ALL** grants a user object operational, object management, and object existence authorities and all data authorities.
- ***USE** grants a user object operational and read data authority.
- ***EXCLUDE** prevents any user (other than the owner) from accessing the file.

For more information on AS/400 security, refer to the IBM *OS/400 Data Management Guide and Security Concepts and Planning* manuals.

IGCDTA(*YES | *NO) specifies that double-byte character set (DBCS) support is installed on the destination AS/400 system and the file being sent contains DBCS data. This parameter is only valid if the destination AS/400 system is configured for DBCS support. Including this parameter in a Process coded to send a file to an AS/400 system without DBCS support will cause an error and termination of the Process.

SNODE
is the secondary node.

Optional Parameters

CKPTINV = [nnnnnnn | nnnnnnK | nnnnnnM]
specifies the number of bytes, from 0 to 2 gigabytes, to send before taking a checkpoint. The format is nnnnnnn, nnnnnnK, or nnnnnnM, where K specifies thousands of bytes and M specifies millions of bytes. A value of 0 specifies no checkpoint.

A checkpoint interval specified here overrides the initialization parameter default value.

Checkpointing will not occur in the following cases, even if you specify a checkpoint interval:

- The file being sent is compressed during the send operation by the local node using the extended compression feature, and decompression is deferred on the receiving node. That is, the (TO) SYSOPTS parameter has DECMPR(*NO) specified.
- A precompressed file is sent and decompressed by the receiving node during the send operation. That is, the (FROM) SYSOPTS has PRECMPR(*YES) specified and (TO) SYSOPTS has DECMPR(*YES) specified.
- A file is sent to a single OS/390 partitioned data set member.

If you request checkpointing when you are transferring multiple members of a file, checkpoints are taken only at member boundaries regardless of the interval specified in the initialization parameters or in the CDSND or CDRCV command.

COMPRESS [[PRIMEchar = X‘xx’ | X‘40’ | C‘cc’] | EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n)]

specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at the destination. The default for the COMPRESS parameter is PRIMEchar=X‘40’.

Note: Compression is CPU-intensive and its effectiveness is data dependent. It should only be used if its benefits are known.

If compression is specified, Connect:Direct reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character are compressed to 1 byte.
- Repetitive occurrences (ranging from 3-63) of any other character are compressed to 2 bytes.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X‘40’).

EXTended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited and data is repetitive.

ECCLEVEL(n) specifies the extended compression level, which affects how much CPU the extended compression routines will use. Higher compression levels use more CPU but achieve greater compression. The valid values for this subparameter are 1-9, inclusive. The default value is specified in the initialization parameter.

ECWINSIZE(n) specifies the extended compression window size, which is specifically for the history buffer that is filled from the user's input buffer (both compressing and decompressing). The window specifies the amount of storage designated to maintain data previously read.

This data can be scanned for string matches. The extended compression window size affects how much virtual memory the extended compression routines will use. Higher window size values use more memory but achieve greater compression. The valid values for this subparameter are 8-15, inclusive. The default value is specified in the initialization parameter.

ECMEMLEVEL(n) specifies the extended compression memory level parameter, which determines how much memory should be allocated for other internal data structures like the hash table and the previous table (pointers to previous strings starting with the same 3 characters). The extended memory level affects how much memory the extended compression routines will use. Higher

memory levels use more virtual memory but achieve greater compression. The valid values for this subparameter are 1-9, inclusive. The default value is specified in the initialization parameter.

(FROM) DISP = ([SHR | OLD] ,[KEEP | DELETE] ,[KEEP | DELETE])
specifies the status of the file and what is to be done with the file after notification of successful or unsuccessful transmission. Subparameters are as follows:

First Subparameter specifies the status of the object. Options for this subparameter are as follows:

- **SHR** specifies that the member can be read simultaneously by another job or Process. The default is SHR.
- **OLD** specifies that the Process is to be given exclusive control of the file.

Second Subparameter specifies the disposition of the file following a normal Process step termination, resulting in a zero completion code. Valid dispositions are as follows:

- **KEEP** specifies that the system keeps the member after the Process step ends.
- **DELETE** specifies that the system deletes the member after the Process step ends.

Third Subparameter specifies the disposition of the file after an abnormal Process step termination, resulting in a nonzero completion code. Valid source file dispositions are as follows:

- **KEEP** specifies that the system keeps the member after the Process step terminates abnormally.
- **DELETE** specifies the system deletes the member if the Process step terminates abnormally.

(TO) DISP = ([NEW | OLD | MOD | RPL | SHR])
specifies the status of the data on the receiving node. Subparameters are as follows:

NEW specifies that the Process step will create the destination member. The file can be an existing file, but the member cannot already exist. NEW is the default. The file is created if it does not exist. The copy fails if the file exists.

OLD specifies that the destination file already exists. The Process will have exclusive control of the member. The copy fails if the member does not exist.

MOD specifies that the Process step will modify the file by adding data to the end of the file or if none exists will allocate a new file. The file is created if it does not exist.

RPL specifies that the destination file will replace any existing member or if none exists will allocate a new member. The file will be created if it does not exist.

SHR specifies that the destination file already exists. The file can be read simultaneously by another job or Process. If the destination file does not exist, the copy fails.

EXCLUDE = (generic | member | (start-range/stop-range) | list)
specifies criteria that identifies the file members that are not to be copied. The EXCLUDE parameter can be specified only in the FROM clause of the COPY state-

ment. EXCLUDE allows the user to make exceptions to members specified generically or by range in the SELECT option. See page 251 for the override priority for the SELECT and EXCLUDE parameters.

generic specifies a generic member name. For example, if CD0* is specified, all member names beginning with CD0 are excluded. The only way to override an excluded generic is to specify an individual member name in the SELECT parameter.

member specifies an individual member name. When a member is specified in the EXCLUDE parameter, its exclusion cannot be overridden.

start-range specifies the first name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the EXCLUDE statement, the first and last members specified in the range, as well as all members between, are not copied.

stop-range specifies the last name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the EXCLUDE statement, the first and last members specified in the range, as well as all members between, are not copied.

Note: The only way to override an excluded range is to specify an individual member name in the SELECT parameter.

---

list specifies a list of member names.

(FROM) REPLACE | NOREPLACE

specifies that members of a sending file replace or do not replace existing members of the same name at the receiving file.

REPLACE specifies that members of the sending file replace members of the same name at the receiving file. REPLACE is the default.

NOREPLACE specifies that members of the sending file do not replace members of the same name at the receiving file. The NOREPLACE parameter takes effect only when copying from a file to a file. Note that NOREPLACE applies to an entire file as opposed to the NR option of the SELECT parameter, which applies to members within a file.

SELECT = (member | generic | (*) | (member, [newname] ,[NR | R]) | (generic,, [NR | R]) (start-range/stop-range,,[NR | R]) | list)

specifies selection criteria by which file members are to be copied. The SELECT parameter can be specified only with the FROM parameter. See page 251 for the override priority for the SELECT and EXCLUDE parameters.

Various specifications can be combined in a list after the SELECT keyword.

If SELECT is specified and EXCLUDE is not specified, all selected members are copied. If SELECT is not specified and EXCLUDE is specified, all members not excluded are copied.

When a generic is specified in the SELECT parameter, its selection can be overridden with any type of specification in the EXCLUDE parameter. When using a generic and specifying NR or R, the second positional parameter (NEWNAME) must be null.

generic specifies a generic member name. If CD0* is specified as either a parameter or as a subparameter, all member names beginning with CD0 are selected for copying.

(*) represents a global generic. A global generic indicates that all members of the file are to be included. A global generic is valid only with the SELECT keyword.

member specifies an individual member name. Note that specifying only one member name is the same as specifying TYPE(MBR) in the SYSOPTS parameter of the COPY statement.

The only way to override a selection by member name is to specify that member name in the EXCLUDE parameter.

newname specifies a new name for a member. The NEWNAME parameter must be null if a generic name or range is used in the first subparameter position.

NR specifies that a member does not replace an existing member of the same name at the receiving file. NR overrides the REPLACE parameter. R is the default.

When used with NEWNAME, NR applies to the NEWNAME and not to the original member name. When used with a generic name or with a range, NR applies to all members selected for that criteria.

Note: NOREPLACE applies to an entire file as opposed to NR, which applies to members within a file.

R specifies that a member replaces an existing member of the same name at the receiving file. R overrides the NOREPLACE parameter.

When used with NEWNAME, R applies to the NEWNAME and not to the original member name. When used with a generic name or with a range, R applies to all members selected for that criteria.

start-range specifies the first name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the SELECT statement, the first and last members specified in the range, as well as all members between, are copied.

stop-range specifies the last name in an alphanumeric range of members. Although member names in a range are treated as generics, they cannot be used with an asterisk (*). A slash (/) separates the first (start-range) and last (stop-range) member names. When used with the SELECT statement, the first and last members specified in the range, as well as all members between, are copied.

When a range in the SELECT parameter is specified, its selection can be overridden with any type of specification in the EXCLUDE parameter.

The second positional parameter (NEWNAME) of SELECT must be null when using a range and specifying NR or R.

list specifies a list of selected members.

(TO) UNIT = (unit-identifier)

specifies the unit identifier of the auxiliary storage unit on which the storage space for the file, and file members, is to be allocated.

unit-identifier can be any value from 1-255. If not specified, storage space is allocated on any available auxiliary storage unit.

Format-Member

This format is used to copy a physical database file member to and from a Connect:Direct OS/400 node.

Label	Statement	Parameters	
stepname	**COPY**	**FROM**	**(**
			DSN='library-name/file-name' | **'library-name/file-name(member-name)' |** **'/QSYS.LIB/library-name.LIB'/** **file-name.FILE/member-name.MBR' |**
			SYSOPTS="TYPE(MBR) **PRECMPR (*YES | *NO)** **EXITCMD(valid OS/400 command)** **FAILCMD(valid OS/400 command)** **SNDFFD(*YES|*NO)** **TEXTFILE(*YES|*NO)** **XTRAN (table-name)** **[XTRANLSO (so-code) | XTRANLSI (si-code) |** **XTRANLDATA (MIXED | DBCS)]"**
			DISP=([SHR | OLD] ,[KEEP | DELETE] ,[KEEP | DELETE])
			)
		TO	**(**
			DSN='library-name/file-name' | **'library-name/file-name(member-name)' |** **'/QSYS.LIB/library-name.LIB/** **file-name.FILE/member-name.MBR' |**
			SNODE (receiving-node-name)

Label	Statement	Parameters	
			SYSOPTS="TYPE(MBR) **DECMPR (*YES | *NO)** **EXITCMD(valid OS/400 command)** **FAILCMD(valid OS/400 command)** **TEXTFILE(*YES | *NO)** **RCDLEN(record-length)** **FILETYPE(*SRC | *DATA)** **TEXT('text-description')** **EXPDATE(expiration-date)** **MAXMBRS(number | *NOMAX)** **SIZE(#-of-recs | incr-value#-of-incrs|*NOMAX)** **AUT(*CHANGE | *ALL | *USE | *EXCLUDE)** **IGCDTA(*YES | *NO)"**
			DISP=([NEW | OLD | MOD | RPL | SHR])
			UNIT=(unit-identifier)
			)
			CKPTINV=[nnnnnnn | nnnnnnK | nnnnnnM]
			COMPRESS [[PRIMEchar = X'xx' | X'40' | C'cc'] | EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n)]

Field Descriptions-Members

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source object characteristics.

(FROM) DSN ='library-name/file-name' | 'library-name/file-name(member-name)'| /QSYS.LIB/library-name.LIB/file-name.FILE/member-name.MBR'
specifies the source member name. Member names are verified based on the OS/400 standard name conventions. The entire DSN must be in single quotation marks.

'library-name/file-name' specifies the library and file name of the member to be copied. The file name is used as the member name.

'library-name/file-name(member-name)' specifies the library, file, and member name of the member to be copied. The member name is only required if it is different from the file name.

'/QSYS.LIB/library-name.LIB/file-name.FILE/member-name.MBR' specifies the library, file name, and member name to be copied using the integrated file system.

Directories like /QOpenSys and /root are case-sensitive. The /QSYS.LIB is case-sensitive only when you enclose the name in single quotation marks.

TO

specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN = 'library-name/file-name' | 'library-name/file-name(member-name)'| '/QSYS.LIB/library-name.LIB/file-name.FILE/member-name.MBR'

specifies the destination object name. Object names are verified based on the OS/400 standard file name conventions. The entire DSN must be in single quotation marks.

'library-name/file-name' specifies the library and file name of the member to be copied using the native file system. The file name is used as the member name.

'library-name/file-name(member-name)' specifies the library, file, and member names of the member to be copied using the native file system. The member name is only required if it is different from the file name.

'/QSYS.LIB/library-name.LIB/file-name.FILE/member-name.MBR' specifies the library, file name, and member name to be copied using the integrated file system.

Directories like /QOpenSys and /root are case-sensitive. The /QSYS.LIB is case-sensitive only when you enclose the name in single quotation marks.

(FROM) SYSOPTS = "TYPE(MBR)
PRECMPR (*YES | *NO)
EXITCMD(valid os/400 command)
FAILCMD(valid os/400 command)
SNDFFD(*YES | *NO)
TEXTFILE(*YES | *NO)
XTRAN (table-name)
[XTRANLSO (so-code) | XTRANLSI (si-code) |
XTRANLDATA (MIXED | DBCS)]

specifies system operation parameters on the Connect:Direct OS/400 COPY statement. This parameter is required. The maximum number of characters for SYSOPTS is 256.

When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in parentheses. The subparameters must be separated by blanks. For example:

```
SYSOPTS="TYPE(MBR) PRECMPR(*YES) XTRAN(EBCXKSC) XTRANLDATA(MIXED)"
```

TYPE(MBR) specifies that the data being copied is a member of a physical database file.

PRECMPR(*YES | *NO) specifies whether the file being sent has previously been compressed using the CDCOMP command. To send a file that has been compressed, you must specify the PRECMPR(*YES) parameter with the CDSND command. The default is PRECMPR(*NO).

EXITCMD(valid os/400 command) specifies a command to be executed only if the copy process is successful.

Note: The EXITCMD subparameter for (FROM) SYSOPTS is only valid when the sender is running Connect:Direct OS/400 Version 3.3 or higher.

FAILCMD(valid os/400 command) specifies a command to be executed only if the copy process is not successful.

Note: The FAILCMD subparameter for (FROM) SYSOPTS is only valid when the sender is running Connect:Direct OS/400 Version 3.3 or higher.

SNDFFD(*YES | *NO) specifies whether the sending node will transfer file field descriptions.

Use the following formula to determine whether a file can be sent with the file field descriptions. If the bytes required value is greater than 8100, you cannot transfer the file field descriptions.

```
(24 x number of keys) + (92 x number of fields) + 256 = bytes required
```

Note: You cannot use this parameter with the integrated file system.

TEXTFILE(*YES | *NO) specifies that the file being sent is a text file. This keyword is for the OS/400 only.

XTRAN(table-name) specifies the extended translation table to use. The named table object must exist in a library that is in the library list of the session manager job. If the library name is not in the list, the COPY step fails.

If the XTRAN keyword is present, then the following related optional keywords may also be used:

XTRANLSO (so-code) specifies extended translate local shift out code. This keyword specifies the hex code to use for the shift out character and overrides the default value of 0E. You can specify any two valid hex digits.

XTRANLSI (si-code) specifies extended translate local shift in code. This keyword specifies the hex code to use for the shift in character and overrides the default value of 0F. You can specify any two valid hex digits.

XTRANLDATA (MIXED | DBCS) specifies extended translate local data format.

- **MIXED** indicates that the data may contain both DBCS and SBCS characters and that SO/SI characters are used. MIXED is the default.
- **DBCS** indicates that the data is pure DBCS characters and that no SO/SI characters are used.

The following rules apply to the use of the XTRAN keyword:

- You must specify the XTRAN keyword to use extended translation; all other keywords are optional.

- The default for local shift-out is the IBM standard x0E.
- The default for local shift-in is the IBM standard x0F.
- The default local data format is MIXED. With SO/SI in use, XTRAN is *not* allowed with PRECMPR(*YES) or DECMPR(*NO)

When sending a member from the AS/400, you are required to specify a binary transfer on the receiving node. See the COPY statement for the appropriate receiving node for instructions on specifying a binary transfer.

(TO) SYSOPTS = "TYPE(MBR)
DECMPR (*YES | *NO)
EXITCMD(valid os/400 command)
FAILCMD(valid os/400 command)
TEXTFILE(*YES | *NO)
RCDLEN(record-length)
FILETYPE(*SRC | *DATA)
TEXT('text-description')
EXPDATE(expiration-date)
MAXMBRS(number | *NOMAX)
SIZE(#-of-recs incr-value #-of-incrs | *NOMAX)
AUT(*CHANGE | *ALL | *USE | *EXCLUDE)
IGCDTA(*YES | *NO)"

specifies system operation parameters on the Connect:Direct OS/400 COPY statement. The maximum number of characters permitted for SYSOPTS is 256.

When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in parentheses. The subparameters must be separated by blanks. For example:

```
SYSOPTS="TYPE(MBR) DECMPR(*YES)"
```

TYPE(MBR) specifies that the data is to be copied to the Connect:Direct OS/400 node as a member of a physical database file. This parameter is required.

DECMPR(*YES | *NO) specifies whether the receiving Connect:Direct node is to decompress the received file. This parameter is valid only when the receiving system is a Connect:Direct OS/400 node.

- ***NO** instructs the receiving system to place the receiving data in a database file without decompressing it.

EXITCMD(valid os/400 command) specifies a command to be executed only if the copy process is successful.

Note: The EXITCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

FAILCMD(valid os/400 command) specifies a command to be executed only if the copy process is not successful.

Note: The FAILCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

TEXTFILE(*YES | *NO) specifies that the file being received is a text file. This keyword is for the OS/400 only.

RCDLEN(record-length) specifies the data length for each record in the file. If FILETYPE(*SRC), valid values range from 1-32754; if FILETYPE(*DATA), valid values range from 1-32766. This parameter is used if a physical database file is created to hold the data received. *RCDLEN is required for VSAM files.*

When transmitting stream data in text mode, always specify this parameter to avoid possible space allocation problems. If this is a new file and RCDLEN has not been specified, the attributes of the source data are used to determine an acceptable record length.

Note: If Connect:Direct creates a physical source file, it uses a record length 12 bytes longer than the value specified for RCDLEN. For example, a 2,000 byte record creates a physical source file with a 2,012 byte record length. These 12 bytes allow for the 6-byte sequence number field and 6-byte data field that precede the data in each record of the member.

FILETYPE(*SRC | *DATA) specifies the type of file to be created. This parameter is used whenever a file is created.

- ***SRC** indicates that a physical source database file is to be created.
- ***DATA** indicates that a physical database file is to be created.

TEXT('text description') specifies a text description to be associated with this member (and file, if created). This description cannot exceed 50 characters and must be enclosed in single quotation marks.

EXPDATE(expiration-date) specifies the date after which the new or replaced member cannot be used. If EXPDATE is not specified, then the file does not have an expiration date. The format you must use is dependent upon the system value QDATFMT. For example, if QDATFMT=MDY, you must input the expiration date in MMDDYY form. To display the system value QDATFMT, type 'DSPSYSVAL QDATFMT' on the OS/400 command line.

MAXMBRS(number | *NOMAX) specifies the maximum number of members a physical file can contain. This parameter is used when a file is being created with this COPY statement. If *NOMAX is specified, then the system maximum of 32,767 members per file is used. The default is 1.

SIZE(#-of-recs incr-value #-of-incrs | *NOMAX) is used when a new file is created for the member received.

- **#-of-recs** indicates the initial number of records for a member. Valid values range from 1-16777215. The default is 10000.
- **incr-value** indicates the number of records in each increment to be added to a member size if the initial space allocated is depleted. Valid values range from 1-32767, the default is 1000. If 0 is specified, the member is not allowed extensions.

- **#-of-incrs** indicates the number of times the increment is automatically applied. Valid values range from 0-32767. The default is 10.
- ***NOMAX** indicates the number of records for a member is limited by the system, not the user.

AUT(*CHANGE | *ALL | *USE | *EXCLUDE) specifies the authority to be given to a user who does not have specific authority to the file or member, is not on the authorization list, and whose user group does not have specific authority to the file or member.

- ***CHANGE** is the default. It grants a user object operational and all data authorities.
- ***ALL** grants a user object operational, object management, and object existence authorities and all data authorities.
- ***USE** grants a user object operational and read data authority.
- ***EXCLUDE** prevents any user (other than the owner) from accessing the file.

For more information on OS/400 security, refer to the IBM *OS/400 Data Management Guide and Security Concepts and Planning* manuals.

IGCDTA(*YES|*NO) specifies that double-byte character set (DBCS) support is installed on the destination AS/400 system and the file being sent contains DBCS data. This parameter is only valid if the destination AS/400 system is configured for DBCS support. Including this parameter in a Process coded to send a file to an AS/400 system without DBCS support will cause an error and termination of the Process.

SNODE
is the secondary node.

Optional Parameters

CKPTINV = [nnnnnnn | nnnnnnK | nnnnnnM]
specifies the number of bytes, from 0 to 2 gigabytes, to send before taking a checkpoint. The format is nnnnnnn, nnnnnnK, or nnnnnnM, where K specifies thousands of bytes and M specifies millions of bytes. A value of 0 specifies no checkpoint. Connect:Direct OS/400 uses the value you specify, rounded up to the nearest record boundary, to determine when to take a checkpoint.

A checkpoint interval specified here overrides the initialization parameter default value.

Checkpointing will not occur in the following cases, even if you specify a checkpoint interval:

- The file being sent is compressed during the send operation by the local node using the extended compression feature, and decompression is deferred on the receiving node. That is, the (TO) SYSOPTS parameter has DECMPR(*NO) specified.

- A precompressed file is sent and decompressed by the receiving node during the send operation. That is, the (FROM) SYSOPTS has PRECMPR(*YES) specified and (TO) SYSOPTS has DECMPR(*YES) specified.
- A file is sent to a single OS/390 partitioned data set member.

If you request checkpointing when you are transferring multiple members of a file, checkpoints are taken only at member boundaries regardless of the interval specified in the initialization parameters or in the CDSND or CDRCV command.

COMPRESS [[PRIMEchar = X‘xx’ | X‘40’ | C‘cc’] | EXTended= ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n)]

specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at the destination. The default subparameter for the COMPRESS parameter is PRIMEchar=X‘40’.

Note: Compression is CPU-intensive and its effectiveness is data dependent. It should only be used if its benefits are known.

If compression is specified, Connect:Direct reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character are compressed to 1 byte.
- Repetitive occurrences (ranging from 3-63) of any other character are compressed to 2 bytes.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X‘40’).

EXTended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited and data is repetitive.

ECCLEVEL(n) specifies the extended compression level, which affects how much CPU the extended compression routines will use. Higher compression levels use more CPU but achieve greater compression. The valid values for this subparameter are 1-9, inclusive. The default value is specified in the initialization parameter.

ECWINSIZE(n) specifies the extended compression window size, which is specifically for the history buffer that is filled from the user's input buffer (both compressing and decompressing). The window specifies the amount of storage designated to maintain data previously read.

This data can be scanned for string matches. The extended compression window size affects how much virtual memory the extended compression routines will use. Higher window size values use more memory but achieve greater compression. The valid values for this subparameter are 8-15, inclusive. The default value is specified in the initialization parameter.

ECMEMLEVEL(n) specifies the extended compression memory level parameter, which determines how much memory should be allocated for other internal data structures like the hash table and the previous table (pointers to previous strings starting with the same 3 characters). The extended memory level affects how much memory the extended compression routines will use. Higher memory levels use more virtual memory but achieve greater compression. The valid values for this subparameter are 1-9, inclusive. The default value is specified in the initialization parameter.

(FROM) DISP =([SHR | OLD] ,[KEEP | DELETE] ,[KEEP | DELETE])
specifies the status of the file and what is to be done with the file after notification of successful or unsuccessful transmission. Subparameters are as follows:

First Subparameter specifies the status of the object. Options for this subparameter are as follows:

- **SHR** specifies that the member can be read simultaneously by another job or Process. The default is SHR.
- **OLD** specifies that the Process is to be given exclusive control of the file.

Second Subparameter specifies the disposition of the file following a normal Process step termination, resulting in a zero completion code. Valid dispositions are as follows:

- **KEEP** specifies that the system is to keep the member after the Process step ends.
- **DELETE** specifies that the system is to delete the member after the Process step ends.

Third Subparameter specifies the disposition of the file after an abnormal Process step termination, resulting in a nonzero completion code. Valid source file dispositions are as follows:

- **KEEP** specifies that the system is to keep the member after the Process step terminates abnormally.
- **DELETE** specifies the system is to delete the member if the Process step terminates abnormally.

(TO) DISP = ([NEW | OLD | MOD | RPL | SHR])
specifies the status of the data on the receiving node. Subparameters are as follows:

NEW specifies that the Process step will create the destination member. The file can be an existing file, but the member cannot already exist. NEW is the default. The copy fails if the member exists.

OLD specifies that the destination member already exists. The Process will have exclusive control of the member. The copy fails if the member does not exist.

MOD specifies that the Process step will modify the member by adding data to the end of the member, or, if none exists, will allocate a new member. The member is created if it does not exist.

RPL specifies that the destination member will replace any existing member or if none exists will allocate a new member. The member will be created if it does not exist.

SHR specifies that the destination member already exists. The file can be read simultaneously by another job or Process. If the destination member does not exist, the copy fails.

(TO) UNIT = (unit-identifier)
specifies the unit identifier of the auxiliary storage unit on which the storage space for the file, and file members, is to be allocated.

unit-identifier can be any value from 1-255. If not specified, storage space is allocated on any available auxiliary storage unit.

Format-Objects

This format is used to copy one or more objects to and from a Connect:Direct OS/400 node. The object must be in save file format on the OS/400 prior to copying. You can view the save file information by displaying the save files on the OS/400.

Label	Statement	Parameters	
stepname	**COPY**	**FROM**	**(**
			DSN='library-name/save-file-name'|
			SNODE
			SYSOPTS="TYPE(OBJ)"
			EXITCMD(valid OS/400 command) **FAILCMD(valid OS/400 command)**
			DISP=([SHR] ,[KEEP | DELETE] ,[KEEP | DELETE])
			)
		TO	**(**
			DSN='library-name/save-file-name'
			SNODE
			SYSOPTS="TYPE(OBJ)
			EXITCMD(valid OS/400 command) **FAILCMD(valid OS/400 command)**
			MAXRCDS(number | *NOMAX) **ASP(auxiliary-storage-pool)**
			TEXT('text-description')
			AUT(*EXCLUDE | *CHANGE | *ALL | *USE)"
			DISP=([NEW | OLD | MOD | RPL | SHR])
			)
			COMPRESS [[PRIMEchar = X'xx' | X'40' | C'cc'] | EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n)]

Field Descriptions-Objects

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source object characteristics.

(FROM) DSN = 'library-name/save-file-name'
specifies the source save file name. File names are verified based on the OS/400 standard file name conventions. This parameter is required.

'library-name/save-file-name' specifies the library and name of the save file to be copied.

TO
specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN = library-name/save-file-name'
specifies the destination save file name. This parameter is required.

library-name/object-name specifies the library and name of the destination save file.

(FROM) SYSOPTS = "TYPE(OBJ)"
EXITCMD(valid os/400 command)
FAILCMD(valid os/400 command)
specifies system operation parameters on the Connect:Direct OS/400 COPY statement. The maximum number of characters permitted for SYSOPTS is 256.

All SYSOPTS keyword values must be enclosed in parentheses. The entire SYSOPTS string must be enclosed in double quotation marks. For example: "TYPE(OBJ)"

TYPE(OBJ) specifies that the object being copied is to be transferred in save file format. This parameter is required.

EXITCMD(valid os/400 command) specifies a command to be executed only if the copy process is successful.

Note: The EXITCMD subparameter for (FROM) SYSOPTS is only valid when the sender is running Connect:Direct OS/400 Version 3.3 or higher.

FAILCMD(valid os/400 command) specifies a command to be executed only if the copy process is not successful.

Note: The FAILCMD subparameter for (FROM) SYSOPTS is only valid when the sender is running Connect:Direct OS/400 Version 3.3 or higher.

(TO) SYSOPTS =“TYPE(OBJ)
EXITCMD(valid os/400 command)
FAILCMD(valid os/400 command)
MAXRCDS(number | *NOMAX),
ASP(auxiliary-storage-pool),
TEXT(‘text description’),
AUT(*EXCLUDE | *CHANGE | *ALL | *USE)"

specifies system operation parameters on the Connect:Direct OS/400 COPY statement. The maximum number of characters permitted for SYSOPTS is 256.

When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in parentheses. The subparameters must be separated by blanks. For example:

```
SYSOPTS="TYPE(OBJ) MAXRCD(*NOMAX) AUT(EXCLUDE)"
```

TYPE(OBJ) specifies that the object is to be copied to the Connect:Direct OS/400 node and is assumed to be in save file format. This parameter is required.

EXITCMD(valid os/400 command) specifies a command to be executed only if the copy process is successful.

Note: The EXITCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

FAILCMD(valid os/400 command) specifies a command to be executed only if the copy process is not successful.

Note: The FAILCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

MAXRCDS(number | *NOMAX) specifies the maximum number of records the save file, which was created to hold the data received, can reach. If the number of records received exceeds this value, the COPY step ends in error. Valid values for this parameter range from 1-3997574. If MAXRCDS is not specified, the system limits the size of the save file.

Note: Two thousand 512-byte records require approximately 1 megabyte of space. To ensure that the save file will not exceed approximately 20 megabytes, specify 40000 (20x2000) for MAXRCDS.

ASP(auxiliary-storage-pool) specifies the auxiliary storage pool from which the system allocates storage for the save file. Valid values range from 1-16. The default is 1.

TEXT(‘text description’) specifies a text description to be associated with this object. This description cannot exceed 50 characters and must be enclosed in single quotation marks.

AUT(*EXCLUDE | *CHANGE | *ALL | *USE) specifies the authority to be given to a user who does not have specific authority to the object, is not on the authorization list, and whose user group does not have specific authority to the object.

- ***EXCLUDE** is the default. It prevents a user from accessing the file.
- ***CHANGE** grants a user object operational and all data authorities.
- ***ALL** grants a user object operational, object management, and object existence authorities and all data authorities.
- ***USE** grants a user object operational and read data authority.

For more information on AS/400 security, refer to the IBM *OS/400 Data Management Guide and Security Concepts and Planning* manuals.

SNODE

is the secondary node.

Optional Parameters

COMPRESS [[PRIMEchar = X‘xx’ | X‘40’ | C‘cc’] | EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n)]

specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at the destination. The default subparameter for the COMPRESS parameter is PRIMEchar=X‘40’.

Note: Compression is CPU-intensive and its effectiveness is data dependent. It should only be used if its benefits are known.

If compression is specified, Connect:Direct reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character are compressed to 1 byte.
- Repetitive occurrences (ranging from 3-63) of any other character are compressed to 2 bytes.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X‘40’).

EXTended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited and data is repetitive.

ECCLEVEL(n) specifies the extended compression level, which affects how much CPU the extended compression routines will use. Higher compression levels use more CPU but achieve greater compression. The valid values for this subparameter are 1-9, inclusive. The default value is specified in the initialization parameter.

ECWINSIZE(n) specifies the extended compression window size, which is specifically for the history buffer that is filled from the user's input buffer (both compressing and decompressing). The window specifies the amount of storage designated to maintain data previously read.

This data can be scanned for string matches. The extended compression window size affects how much virtual memory the extended compression routines will use. Higher window size values use more memory but achieve greater compression. The valid values for this subparameter are 8-15, inclusive. The default value is specified in the initialization parameter.

ECMEMLEVEL(n) specifies the extended compression memory level parameter, which determines how much memory should be allocated for other internal data structures like the hash table and the previous table (pointers to previous strings starting with the same 3 characters). The extended memory level affects how much memory the extended compression routines will use. Higher memory levels use more virtual memory but achieve greater compression. The valid values for this subparameter are 1-9, inclusive. The default value is specified in the initialization parameter.

(FROM) DISP =([SHR] ,[KEEP | DELETE] ,[KEEP,DELETE])

specifies the status of the object and what is to be done with the object after notification of successful or unsuccessful transmission. Subparameters are as follows:

First Subparameter specifies the status of the object. The only allowed subparameter is SHR.

- **SHR** specifies that the object can be read simultaneously by another job or Process.

Second Subparameter specifies the disposition of the object following a normal Process step termination, resulting in a zero completion code. Valid dispositions are as follows:

- **KEEP** specifies that the system keeps the object after the Process step ends. KEEP is the default.
- **DELETE** specifies that the system deletes the object after the Process step ends.

Third Subparameter specifies the disposition of the object after an abnormal Process step termination, resulting in a nonzero completion code. Valid source object dispositions are as follows:

- **KEEP** specifies that the system keeps the object after the Process step terminates abnormally. KEEP is the default.
- **DELETE** specifies that the system deletes the object if the Process step terminates abnormally.

(TO) DISP = ([NEW | OLD | MOD | RPL | SHR])

specifies the status of the data on the receiving node. Subparameters are as follows:

NEW specifies that the Process step will create the destination object. The object cannot already exist. NEW is the default. The copy fails if the object exists.

OLD specifies that the destination object already exists. The Process has exclusive control of the object. The copy fails if the object does not exist.

MOD specifies that the Process step will modify the object by adding data to the end of the object. The object is created if it does not exist.

RPL specifies that the destination object will replace any existing object or if none exists will allocate a new object. The object is created if it does not exist.

SHR specifies that the object can be read simultaneously by another job or Process. If the destination member does not exist, the copy fails.

Format-Spooled Files

This format is used to copy to a spooled file on a Connect:Direct OS/400 node.

Label	Statement	Parameters	
		TO	
			(DSN=spooled-file-name
			SNODE

Label	Statement	Parameters	
			SYSOPTS="TYPE(SPLF) EXITCMD(valid OS/400 command) FAILCMD(valid OS/400 command) DEV(*JOB|*SYSVAL|device-name) DEVTYPE(*IPDS|*SCS) PAGESIZE(page-length page-width) LPI(3|4|6|7.5|8|9) CPI(5|10|12|13.3|15|16.7|18|20) OVRFLW(overflow-line-number) FOLD(*NO|*YES) RPLUNPRT(*YES 'replacement-character'|*NO) ALIGN(*NO|*YES) CTLCHAR(*NONE|*FCFC) CHLVAL(*NORMAL| (channel#1line#1) (channel#2line#2) (channel#3line#3) (channel#4line#4) (channel#5line#5) (channel#6line#6) (channel#7line#7) (channel#8line#8) (channel#9line#9) (channel#10line#10) (channel#11line#11) (channel#12line#12) FORMFEED(*DEVD|*CONT|*CUT|*AUTOCUT) PRTQLTY(*STD|*DRAFT|*DEVD|*NLQ) DRAWER(1|2|3|*E1) FONT(*CPI|*DEVD|font-identifier) CHRID(*DEVD|*SYSVAL|graphic-character-set code-page) PAGRTT(*DEVD|*COR|0|90|180|270) PRTTXT('print-text') JUSTIFY(0|50|100) DUPLEX(*NO|*YES|*TUMBLE) SPOOL(*YES|*NO) OUTQ(*JOB|*DEV|library-name/output-queue-name) FORMTYPE(form-type) COPIES(number-of-copies) MAXRCDS(maximum-records) FILESEP(number-of-file-separators) HOLD(*YES|*NO) SAVE(*YES|*NO) OUTPTY(*JOB|output-priority) USRDTA(user-data)")

Field Descriptions-Spooled Files

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

TO
specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO)DSN = spooled-file-name
specifies the destination spooled output file name. This parameter is required.

spooled-file-name is the name of the destination spooled output file. The name can be up to 10 characters long and must be in single quotation marks if it contains any special characters.

```
(TO) SYSOPTS = “TYPE(SPLF)
        EXITCMD(valid os/400 command)
        FAILCMD(valid os/400 command)
        DEV(*JOB | *SYSVAL | device-name)
        DEVTYPE(*IPDS | *SCS)
        PAGESIZE(page-length page-width,
        LPI(3 | 4 | 6 | 7.5 | 8 | 9)
        CPI(5 | 10 | 12 | 13.3 | 15 | 16.7 | 18 | 20)
        OVRFLW(overflow-line-number)
        FOLD(*NO | *YES)
        RPLUNPRT(*YES  ‘replacement-character’ | *NO)
        ALIGN(*NO | *YES)
        CTLCHAR(*NONE | *FCFC)
        CHLVAL(*NORMAL|
        (channel#1  line#1)
        (channel#2  line#2)
        (channel#3  line#3)
        (channel#4  line#4)
        (channel#5  line#5)
        (channel#6  line#6)
        (channel#7  line#7)
        (channel#8  line#8)
        (channel#9  line#9)
        (channel#10 line#10)
        (channel#11 line#11)
        (channel#12 line#12))
        FORMFEED(*DEVD|*CONT|*CUT|*AUTOCUT,
        PRTQLTY(*STD | *DRAFT | *DEVD | *NLQ)
        DRAWER(1 | 2 | 3 | *E1)
        FONT(*CPI | *DEVD | font-identifier)
        CHRID(*DEVD | *SYSVAL | graphic-character-set  code-page)
        PAGRTT(*DEVD | *COR | 0 | 90 | 180 | 270)
        PRTTXT(‘print-text’)
```

JUSTIFY(0 | 50 | 100)
DUPLEX(*NO | *YES | *TUMBLE)
SPOOL(*YES | *NO)
OUTQ(*JOB | *DEV | library-name/output-queue-name)
FORMTYPE(form-type)
COPIES(number-of-copies)
MAXRCDS(maximum-records)
FILESEP(number-of-file-separators)
HOLD(*YES | *NO)
SAVE(*YES | *NO)
OUTPTY(*JOB | output-priority)
USRDTA(user-data)"

specifies system option parameters on the Connect:Direct OS/400 COPY statement. The name of the Connect:Direct printer device file is NDMPRINT. This file is created by the installation process. The defaults used by Connect:Direct for the SYSOPTS subparameters are taken from this file. The SYSOPTS subparameters are used to override the defaults. The maximum number of characters permitted for SYSOPTS is 256.

When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in parentheses. The subparameters must be separated by blanks. For example:

```
SYSOPTS="TYPE(SPLF) FORMFEED(*AUTOCUT) JUSTIFY(50)"
```

Note: If you regularly override the printer or spooled file attributes with SYSOPTS, you may want to modify the NDMPRINT printer device file with the OS/400 CL command CHGPRTF (Change Printer File) to use the options.

TYPE(SPLF) specifies that the data is copied to an OS/400 spooled output file. This parameter is required.

EXITCMD(valid os/400 command) specifies a command to be executed only if the copy process is successful.

Note: The EXITCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

FAILCMD(valid os/400 command) specifies a command to be executed only if the copy process is not successful.

Note: The FAILCMD subparameter for (TO) SYSOPTS is only valid when the receiver is running Connect:Direct OS/400 Version 3.3 or higher.

DEV(*JOB|*SYSVAL|device-name) specifies the name of the printer device description.

- ***JOB** indicates the printer used by the Connect:Direct job is to be used as the printer device.
- ***SYSVAL** indicates that the value in the OS/400 system value QPRTDEV is to be used as the printer device.
- **device-name** identifies the printer device used for nonspooled output with the printer device to produce the printed output. For spooled output, if

OUTQ(*DEVD), the default output queue for the specified printer is used for the spooled output data.

DEVTYPE(*IPDS | *SCS) specifies the type of data stream created for a printer file from the data received. This parameter indicates whether the resulting data stream is an Intelligent Printer Data Stream (IPDS) or an SNA Character Stream (SCS).

- ***IPDS** indicates an IPDS is to be created.
- ***SCS** indicates an SCS is to be created.

PAGESIZE(page-length page-width) specifies the length and width of the page used by the printer. This parameter overrides the FORMTYPE parameter.

- **page-length** is in lines per page.
- **page-width** is in characters per line.

LPI(3 | 4 | 6 | 7.5 | 8 | 9) specifies the line space setting (lines per inch) on the printer.

CPI(5 | 10 | 12 | 13.3 | 15 | 16.7 | 18 | 20) specifies the printer character density, in characters per inch, for the printer.

For the printers that support fonts, the value specified for font implies the CPI. If FONT(*CPI) is specified, the font used is based on the CPI value. The following table shows fonts based on CPI value.

CPI	Corresponding Font
5.0	245
10.0	011
12.0	087
13.3	204
15.0	222
16.7	400
18.0	252
20.0	281

OVRFLW(overflow-line-number) specifies the line number on the page at which overflow to a new page begins. The value specified must not exceed the forms length specified for PAGESIZE.

FOLD(*NO | *YES) specifies whether entire records are printed when the record length exceeds the form width. If DEVTYPE(*IPDS), then this parameter is ignored and records are truncated.

- ***NO** indicates that records are truncated if they exceed the form width.
- ***YES** indicates records wrap to the next line or lines until the entire record is printed.

RPLUNPRT(*YES ‘replacement-character’ | *NO) specifies whether unprintable characters are replaced with printable characters when printed. The replacement character is specified as well, separated by the *YES and a single blank. Any printable EBCDIC character can be specified as a replacement character.

ALIGN(*NO | *YES) specifies whether the page must be aligned in the printer before printing is started.

CTLCHAR(*NONE | *FCFC) specifies whether the data contains printer control characters.

- ***NONE** indicates that the data does not contain printer control characters.
- ***FCFC** indicates that the first character of each record contains an ANSI forms-control character. Any incorrect control characters are ignored, and single spacing is assumed. This subparameter should be used when the source OS/390 file is RECFM=xxA, which indicates it contains ANSI carriage control.

CHLVAL(*NORMAL| (channel#1 line#1) (channel#2 line#2) (channel#3 line#3) (channel#4 line#4) (channel#5 line#5) (channel#6 line#6) (channel#7 line#7) (channel#8 line#8) (channel#9 line#9) (channel#10 line#10) (channel#11 line#11) (channel#12 line#12)) specifies the list of channel numbers with their assigned line numbers. CTLCHAR(*FCFC) must be specified as part of the SYSOPTS parameter for this to be valid.

- ***NORMAL** indicates that channel 1 causes a skip to the next line, and channel 12 causes a skip to the overflow line (OVERFLOW parameter). Channels 2-11 cause a space-one-line operation.
- **(channel#1 line#1) ...(channel#12 line#12)** Any combination of channel numbers, 1 through 12, may be specified along with a line number to be assigned to that channel number. Valid line numbers range from 1-255. If no line number is assigned to a channel number and that channel number is found in the data, a default of space-one-line before printing is taken. Each channel and line number may be specified once.

FORMFEED(*DEVD | *CONT | *CUT | *AUTOCUT) specifies the form feed attachments used by the printer (4214, 5219, and 5553 printers only).

- ***DEVD** indicates that forms are fed according to the printer device description.
- ***CONT** indicates that continuous forms are used by the printer.
- ***CUT** indicates that single-cut sheets are used by the printer.
- ***AUTOCUT** indicates that single-cut sheets are fed semi-automatically into the printer. The sheet-feed attachment must be on the printer.

PRTQLTY(*STD | *DRAFT | *DEVD | *NLQ) specifies the quality of print produced.

- ***STD** indicates standard quality.
- ***DRAFT** indicates draft quality.
- ***DEVD** indicates the print quality is set on the printer by the user, not in the data stream.
- ***NLQ** indicates near letter quality.

DRAWER(1 | 2 | 3 | *E1) specifies the source drawer to be used when automatic cut-sheet feed mode is used. FORMFEED(*AUTOCUT) must be specified as part of SYSOPTS for this to be valid.

- **1 | 2 | 3** indicate drawer number 1, 2, or 3 on the sheet-feeder paper handler.
- ***E1** indicates that envelopes are to be fed from the envelope drawer on the sheet-feeder paper handler.

FONT(*CPI | *DEVD | font-identifier) specifies the font identifier to be used for the spooled output file.

- ***CPI** indicates that the value specified in the CPI parameter is to be used to determine the font. See the CPI parameter described on page 280 for corresponding fonts.
- ***DEVD** indicates that the font specified in the device description for the printer is to be used.
- **font-identifier** indicates that a user-specified font identifier has been supplied. Any valid 3- or 4-digit font identifier is allowed.

CHRID(*DEVD | *SYSVAL | graphic-character-set code-page) specifies the character identifier to use for the spooled output file. This parameter allows you to print data that is in different character identifier coding. The value specified is used to command the printer device to interpret the hexadecimal byte stream by printing the same characters intended when the text was created.

- ***DEVD** indicates that the CHRID value the device is designed to handle is used.
- ***SYSVAL** indicates that the CHRID specified for the system on which Connect:Direct is running is used.
- **graphic-character-set code-page** indicates that the user is supplying the graphic-character-set and code-page. Any value ranging from 1-32767 may be specified for both.

PAGRTT(*DEVD | *COR | 0 | 90 | 180 | 270) specifies the degree of rotation (clockwise from the edge of the paper first loaded into the printer) of text on each page printed.

- ***DEVD** indicates that forms are rotated according to the hardware switches on the printer.
- ***COR** indicates that computer output reduction is done when the output is printed.
- **0 | 90 | 180 | 270** indicates the specific angle of rotation.

PRTTXT('print-text') specifies a line of text to be printed at the bottom of each page printed. Up to 30 characters enclosed in single quotation marks may be specified.

JUSTIFY(0 | 50 | 100) controls the print positions of the characters on the page (in the spooled file) so the right margin is regular.

- **0** indicates that no justification occurs.
- **50** indicates that blanks between words are padded to obtain a more closely aligned right margin (not flush).

- **100** indicates that the text is padded with spaces to obtain an even right margin.

DUPLEX(*NO | *YES | *TUMBLE) specifies whether the spooled output file is printed on one or both sides of the paper.

- ***NO** indicates that the file prints on one side of the paper.
- ***YES** indicates that the file prints on both sides of the paper, with the top of each printed page at the same end of the paper.
- ***TUMBLE** indicates that the file prints on both sides of the paper, with the top of each page printed at opposite ends of the paper.

SPOOL(*YES | *NO) specifies whether the data is sent to a spooled file prior to printing.

- ***YES** indicates the file is sent to a spooled file for processing by a print-writer.
- ***NO** indicates that the file is not spooled but sent directly to the device specified for print.

OUTQ(*JOB | *DEV|library-name/output-queue-name) specifies the output queue for the spooled output file created. OUTQ is valid only when SPOOL(*YES).

- ***JOB** indicates that the output queue specified for Connect:Direct is to be used.
- ***DEV** indicates that the output queue associated with the device specified by the DEV parameter is to be used.
- **library-name/output-queue-name** allows the user to specify a qualified name for the output queue for the spooled output file created.

FORMTYPE(form-type) specifies the type of form to use in the printer when the spooled file is printed. A form-type identifier is user-defined and is no longer than 10 characters.

If FORMTYPE *STD is specified, the standard form for a particular computer system is used when the spooled output file is printed.

COPIES(number-of-copies) specifies the number of copies to be printed. Valid values range from 1-255. This parameter is only valid when SPOOL(*YES).

MAXRCDS(maximum-records) specifies the maximum number of records that can be placed in the output queue. Valid values range from 1-500000. This parameter is only valid when SPOOL(*YES).

If MAXRCDS is not specified, then the number of records that can be placed on the output queue is limited to 100,000.

FILESEP(number-of-file-separators) specifies the number of blank separator pages to be placed between each copy of the file printed.

HOLD(*YES | *NO) specifies whether the file is to be held on the output queue until released by the user. This parameter is only valid if SPOOL(*YES).

SAVE(*YES | *NO) specifies whether the spooled output file is to be saved on the output queue once printed. This parameter is only valid if SPOOL(*YES).

OUTPTY(*JOB | output-priority) specifies the scheduling priority of the file on the output queue. This parameter is only valid if SPOOL(*YES).

- ***JOB** indicates that the priority is to be determined by the output priority associated with the Connect:Direct job.
- **output-priority** indicates a user-defined priority of 1 (high) to 9 (low).

USRDTA(user-data) allows up to 10 characters of data to identify the spooled output file. This parameter is only valid if SPOOL(*YES).

Optional Parameters

SNODE
is the secondary node. The Connect:Direct OS/400 node is always the SNODE. The SNODE is the default and is included in the Process as documentation only.

Example

In this example COPY statement, the member FILEA in PDS USER.TESTLIB on the OS/390 system is copied to the member TEST in the TESTFILES/PROCLIB file on the Connect:Direct OS/400 node. If the member TEST already exists in the file TESTFILES/PROCLIB on the Connect:Direct OS/400 node, it is replaced by this version.

```
/* COPY FROM: SOURCE DATA SET ATTRIBUTES  (OS/390) */
STEP01     COPY   FROM  (DSN=USER.TESTLIB(FILEA)
                        PNODE
                        DISP=SHR
                        )
/* COPY TO: DESTINATION DATA SET ATTRIBUTES (AS/400) */
                   TO    (SNODE
                        DSN='TESTFILES/PROCLIB(TEST)'
                        DISP=RPL
                        SYSOPTS=\"\
                                \TYPE(MBR)\
                                \TEXT('COPIED FROM FILEA')\
                                \"\
                        )
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 32

Connect:Direct OS/400 RUN JOB Statement

The Connect:Direct OS/400 RUN JOB statement is used in a non-OS/400 Process to execute OS/400 CL commands on a remote Connect:Direct OS/400 node. The OS/400 CL command is submitted to the OS/400 to run as a separate job through the SBMJOB command. The Process does not wait until the OS/400 command has finished running before executing the next step in the Process.

The execution of the RUN JOB statement results in a return code but Connect:Direct does not verify the execution of the CL commands.

Format

The Connect:Direct OS/400 RUN JOB statement format includes the following parameters. Required parameters are shown in bold characters. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**RUN JOB**	**(DSN=AS400)**
		SYSOPTS="cmd(CL command)"
		SNODE

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string. Statement names and keywords are reserved and cannot be used as labels.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the RUN JOB statement.

RUN JOB
identifies the statement with all its parameters as the RUN JOB statement.

Required Parameters

DSN = AS400
is required when submitting a Process with RUN JOB from OS/390 that executes commands on OS/400. This parameter is used to satisfy syntax requirements on the OS/390 node. In this case, both SYSOPTS and DSN are required. Code the value for this parameter in uppercase characters.

SYSOPTS = "cmd(CL command) "
specifies system-specific submission information. For a remote OS/400 system, this parameter specifies a string 'cmd(any valid batch CL command)'. This CL command will be submitted on the remote OS/400 system through a SBMJOB command.

Optional Parameters

SNODE
specifies that the commands included in the SYSOPTS parameter are to be executed on the secondary node (SNODE), which is the Connect:Direct OS/400 node participating in the Process.

Example

In this example OS/390 Process, the job JANPMTS is submitted on the OS/400 node.

```
JOBSTEP     PROCESS     SNODE=CDAS400
                        SNODEID(USER01,PSWD01)
STEP01      RUN JOB     (DSN=AS400) SNODE
                        SYSOPTS=\"\
                        \CMD(\
                        \CALL PGM(INV/JANPMTS)\
                        \)\
                        \"\
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 33

Connect:Direct OS/400 RUN TASK Statement

The Connect:Direct OS/400 RUN TASK statement is used in a non-OS/400 Process to execute OS/400 CL commands on a remote Connect:Direct OS/400 node. Possible uses include:

- Calling programs before or after copying files (CALL command)
- Submitting jobs before or after copying files (SBMJOB command)
- Creating save files prior to transport (CRTSAVF and SAVxxx commands)
- Restoring save files after transport (RSTxxx command)
- Sending notification to a user on the Connect:Direct OS/400 node (SNDBRKMSG, SNDMSG, or an equivalent command)

Format

The Connect:Direct OS/400 RUN TASK statement consists of the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN TASK**	**(PGM=AS400)**
		SNODE
		SYSOPTS = “string”

Field Descriptions

RUN TASK
identifies the statement with all its parameters as the RUN TASK statement.

Required Parameters

PGM = AS400
is used to satisfy syntax checking. PGM=AS400 is required.

SYSOPTS = “string”
allows you to specify batch-capable OS/400 CL commands and parameters in the statement. The maximum number of permitted characters is 256.

A CMD(CL command) specifies that a CL command is to be executed by the Process. An arbitrary number of CMD( ) parameters can be coded. Any batch-capable CL command the user is authorized to issue may be specified.

Optional Parameters

SNODE
specifies that the commands included in the SYSOPTS parameter are to be executed on the secondary node (SNODE), which is the Connect:Direct OS/400 node participating in the Process.

Example

In this RUN TASK statement submitted from OS/390, the first command CRTSAVF is used to create a save file named SAVFILE. The file has the text description *CREATED BY PROCESS TEST*. The second command SAVLIB saves the library TESTDT1 in the save file created in the first command.

```
STEP1     RUN TASK     (PGM=AS400)
                       SNODE
                       SYSOPTS=\"\
                               \CMD(\
                                    \CRTSAVF\
                                    \FILE(TEST/SAVEFILE)\
                                    \TEXT('CREATED BY PROCESS TEST')
                                    \)\
                               \CMD(\
                                    \SAVLIB\
                                    \LIB(TESTDT1)\
                                    \DEV(*SAVF)\
                                    \SAVF(TEST/SAVEFILE)\
                                    \)\
                               \"\
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 34

Connect:Direct OpenVMS PROCESS Statement

A Connect:Direct OpenVMS PROCESS statement defines the attributes of a Connect:Direct Process and is always the first statement in a Process.

Format

The Connect:Direct OpenVMS PROCESS statement format includes the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the PROCESS statement format.

Label	Statement	Parameters
process name	**PROC**ess	**SNODE=secondary-node-name**
		SACCT='snode-accounting-data'
		PNODE=primary-node-name
		PNODEID=(id | [,pswd] [,newpswd])
		SNODEID=(id | [,pswd] [,newpswd])
		PACCT='pnode-accounting-data'
		CLASS=n
		DEFCONN.MODE=(first | scan | name)
		HOLD=Yes | No | Call
		PRTY=n
		RETAIN=Yes | No | Initial
		STARTT=([date |day][,hh:mm:ssXM])

Label	Statement	Parameters
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Field Descriptions

process name
specifies the name of the Process. The Process name may be 1-8 characters long. The first character must be alphabetic. The Process name must start in column one. This label is used to identify the Process in any messages or statistics relating to this Process.

PROCess
identifies the statement with all its parameters as the PROCESS statement. This statement identifier can be abbreviated to PROC.

Required Parameters

SNODE = secondary-node-name
is a 1-16 character alphanumeric name that specifies the secondary node (SNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

This is the logical node name that has been defined as the adjacent node in the network map.

Note: This parameter is not required if it is specified on the SUBMIT command.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. If CLASS is not specified in your Process, it will default to the class value specified for PARSESS in the network map.

DEFCONN.MODE = (first | scan | name)
specifies the default method of selecting a communications path that will be used to establish a session for Process execution.

first specifies that the Connect:Direct OpenVMS system will only use the first COMM.PATH specification in making a connection.

scan specifies that the Connect:Direct OpenVMS system will use each COMM.PATH specification in turn, until a connection is successfully made.

name represents an actual COMM.PATH name that the Connect:Direct OpenVMS system should use when making connections.

HOLD = Yes | No | Call
specifies whether the Process is to be placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process.

No specifies that the Process is to execute as soon as possible. HOLD=NO is the default.

Call specifies that the Process is to be placed in the Hold queue until a session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks.

PNODE = primary-node-name
is a 1-16 character alphanumeric name that specifies the primary node (PNODE) to be used in this Process. The name can be expressed in alphanumerics or nationals (@ # $), with embedded periods.

The node to which the Process is submitted is always the PNODE. This parameter defaults to the name of the node submitting the Process and need not be specified. It is used for documentation purposes only.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one you used to sign on to a Connect:Direct node.

id specifies the security ID passed to the security system at the PNODE (1-8 alphanumeric characters).

pswd specifies the current security password for the specified ID. This parameter can be used by the security system at the PNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

newpswd specifies the new security password. It can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

PRTY = n

specifies the Process priority in the Transmission Control Queue (TCQ). This priority is used for Process selection and does not affect OpenVMS priority.

The default priority is defined during installation of Connect:Direct OpenVMS. The range is from 0-15. See the *Connect:Direct OpenVMS Installation Guide* for more information.

RETAIN = Yes | No | Initial

keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies the Process remains on the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval. However, a date is invalid as a STARTT subparameter when used in conjunction with RETAIN.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time the Connect:Direct system is initialized. The Process will not execute when initially submitted. Note that STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks.

For Connect:Direct OS/400: This parameter is ignored when the SNODE is a Connect:Direct OS/400 node.

SNODEID = (id [,pswd] [,newpswd])

specifies security user IDs and passwords at the secondary node (SNODE).

id specifies the security ID passed to the security system on the SNODE (1-8 alphanumeric characters).

For Connect:Direct Tandem: This subparameter specifies the Tandem group number and user number. These numbers can range from 0-255.

For Connect:Direct OS/400: This subparameter specifies the AS/400 user profile used for authorization checks during Process execution and is limited to 8 characters even though AS/400 user profiles may be 10 characters long.

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password (1-8 alphanumeric characters). This is optional unless the user has security set to require a password.

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD™ must be running on Tandem.

STARTT = ([date | day][,hh:mm:ssXM])
specifies that the Process is not to be executed until a specified date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

---

date specifies that the Process is to be held until the desired date. The day (dd), month (mm), and year (yy) can be specified in one of the following formats:

yymmdd	mm/dd/yy
yy/mm/dd	mm.dd.yy
yy.mm.dd	yyddd (Julian date)
mmddyy	yy/ddd (Julian date)

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

---

day specifies the day of the week that the Process is to be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value may be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday, with Monday as the only STARTT parameter, the Process will not run until the following Monday.

You can also specify TODAY, which releases the Process for execution the day and time of Process submission (unless the time of day is specified), or TOMORROW, which releases the Process for execution the next day. If a time of day is not specified with TOMORROW, the Process will execute after midnight.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If the 12-hour clock is used, 01:00:00 hours can be expressed as 1:00AM, and 13:00 hours can be expressed as 1:00PM.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT to release the Process for execution at midnight.

Note: When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time is specified.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden in the SUBMIT command.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Example

The following is a example PROCESS statement; its description follows:

```
PROC1     PROCESS     SNODE=CD.NODE.A
                      SNODEID=(JONES,OPENUP)
                      CLASS=4
                      HOLD=YE
                      PACCT='OPERATIONS, DEPT. 87'
                      RETAIN=NO
```

The Process named PROC1 specifies a secondary node (SNODE) of CD.NODE.A. The corresponding security userids and passwords (SNODEID) have been included.

This Process will run in CLASS 4.

The Process will be placed on the Hold queue until it is released for execution with a CHANGE PROCESS command.

The PACCT parameter specifies that all accounting information will be attributed to the operations account, department 87, if the node has a program that maintains this information.

Once the Process executes, it will be deleted because the RETAIN parameter is set to NO. Note that NO is the default value for the RETAIN parameter.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 35

Connect:Direct OpenVMS COPY Statement

The Connect:Direct OpenVMS COPY statement allows you to copy files from one node to another.

Format

The COPY statement contains a FROM parameter that includes the source file name and a TO parameter that includes the destination file name. Additional parameters and subparameters can be specified to further customize the file transfer operation.

The following pages provide the format of the COPY statement. All valid parameters and subparameters are listed along with their possible values. All required entries are shown in bold. Following these pages is a definition of each parameter and subparameter.

To copy from one Connect:Direct system environment to another, refer to the appropriate COPY FROM and TO sections for those environments. For example, if the source file is located on a Connect:Direct OpenVMS node, refer to the COPY FROM information in this chapter. If the destination for the file is a Connect:Direct OS/390 node, refer to the TO information for Connect:Direct OS/390.

Label	**Statement**	**Parameters**	
stepname	**COPY**	**FROM**	**(**
			DSN=filename | FILE=filename
			PNODE | SNODE
			DISP=([OLD | SHR]

Label	Statement	Parameters	
			SYSOPTS="[MOUNT='string'] [DISMOUNT | NODISMOUNT] [TYPE='string'] [LIBRARY='string'] [REPLACE | NOREPLACE] [BINARY | NOBINARY] [STREAM | NOSTREAM] [PROTECTION='string'] [DIROWN | NODIROWN] [DISSETPROT | NODISSETPROT] [XLATE='string']"
			SELECT=(name | *)
			DATAEXIT=((exit-name-1 [, C 'p1', C 'p2',..., C 'pn']) . . . (exit-name-n [, C 'p1', C 'p2',..., C 'pn']))
			)
		TO	**(**
			DSN=filename | FILE=filename
			PNODE | SNODE
			TYPE=typekey
			DISP=[RPL | NEW | OLD | SHR | MOD]
			DCB=([DSORG=PS | PO | KSDS | RRDS] [,KEYLEN=number-of-bytes] [,LRECL=number-of-bytes] [,RECFM=V | F | VB | FB])

Label	Statement	Parameters	
			SYSOPTS="[MOUNT='string'] [DISMOUNT | NODISMOUNT] [TYPE='string'] [LIBRARY='string'] [REPLACE | NOREPLACE] [BINARY | NOBINARY] [PROTECTION='string'] [DIROWN | NODIROWN] [DISSETPROT | NODISSETPROT] [XLATE='string']"
			DATAEXIT=((exit-name-1 [, C 'p1', C 'p2',..., C 'pn']) . . . (exit-name-n [, C 'p1', C 'p2',..., C 'pn']))
			)
			COMPRESS [PRIMEchar= X'xx' | X'20' |C'c']
			CKPT=[nK|nM]

Additional Considerations

If you are copying a library from a Connect:Direct OpenVMS node to a PDS at a Connect:Direct OS/390 node, the following factors must be considered:

- Connect:Direct OpenVMS software transmits modules in alphabetical order.
- OpenVMS allows module names to be 39 characters. Because an OS/390 member name can only be eight characters long, the name is truncated to eight characters. It is recommended that OpenVMS module names be unique to eight characters.

These factors affect a library with modules having names with the same first eight characters. For example, if the OpenVMS module names for two files are **modulenamea** and **modulenameb**, both module names are truncated to **modulena** and transferred in alphabetical order.

If REPLACE is in effect, the OpenVMS module **modulenamea** is sent to the Connect:Direct OS/390 node and named **modulena**. Next, the OpenVMS module **modulenameb** is sent and replaces the module previously sent to the Connect:Direct OS/390 node.

Note: **Modulena** did not previously exist on the Connect:Direct OS/390 node.

If NOREPLACE is in effect, the OpenVMS module **modulenamea** is sent to the Connect:Direct OS/390 node and created as a new member. Because NO REPLACE is in effect, **modulenameb** does not replace the module previously sent to the Connect:Direct OS/390 node. Because both modules are identical to eight characters, **modulenameb** is not reflected in the library as a new member. Connect:Direct statistics show the results from the COPY.

Field Descriptions

COPY
identifies the statement with all its parameters as the COPY statement. This statement identifier is specified with either the FROM or TO parameter, whichever is coded first.

Required Parameters

FROM
specifies that the subsequent parameters and subparameters define the source file characteristics.

(FROM) DSN | FILE
specifies the originating file specification. Either the DSN or FILE parameter is valid.

If the source file is a OpenVMS file that has a maximum record size of 0 and a longest record size of 0, then DCB LRECL and BLKSIZE information will need to be specified for the destination file. The DCL command ANALYZE/RMS can be used to determine if the maximum record size and longest record size of a OpenVMS file are both 0. This is common with STREAM_LF files.

If you are copying a file from an IBM node, the file specification must be enclosed in single quotation marks if it uses spaces or other special characters not recognized by the IBM node.

If the OpenVMS file specification is not complete, the login default device and directory information is used to complete the file specification.

If the SUBMITTER qualifier is specified in the Connect:Direct OpenVMS SUBMIT command string and PNODEID and SNODEID are omitted, defaults for the userid provided in the subparameters of the SUBMITTER qualifier are used to complete the file specification. If either PNODEID or SNODEID is provided, defaults for the userid provided in the subparameter of either the PNODEID or SNODEID are used to complete the file specification.

Use of PNODEID and SNODEID is dependent on the direction of the transfer.

TO
specifies that the subsequent parameters and subparameters define the destination file characteristics.

(TO) DSN | FILE
specifies the destination file specification. Either the DSN or FILE parameter is valid.

If you are copying a file to an IBM node, the file specification must be enclosed in single quotation marks if it uses spaces or other special characters not recognized by the IBM node.

If the OpenVMS file specification is not complete, the login default device and directory information is used to complete the file specification.

If the SUBMITTER qualifier is specified in the Connect:Direct OpenVMS SUBMIT command string and PNODEID and SNODEID are omitted, defaults for the userid provided in the subparameters of the SUBMITTER qualifier are used to complete the file specification. If either PNODEID or SNODEID is provided, defaults for the userid provided in the subparameter of either the PNODEID or SNODEID are used to complete the file specification.

Use of PNODEID and SNODEID is dependent on the direction of the transfer.

Optional Parameters

CKPT = [nK | nM]
specifies the byte interval for checkpoint support. Checkpointing enables restart of interrupted transmissions at the last transmission checkpoint, thus avoiding the need to restart a transmission from the beginning; therefore, transfer time after a restart is reduced.

The Connect:Direct system converts the specified value to a block boundary, and a data transmission checkpoint is taken at that position.

Checkpointing is the responsibility of the node receiving the files. The Connect:Direct OpenVMS system can only checkpoint a Connect:Direct OS/390 file if Connect:Direct OS/390 initialization parameters are set as CKPT.MODE=(Record Record...). For details, refer to the *Connect:Direct OS/390 Installation and Administration Guide*.

Specifying a checkpoint value of zero in the Process disables automatic checkpointing. To override automatic checkpointing, use K to denote thousands; M for millions. *Only sequential disk files can be checkpointed.*

COMPRESS [PRIMEchar = X‘xx’ | X’20’ | C‘c’]
specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another and decreases transfer time. The file is automatically decompressed at the destination.

The default subparameter for the COMPRESS parameter is PRIMEchar=X‘20’, a blank, where char is optional. However, for variable byte files, it is suggested that a character other than blank (X‘20’) be used as a compression character.

If compression is specified, the Connect:Direct system reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character are compressed to 1 byte.

- Repetitive occurrences (ranging from 3-63) of any other character are compressed to 2 bytes.

Note: Compression is CPU-intensive, and its effectiveness is data dependent. It should only be used if its benefits are known.

PRIMEchar specifies the primary compression character. The default value for PRIMEchar is a blank (X‘20’).

DATAEXIT =((exit-name-1 [, C ‘p1’, C ‘p2’, ... , C ‘pn’]) . . . (exit-name-n [, C ‘p1’, C ‘p2’, ... , C ‘pn’]))
indicates that a user-written program is to be called to examine or modify the data for the associated COPY Process. Multiple exit names can be specified.

An example of the format for DATAEXIT is as follows:

```
DATAEXIT=((USER.EXIT1,C’PARM1’,C’PARM2’))
```

exit-name specifies the name of the user-written program to be given control to examine or modify the data.

parameter specifies a parameter to be passed to the specified exit. Each parameter must be enclosed in single quotation marks and prefaced by C.

Note: Currently, the DATAEXIT parameter is supported on VAX platform only.

DCB =([DSORG=PS | PO | KSDS |RRDS]
[,KEYLEN = number-of-bytes]
[,LRECL = number-of-bytes]
[,RECFM = V | F | VB | FB)]
specifies attributes to be used in allocating destination files. These parameters override the DCB information provided in the source file at the time it is opened. For SAM-to-SAM copies where the destination file is new and the DCB parameter is not coded with the TO parameter, DSORG is taken from the source file or from the TYPE defaults file if the TYPE keyword has been specified. Subparameters are as follows:

DSORG specifies the file organization. File organizations supported are: Physical Sequential (PS), library files (PO), keyed files (KSDS), and relative files (RRDS); the default is PS.

For copies from Connect:Direct OS/390: Connect:Direct OpenVMS does not support copying a VSAM KSDS file from OS/390 to a keyed file under OpenVMS.

KEYLEN specifies, in bytes, the length of the key used in the file.

LRECL specifies the record length.

RECFM specifies the record format. Valid record formats are V (variable-length files), F (fixed files), VB (variable block files), and FB (fixed block files). VB is the default.

Note: STREAM overrides RECFM if it is specified.

(FROM) DISP = [OLD | SHR]

specifies the status of data on the originating node. SHR is the default. Subparameters are as follows:

OLD requests exclusive access to a file.

SHR requests shared access to a file.

(TO) DISP = [RPL |NEW |OLD | SHR | MOD]

specifies the status of the data on the receiving node. NEW is the default. Subparameters are as follows:

RPL replaces the contents of the file if the version is specified and that version exists. Otherwise, the Connect:Direct system creates the file or a new version of the file if the file already exists and a version number is not specified.

NEW creates the file if it does not exist. Otherwise, it creates a new version of the file if it already exists.

OLD overwrites an existing version with exclusive access to the file. The file must exist.

SHR overwrites an existing version with shared access to the file. The file must exist.

MOD specifies that the Process step modifies the file by adding data to the end of the file. The file must exist.

Note: If DISP=MOD is specified on the TO clause of the Connect:Direct OpenVMS COPY statement and the destination file is a OpenVMS sequential file, checkpoint-restart is supported by Connect:Direct OpenVMS. Connect:Direct OS/390 does not support checkpoint-restart when DISP=MOD is specified on the TO clause of the Connect:Direct OS/390 COPY statement.

PNODE

specifies that the file to be copied resides on the primary node. PNODE is the default in the FROM clause of the COPY statement. If the file specification is incomplete and the PNODEID is specified, the default userid coded as a subparameter to the PNODEID is used to complete the file specification. Otherwise the defaults of the submitter userid are used.

SELECT = (name | *)

specifies selection criteria by which PDS members are to be copied. The SELECT parameter can be specified only with the FROM parameter.

name specifies an individual member name.

***** represents a global generic. A global generic indicates that all members of the file are to be included.

SNODE

specifies that the file to be copied resides on the secondary node. SNODE is the default in the TO clause of the COPY statement. If the file specification is incomplete and the SNODEID is specified, the default userid coded as a subparameter to the SNODEID is used to complete the file specification. Otherwise the defaults of the submitter userid are used.

SYSOPT="[MOUNT='string']
[DISMOUNT | NODISMOUNT]
[TYPE='string']
[LIBRARY='string']
[REPLACE | NOREPLACE]
[BINARY | NOBINARY]
[PROTECTION='string']
[DIROWN | NODIROWN]
[DISSETPROT | NODISSETPROT]
[XLATE='string']"

allows Connect:Direct OpenVMS system operations to be performed during file transfer.

When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in single quotation marks. The subparameters must be separated by blanks. For example:

```
SYSOPTS="MOUNT='MUA0 TAPELABEL' NODISMOUNT"
```

MOUNT='string' is any valid Digital Command Language (DCL) MOUNT command qualifiers to be used for tape operations. The DCL command MOUNT should not be in the string. There is no default for this subparameter, but it must be defined when you mount tape devices.

Coding ASSIST on the MOUNT command is discouraged unless an attendant is available to perform operator duties.

Allocating the tape drive is not required. The tape must have been previously initialized using the standard OpenVMS INITIALIZE command. If the tape has not been initialized, it does not have a volume label. For normal file access, OpenVMS cannot mount a tape unless it has a volume label; however, if the volume label is unknown, the tape may be mounted with the /OVERRIDE=ID qualifier positioned to modify either the command or the tape label parameter. For example:

```
MOUNT='MUA0 TAPELABEL /OVERRIDE=ID TAPELOGICAL'
```

DISMOUNT | NODISMOUNT specifies whether the tape volume is automatically dismounted once a COPY step is complete. The user should specify DISMOUNT in the SYSOPTS parameter of the last COPY step that references the tape volume. The default is DISMOUNT.

A tape may remain mounted across multiple copies within the same Connect:Direct Process. For example, specify the following in the SYSOPTS parameter of the first COPY step:

```
SYSOPTS="MOUNT='MUA0 TAPELABEL' NODISMOUNT"
```

TYPE='string' contains file attribute information. This SYSOPTS subparameter overrides the TYPE parameter on the COPY statement. String specifies the entry name

in the type library (NDM_TYPE.TLB). There is no default for this subparameter, and it is not required. For example:

```
TYPE='IMAGE'
```

The following are included in the type library:

- FIXED
- FORTRAN
- RELATIVE
- IMAGE
- SAVESET
- STREAM
- STREAM_CR
- STREAM_LF

These types provide FDL (file definition language) statements that are required to create the various files.

LIBRARY='string' specifies the library type of a Connect:Direct OpenVMS file (whether it is being sent or received). The string can be one of the following: TEXT, HELP, MACRO, <type-number>. The <type-number> is a binary key. TEXT is the default for this subparameter.

REPLACE | NOREPLACE specifies whether to replace library modules. NOREPLACE is the default.

Note: The version number must be included in the OpenVMS library name to ensure that the correct library is used.

BINARY | NOBINARY specifies whether data conversion occurs during a file transfer. BINARY (no text conversion) specifies that Connect:Direct OpenVMS files are not converted from ASCII to EBCDIC. NOBINARY is the default.

If the size of the binary file is not an even multiple of the record length of the IBM file, IBM pads the file (that is, enters binary zeros (0) in the last record of the file). Padding may cause the file to be unusable. If padding and data conversion are not desired when moving text files, send the Connect:Direct OpenVMS text file to an IBM file of variable-length records and include the BINARY subparameter in the Process.

PROTECTION='string' specifies the protection level desired on files copied to OpenVMS. This subparameter is coded in the format of the DCL SET PROTECTION command. The default is the protection previously defined on the OpenVMS process running Connect:Direct OpenVMS. The following is an example of using the PROTECTION subparameter:

```
PROTECTION='S:RWED,O:RWED,G:RE,W:E'
```

DIROWN | NODIROWN specifies file ownership. By default, the file is owned by the owner of the directory where the file was placed. If NODIROWN is specified, then the file will be owned by the submitter of the Connect:Direct Process. DIROWN is the default. The COPY SYSOPTS="DIROWN" overrides the NDM$$DIROWN logical in the server logical table.

If you are using a PROXY account and you are copying a file from an IBM node to a file on a remote DECNET node, SYSOPTS="NODIROWN" may need to be specified if the account which was used in the proxy does not exist in the remote DEC node.

DISSETPROT | DISSETPROT specifies protection/ownership.

If the Connect:Direct OpenVMS NDM$$DISSETPROT initialization parameter definition is 1, Connect:Direct OpenVMS will not attempt to set the protection/ownership on the file after it has been copied.

If the NDM$$DISSETPROT definition is not 1, Connect:Direct OpenVMS will attempt to set the protection/ownership as it normally does unless overridden by specifying DISSETPROT in the SYSOPTS parameter of the COPY statement.

If the NDM$$DISSETPROT definition is 1 but NODISSETPROT is specified in the SYSOPTS parameter of the COPY statement, Connect:Direct OpenVMS acts as if NDM$$DISSETPROT were not defined to 1 and attempts to set file ownership/protection.

XLATE='string' specifies ASCII-to-EBCDIC and EBCDIC-to-ASCII translation tables for a COPY operation. These tables are maintained in a OpenVMS text library. The value is a table name in the translate table text library. If no table name is specified, the Connect:Direct system uses the table named NDM_DEFAULT.

The following is an example of the format for XLATE:

```
XLATE='XLT_TABLE1'
```

TYPE = typekey

specifies the module name of the type file containing the default file attributes used to open the destination file. This typekey is specified only when defaults are requested by the user.

The SYSOPTS TYPE subparameter overrides this TYPE parameter on the COPY statement.

For copies from Connect:Direct OS/390, VM, or VSE to OpenVMS: For OS/390, VM, or VSE to OpenVMS copies where the typekey exceeds eight characters, the typekey must be entered into the SYSOPTS parameter on the TO clause of the Connect:Direct OpenVMS COPY statement.

For copies from Connect:Direct OpenVMS to OS/390: The typekey must not be greater than eight characters.

Example

These example COPY statements (STEP01 and STEP02) copy an executable file from a Connect:Direct OpenVMS node to a Connect:Direct OS/390 node and then back to the Connect:Direct OpenVMS node. Because BINARY is specified as part of the SYSOPTS parameter, ASCII-to-EBCDIC translation does not occur. The SYSOPTS string must be enclosed in double quotation marks.

```
STEP01     COPY FROM    (PNODE DSN='$DISK1:[USER.FILE]TEST.EXE'
                        SYSOPTS="BINARY"
                        TYPE=IMAGE
                        DISP=SHR)
                TO      (SNODE DSN=OS390.FILE
                        DISP=RPL)
STEP02     COPY FROM    (SNODE DSN=OS390.FILE
                        DISP=SHR)
                TO      (PNODE DSN='$DISK1:[USER.FILE]TEST2.EXE'
                        TYPE=IMAGE
                        SYSOPTS="BINARY"
                        DISP=RPL)
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 36

Connect:Direct OpenVMS RUN JOB Statement

The Connect:Direct OpenVMS RUN JOB statement allows a job to be submitted at the OS/390, VM, VSE, or OpenVMS nodes. Jobs can only be submitted from a file existing on the node executing the RUN JOB statement.

Connect:Direct does not verify the job statements. To determine the completion status of a RUN JOB statement, check Connect:Direct statistics records.

Format

The RUN JOB statement is comprised of the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN JOB**	**(DSN='file-specification')**
		SYSOPTS="[KEEP | NOKEEP] [LOG['file-specification'] | NOLOG] [Pn='string'] [PRINT | NOPRINT] [QUEUE='queue-name[:]'] [WAIT]"
		PNODE | SNODE

Field Descriptions

RUN JOB
identifies the statement with all its parameters as the RUN JOB statement.

Required Parameters

DSN = '**file-specification'**
specifies the file specification of the DCL command procedure. The DSN must be enclosed in single quotation marks. If the OpenVMS device and/or directory is not specified, the login default device and/or directory is used for the OpenVMS user name, which was derived from the Connect:Direct SUBMITTER (nodeid, userid) or specified in the PNODEID or SNODEID as applicable.

Optional Parameters

PNODE
specifies that the job is submitted on the primary node (PNODE), which is the node with Process control. PNODE is the default value.

SNODE
specifies that the job is submitted on the secondary node (SNODE), which is the node that interacts with the PNODE.

SYSOPT ="[KEEP | NOKEEP]
[LOG['file-specification'] | NOLOG]
[Pn='string']
[PRINT | NOPRINT]
[QUEUE='queue-name[:]']
[WAIT]"
allows you to specify system operation parameters on a Connect:Direct OpenVMS RUN JOB statement at the OpenVMS node. When coding a Connect:Direct Process, the entire SYSOPTS string must be enclosed in double quotation marks.

KEEP | NOKEEP specifies the disposition of a log file. KEEP specifies that the log file is kept. NOKEEP specifies that the log file is deleted after it is printed. NOKEEP is the default unless NOPRINT is specified.

LOG['file-specification'] | NOLOG specifies whether output is saved in a log file. If LOG has a file specification coded as part of the SYSOPTS parameter of the RUN JOB statement, output is saved in a log file. If the file specification is not included, the name of the log file defaults to the name of the command procedure with an extension of LOG. NOLOG specifies that a log file is not created. The default for this parameter is to keep the log and name it after the first (or only) file in the job.

Pn='string' allows parameters to be given to the command procedure. The range for n is 1-8, inclusive.

PRINT | NOPRINT specifies whether the log file for the job is queued for printing upon completion of the job. NOPRINT specifies that the batch job is not printed. The default is to submit the log file for printing to the system as a batch job. The default print queue for the log file is SYS$PRINT. If NOPRINT is specified, KEEP and LOG are assumed and the log file is not deleted.

QUEUE='queue-name[:]' specifies the name of the batch job queue where the job is entered. SYS$BATCH is the default when QUEUE is not specified.

WAIT specifies that the Connect:Direct OpenVMS RUN JOB statement waits for the command procedure to finish before processing continues. The default is that the statement does not wait for the OpenVMS command procedure to complete.

Example

In this example, the RUN JOB statement specifies that RJOB.COM executes on the PNODE. During Process execution, two parameters (P1 and P2) are passed to the command procedure. The LOG subparameter specifies that the file RJOB.LOG is created. It is not printed or deleted because NOPRINT is specified; KEEP is assumed.

```
RUNJ4 PROCESS PNODE=CD.VAX
STEP1 RUN JOB (DSN=$DISK1:[CD_20.PROCESS]RJOB.COM PNODE)
             SYSOPTS="P1='CD_20' P2='1-JAN-1995' NOPRINT LOG='$DISK1:[CD_20.LOG]RJOB.LOG'"
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 37

Connect:Direct OpenVMS RUN TASK Statement

Execution of the Connect:Direct OpenVMS RUN TASK statement creates a detached OpenVMS process that executes one or more DCL commands or command procedures. The SYSOPTS parameter of the Connect:Direct OpenVMS RUN TASK statement identifies operations to be performed by the operating system.

The Connect:Direct OpenVMS RUN TASK statement is permitted only within a Connect:Direct Process and is structured so that the Process waits until the OpenVMS process has finished running before the next step in the Process executes. The Process with a Connect:Direct OpenVMS RUN TASK statement can be submitted from either node.

Measuring Successful Execution

On OpenVMS, each command procedure exits with a severity level. The $SEVERITY value is translated according to the following table. The resulting return code is set for the last SYSOPTS command parameter specified in the Process (the last command executed under RUN TASK).

$Severity Value	Connect:Direct Return Code
0 (warning)	4
1 (success)	0
2 (error)	8
3 (info)	0
4 (severe)	16

The return code can be tested within the Process using the IF statement. See *Connect:Direct OpenVMS Conditional Statements* on page 323 for the statement format.

Avoiding I/O Errors

The RUN TASK starts a OpenVMS Process using a mailbox for terminal I/O. During the OpenVMS login process, the command procedures, SYLOGIN.COM and LOGIN.COM (if standard names are used), cannot include commands that may attempt to perform a terminal function.

For example, a command like **SET TERMINAL /INQUIRE** requires a response from the terminal; however, the Connect:Direct OpenVMS system does not return a response. If attempted, the RUN TASK results in an I/O error.

To avoid I/O errors when a RUN TASK is specified in a Process, enter the following commands in any SYLOGIN and LOGIN that may be used with a Connect:Direct RUN TASK. Note that the required portion of the commands is in bold print; other commands are for illustration only.

```
$ IF ( F$GETDVI ( “SYS$COMMAND”, “MBX” ))
         THEN  GOTO MBA
$ IF ( F$MODE() .NES. "INTERACTIVE")
         THEN GOTO NOT_INTERACTIVE
$   SET TERMINAL /INQUIRE
$   SET TERMINAL /WIDTH=80
$   SET TERMINAL /BROADCAST
$NOT_INTERACTIVE:
$MBA:
```

Format

The RUN TASK statement is comprised of the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
stepname	**RUN TASK**	**(PGM=VMS)**
		PNODE | SNODE
		SYSOPTS=“[OUTPUT=‘file specification’] [CMD=‘DCL command’]

Field Descriptions

RUN TASK
identifies the statement with all its parameters as the RUN TASK statement.

Required Parameters

PGM = VMS
indicates that the RUN TASK executes on a Connect:Direct OpenVMS node.

Optional Parameters

PNODE
specifies that the task will be executed on the PNODE, which is the default.

SNODE
specifies that the task will be executed on the secondary node (SNODE), which is the destination node.

SYSOPT="[OUTPUT='file specification']
[CMD='DCL command']"
allows you to specify DCL commands and parameters on the Connect:Direct OpenVMS RUN TASK statement. When coding a Process, the entire SYSOPTS string must be enclosed in double quotation marks. Each subparameter string must be enclosed in single quotation marks. The subparameters must be separated by blanks.

OUTPUT='file specification' specifies the SYS$OUTPUT of the OpenVMS process. If the file is not included as part of this parameter, the name of the log file defaults to the name of the command procedure with an extension of LOG. A log of the detached process is not created if the OUTPUT subparameter of the SYSOPTS parameter is not specified.

If only OUTPUT='device' is specified, the output from the OpenVMS process is written to the root directory of the account that issued the command. The output is assigned a type of .LOG. There is no filename.

If only OUTPUT='file' (without a directory) is specified, the file is written to the root directory on the device assigned to the account that created the Process.

CMD='DCL command' specifies a DCL command to be executed by the OpenVMS process. An arbitrary number of CMD parameters can be coded. Any DCL commands that the user is authorized to issue may be specified, with the exception of commands that require the use of single or double quotation marks.

If a DCL command procedure requires a string parameter containing embedded blanks and normally requiring single or double quotation marks, use underscores to replace the embedded blanks, therefore eliminating single or double quotation marks. For

example, the DCL command **MAIL/SUBJECT “two words” filename** can be coded in the RUN TASK statement as follows:

```
SYSOPTS=“CMD=‘MAIL/SUBJECT=two_words filename’”
```

Example

In this example, the RUN TASK statement executes the PURGE command at the OpenVMS node, which in this Process is also the PNODE. PURGE deletes all files with a type of .OUT in directory ACCT.TEST on device U1. The RUN TASK statement also executes the command procedure CREATE_DATA.COM in directory SC1 on device U1. Output from the OpenVMS process created by the Connect:Direct OpenVMS RUN TASK statement is routed to device NL.

```
RUNT4     PROCESS     PNODE=CD.VAX
STEP1     RUN TASK    (PGM=VMS)  PNODE SYSOPTS="
                      OUTPUT=’NL:’
                      CMD=’PURGE U1:[ACCT.TEST]*.OUT  ’
                      CMD=’@U1:[SC1]CREATE_DATA.COM’"
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 38

Connect:Direct OpenVMS SUBMIT Statement

During execution of a Connect:Direct Process, the SUBMIT statement causes another Process to be submitted to the PNODE (primary node), the node with Process control, or the SNODE, the node that interacts with the PNODE during Process execution. The Process to be submitted must reside in a file on the node where the SUBMIT statement executes. This node is referred to as the SUBNODE.

Format

The Connect:Direct OpenVMS SUBMIT statement format includes the following parameters. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the SUBMIT statement format.

Label	Statement	Parameters
stepname	**SUBMIT**	**DSN=filename | FILE=filename**
		CLASS=n
		NEWNAME=new-name
		SUBNODE=PNODE | SNODE
		SNODE=secondary-node
		SNODEID=(id [,pswd] [,newpswd])
		SACCT='snode-accounting-data'
		PNODEID=(id [,pswd] [,newpswd])
		PACCT='pnode-accounting-data'
		HOLD=Yes | No | Call
		PRTY=n
		RETAIN=Yes | No | Initial

Label	Statement	Parameters
		STARTT=([date | day][,hh:mm:ssxm])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Field Descriptions

SUBMIT
identifies the statement with all its parameters as the SUBMIT statement.

Required Parameters

DSN = filename | FILE = filename
specifies the name of the file that contains the Process. Either DSN or FILE must be specified.

Optional Parameters

CLASS = n
determines the node-to-node session on which a Process can execute. If CLASS is not specified in your Process, it defaults to the class value specified for PARSESS in the network map.

HOLD = Yes | No | Call
specifies whether the Process is placed in the Hold queue at submission.

Yes specifies that the Process is submitted to the Hold queue and remains there until the operator explicitly releases the Process. When both HOLD=YES and a START value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time is specified.

No specifies that the Process executes as soon as possible. HOLD=NO is the default.

Call specifies that the Process is placed in the Hold queue until a session is established with the specified SNODE. This session could be established by either another Process running on the PNODE or the SNODE contacting the PNODE. For example, a Process submitted HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

NEWNAME = new-name
specifies the new name to be given to the Process. The default value is the label on the PROCESS statement.

PACCT = 'pnode-accounting-data'
specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks.

PNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the primary node (PNODE). This parameter should only be used to validate security with an ID different from the one used to sign on to the Connect:Direct system.

id specifies the security ID passed to a security exit (1-8 alphanumeric characters).

pswd specifies the current security password. This parameter can be used by the security exit to validate the current security password (1-8 alphanumeric characters).

newpswd specifies the new security password. It can be used by the security exit to change the current security password to the new security password (1-8 alphanumeric characters).

PRTY = n
specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ is a file that holds all Processes that have been submitted to the Connect:Direct system. High numbers indicate high priorities; low numbers indicate low priorities.

This priority is used only for Process selection within class and does not affect transmission priority. The range is from 0-15. If PRTY is not specified, the default is the priority defined by NDM$$PRTYDEF in CONFIGSRV.COM. See the *Connect:Direct OpenVMS Installation and Administration Guide* for more information.

RETAIN = Yes | No | Initial
keeps a copy of the Process in the Hold queue after the Process executes.

Yes specifies that the Process remains on the Hold queue after initial execution. The Process must then be released manually through the CHANGE PROCESS command to cause it to be executed, or explicitly deleted through the DELETE PROCESS command.

If RETAIN=YES is specified, the Process is automatically held until released unless the STARTT parameter is coded. Use RETAIN in conjunction with STARTT to cause a Process to run repeatedly at a given interval.

When a Process is submitted with RETAIN=YES and HOLD=NO or CALL, the HOLD parameter is ignored.

No specifies that the system deletes the Process after execution. The default value for RETAIN is NO.

Initial specifies that the Process is to be executed every time the Connect:Direct system is initialized. Processes submitted with RETAIN=INITIAL do not execute when initially submitted. STARTT should not be coded with RETAIN=INITIAL.

SACCT = 'snode-accounting-data'
specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. If special characters are part of the accounting data, the string must be enclosed in single quotation marks.

SNODE = secondary-node
is a 1-16 alphanumeric character name that specifies the symbolic node name of the secondary node. The name can be expressed in alphanumerics or nationals (@ # $) with embedded periods. The first character must be alphabetic.

This parameter can override the value specified in the PROCESS statement. The default value for SNODE is the value coded in the PROCESS statement. Connect:Direct allows the PNODE and SNODE to specify the same symbolic node name.

SNODEID = (id [,pswd] [,newpswd])
specifies security user IDs and passwords at the secondary node (SNODE).

id specifies the security ID passed to the security system on the SNODE (1-8 alphanumeric characters).

pswd specifies the current security password and can be used by the security system on the SNODE to validate the current security password (1-8 alphanumeric characters).

newpswd specifies the new security password and can be used by the security system to change the current security password to the new security password (1-8 alphanumeric characters).

For Connect:Direct Tandem: SAFEGUARD must be running on Tandem.

STARTT = ([date | day] [,hh:mm:ssXM])
specifies that the Process is not to execute until a specified date or time. The date, day, and time are positional parameters. If the date or day is not specified, a comma must precede the time.

Note: STARTT should not be coded with RETAIN=INITIAL.

When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence. Therefore, a Process submitted with HOLD=YES is placed on the Hold queue even if a start time has been specified.

date specifies that the Process is to be held until the desired date. The day (dd), month (mm), and year (yy) can be specified in one of the following formats:

yymmdd	mm/dd/yy
yy/mm/dd	mm.dd.yy
yy.mm.dd	yyddd (Julian date)
mmddyy	yy/ddd (Julian date)

If only date is specified, the time defaults to 00:00.

Note: If RETAIN=YES, a date cannot be specified in the STARTT parameter.

---

day specifies the day of the week that the Process is to be released for execution. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. The day value may be abbreviated to the first two characters.

If the day of the week is specified with RETAIN=YES, the Process executes the same day every week. If only day is specified, the time defaults to 00:00. This means that if a Process is submitted on Monday with Monday as the only STARTT parameter, the Process does not run until the following Monday.

You can also specify TODAY, which releases the Process for execution today, or TOMORROW, which releases the Process for execution the next day.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM.

The time of day can be expressed using the 24-hour clock or the 12-hour clock. If the 24-hour clock is used, valid times are from 00:00:00 to 24:00:00. If AM and PM are not used, the 24-hour clock is assumed.

If hh:mm:ssXM is specified with RETAIN=YES, the Process executes at the same time every day. Minutes and seconds need not be specified.

You can also specify NOON, which releases the Process for execution at noon, or MIDNIGHT, which releases the Process for execution at midnight.

SUBNODE = PNODE | SNODE

specifies the node on which the Process defined in this SUBMIT statement executes. Specifying PNODE means that the Process is submitted on the node that has Process control. Specifying SNODE means that the Process is submitted on the node participating in, but not controlling, Process execution. In both cases, the Process must reside on the node on which it is being submitted. The default is PNODE.

&symbolic_name_1 = variable-string-1
&symbolic_name_2 = variable-string-2
.
.
.
&symbolic_name_n = variable-string-n

specifies the default value for a symbolic parameter in the Process. This default can be overridden when submitting the Process.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

The symbolic parameter for the SUBMIT statement must begin with a single ampersand. This allows submission of the Process that contains the SUBMIT statement to resolve symbolic parameters correctly.

An ampersand symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Example

In this Process, STEP1 copies ACCT.VMSPROC to ACCTPROC.Connect:Direct in directory WORKDIR on device $DISK1 at the SNODE. STEP2 submits ACCTPROC.CD, which executes on the SNODE. The SNODE parameter specified in the SUBMIT statement overrides the value specified in the PROCESS statement.

```
PROC1   PROCESS     PNODE=CD.CA
                    SNODE=CD.ATLANTA
STEP1   COPY  FROM  (DSN=ACCT.VMSPROC
                    DISP=SHR
                    PNODE)
              TO    (DSN=$DISK1:[WORKDIR]ACCTPROC.CD
                    DISP=RPL
                    SNODE)
STEP2   SUBMIT      DSN=$DISK1:[WORKDIR]ACCTPROC.CD
                    SUBNODE=SNODE
                    SNODE=CD.CA
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 39

Connect:Direct OpenVMS SYMBOL Statement

The Connect:Direct OpenVMS SYMBOL statement allows you to build symbolic substitution values.

Format

The Connect:Direct OpenVMS SYMBOL statement format includes the following parameters. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**SYMBOL**	**&symbolic_name=variable-string**

Field Descriptions

SYMBOL
identifies the statement with all its parameters as the SYMBOL statement.

Required Parameters

&symbolic_name=variable-string
specifies the string that is substituted into the Process.

When the Connect:Direct system encounters an ampersand (&) plus 1-17 alphanumeric characters, Connect:Direct substitutes a string represented by that ampersand and the alphanumeric characters.

Symbols in the string are resolved from previously specified values in a PROCESS, SUBMIT, or SYMBOL statement. With the SYMBOL statement, different pieces of a Connect:Direct statement string can be concatenated, allowing the user to move data in a variety of ways.

A null value can be specified if the equal sign (=) is immediately followed by a comma. A symbolic parameter containing special characters must be enclosed in single quotation marks.

Example

This example demonstrates using the SYMBOL statement in a Connect:Direct Process to build a OpenVMS filename. The substitution variable is the directory name, which is resolved at Process submission.

Assume that the file, SYM2, contains the following Process:

```
SYM2     PROCESS     SNODE=CD.VMS.NODE
                     &NAME=,
         SYMBOL      &DSN1=DISK1:[
         SYMBOL      &DSN2=]LOGIN.COM
         SYMBOL      &DSN3=&DSN1.&NAME.&DSN2
STEP01   COPY  FROM  (DSN=&DSN3 PNODE)
               TO    (DSN=DISK1:[USER1]SYM2.COM SNODE)
```

To submit the Process, issue the following SUBMIT command:

```
$ CDUI SUBMIT SYM2 /SYMBOLICS=(“NAME=‘USER1’”)
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 40

Connect:Direct OpenVMS Conditional Statements

Conditional statements permit you to alter the sequence of Connect:Direct Process execution based on the completion of the previous step in the Process.

Formats

Formats for Connect:Direct conditional statements follow. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	IF	(label condition nn) THEN
		(process steps)
	ELSE	
		(alternative process steps)
	EIF	
optional	GOTO	label
	EXIT	

Statement Descriptions

IF THEN

specifies that the Connect:Direct system executes a block of Connect:Direct statements based on the completion code of a Process step. An EIF statement must be used in conjunction with an IF THEN statement. See the following *Field Descriptions* section for details.

ELSE
designates a block of Connect:Direct statements that execute when the IF THEN condition is not satisfied. No parameters exist.

EIF
is required for specifying the end of the IF THEN or IF THEN ELSE block of statements. No parameters exist.

GOTO
moves to a specific step within a Process. See the following *Field Descriptions* section for details.

EXIT
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

label
For the **IF THEN** statement**,** the label specifies the name of a previous step whose completion code is used for comparison.

For the **GOTO** statement**,** the label specifies the name of a subsequent step in a Process (required for GOTO only). The name can neither be the label of a preceding step nor the label of the GOTO statement of which it is a part.

Note: User-defined labels must begin in column one. The label consists of a 1-8 character alphanumeric string, with the first letter alphabetic only.

condition
specifies the type of comparison to be performed. This condition checking can be based on comparisons for equality, inequality, greater than, less than, greater than or equal to, and less than or equal to.

Note: The completion code from RUN JOB is for the job submission only and not the completion code of the job submitted.

Valid symbols, alternate symbols, and conditions follow:

= or EQ specifies that the completion code must be equal to the value nn for the condition to be satisfied.

<> or NE specifies that the completion code must not equal the value nn for the condition to be satisfied.

>= or GE specifies that the completion code must be greater than or equal to the value nn for the condition to be satisfied.

> or GT specifies that the completion code must be greater than the value nn for the condition to be satisfied.

<= or LE specifies that the completion code must be less than or equal to the value nn for the condition to be satisfied.

< or LT specifies that the completion code must be less than the value nn for the condition to be satisfied.

nn

specifies the numeric value to be used for completion code checking. If coded as X‘nn’, it is a hexadecimal value; any other coding indicates it as decimal.

If a completion code less than 4 is returned, typically the Process completed successfully. In most cases, a return code greater than 4 indicates the Process ended in error. A return code equaling 4 indicates a warning.

THEN

specifies subsequent processing to be performed if the condition specified is true.

Example

In this example, the Process runs a job on OpenVMS. It uses conditional statements to check the completion code from STEP01. If the completion code equals 0, a message is issued to the operator indicating that the command procedure was successful. If the completion code is any value other than 0, a message is issued to the operator indicating that the command procedure failed.

```
PROC01      PROCESS     SNODE=CD.VMS.NODE
STEP01      RUN JOB     (DSN=DISK1:[USER1.PROC_CD1]DIR.COM
                        SYSOPTS=“NOPRINT LOG WAIT” SNODE
STEP02      IF          (STEP01 EQ 0) THEN
            RUN TASK    (PGM=VMS PNODE  SYSOPTS=“
                        CMD=‘REPLY/USER=USER1/BELL BATCH_JOB_SUCCEED’"
            ELSE
            RUN TASK    (PGM=VMS PNODE  SYSOPTS=“
                        CMD=‘REPLY/USER=USER1/BELL BATCH_JOB_FAILED’"
            EIF
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 41

Connect:Direct UNIX Process Statement

The Connect:Direct UNIX **process** statement defines the beginning of a series of one or more statements that specify functions to be performed. A Process is made up of a required **process** statement, followed by one or more other statements, such as **copy**, **run job**, **run task**, **submit**, and **pend**.

A Connect:Direct UNIX **process** statement defines the attributes of a Connect:Direct Process and is always the first statement in a Process.

Note: The maximum storage area allowed for a Connect:Direct UNIX Process statement is 60K. To accommodate a larger Process, split the Process into two separate Processes. Include a SUBMIT statement in the first Process to run the second Process.

Format

The Connect:Direct UNIX **process** statement format includes the following parameters. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the **process** statement format.

Label	Statement	Parameters
process name	**process**	class=*n*
		crc= on | <u>off</u>
		hold=yes | <u>no</u> | call
		notify=*username@hostname or user@localhost*
		pacct="*pnode accounting data*"
		plexclass=*remote_plexclass*
		pnodeid=(*id* [,*pswd*])
		prty=*nn*

Label	Statement	Parameters
		retain=yes | no | initial
		sacct="*snode accounting data*"
		snode=[*nodename*] | [(*hostname* | *nnn.nnn.nnn.nnn*);(*portnumber* | *portname*)] The **snode** parameter must be specified on the **process** statement or **submit** command.
		snodeid=(*id* [,*pswd*[,*newpswd*]])
		startt=([*date* | *day*][,*hh*:*mm*:ss[am | pm]])
		&symbolic name1=variable string 1 *&symbolic name2=variable string 2* . . . *&symbolic namen=variable string n*

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

You can use the label to identify the Process in any messages or statistics relating to this Process

Note: Although the Connect:Direct UNIX label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

The label field defines a Process name (**pname**) that may be used as the **pname** parameter value on the **change process, delete process, flush process, select process,** and **select statistics** commands as the name of the target Process.

Required Parameters

There are no required parameters for the **process** statement. However, **snode** must be specified on the **submit** command or the **process** statement.

Optional Parameters

class = *n*
determines the node-to-node session on which a Process can execute. A Process may execute on the class specified or any higher session class. The default class is specified as the sess.default parameter in the Initialization Parameters file. The **class** default is 1.

crc =on | off
determines if CRC checking is activated for a process. To define this parameter, the user must be given the authority to perform CRC checking in the user authority. You can globally turn on CRC checking using the initialization parameter. CRC checking can only be performed for TCP/IP processes and cannot be used by Processes using Secure+ Option or SNA. If CRC checking is enabled for a Process on a Secure+ enabled node or on a computer with SNA enabled, CRC checking is ignored.

When CRC checking is performed, the copy termination statistics record contains a message indicating that CRC was performed for a Process.

hold = yes | no | call
specifies how the Process is handled in the Hold queue.

yes specifies that the Process is held in the Hold queue in HI status until it is released by a **change process** command. When **hold=yes** and a **startt** value is specified, the **hold** parameter takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

Note: When a Process is submitted with **retain=yes** and **hold=no** or **hold=call,** the **hold** parameter is ignored.

---

no specifies that the Process is not held in the Hold queue. It is executed as soon as resources are available. The default is **no**.

Note: Setting **hold** to no value, such as hold or hold=, does not produce an error. It is another way of placing the Process in the Hold queue similar to setting **Hold=yes**.

---

call specifies that the Process is held (no prompt returns) until the remote node (**snode**) connects to the local node (**pnode**). At that time, the Process is released for execution. The Process is also released when another Process on the local node connects to the **snode**.

notify = *username@hostname or user@localhost*
specifies the user name to receive Process completion messages. This parameter uses the rmail utility available in the UNIX System V mail facility to deliver the completion messages.

pacct = "*pnode accounting data*"
specifies accounting data for the **pnode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

plexclass=*remote_plexclass*
Specifies the 1–8 character name of a valid plexclass on the remote server. This parameter gives you control over which Connect:Direct/Plex Server is selected by the Con-

nect:Direct/Plex Manager to execute a Process. For example, you can specify plexclass=tape in a Process to direct a Process to a Connect:Direct/Plex Server that has a tape drive.

pnodeid = (*id* [,*pswd*])
specifies security information for the Process on the **pnode**. Each **pnodeid** parameter value can be 1-64 alphanumeric characters long.

id specifies the userid to be used as a security ID on the **pnode**.

pswd specifies a user password on the **pnode**.

If ***pswd*** is specified, ***id*** also must be specified. These values must be coded in the order of ***id*** and ***pswd***.

prty = *nn*
specifies the priority of the Process on the Transmission Control Queue (TCQ). The **prty** parameter is used only for Process selection. A Process with a higher priority is selected for execution before a Process with a lower priority. The value specified for the **prty** parameter does not affect the priority during transmission. Values range from 0-15. The highest priority is 15. If **prty** is not specified, the default is 10.

retain = yes | no | initial
specifies whether the Process is retained on the TCQ in the Hold queue for re-execution after execution has completed.

yes specifies that the Process is retained on the Hold queue in HR status after execution. The Process must then be released manually through the **change process** command to cause it to be executed, or explicitly deleted through the **delete process** command. When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

no specifies that the Process is not retained. The default is **no**.

initial specifies that the Process is retained on the Hold queue in HR status and automatically executed each time the Process Manager initializes. The **startt** parameter should not be coded when **retain=initial** is coded. This causes the SUBMIT command to fail.

sacct = "*snode accounting data*"
specifies accounting data for the **snode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

The **sacct** parameter overrides any accounting data specified on the **process** statement of the submitted Process.

snode = [*nodename*] | [(*hostname | nnn.nnn.nnn.nnn*);(*port number | portname*)]
is a 1-256 alphanumeric character string that specifies the node name of the secondary node. The **snode** default value is the value coded in the **process** statement. It is required either on the **submit** command or **process** statement.

nodename is the node name of the adjacent Connect:Direct node. The secondary node name corresponds to an entry in the network map file.

hostname is the name of the host machine where the remote Connect:Direct is running. This is applicable only for TCP/IP.

nnn.nnn.nnn.nnn is the IP address of the remote Connect:Direct node. Each **nnn** is a decimal number from 0-255, inclusive. This is applicable only for TCP/IP.

portnumber* | *portname identifies the communications port for the Connect:Direct software. The ***portnumber*** is a decimal number from 1024-5535. This is applicable only for TCP/IP.

snodeid = (*id* [,*pswd*[,n*ewpswd*]])
specifies security user IDs and security passwords for the Process on the SNODE. Each **snodeid** parameter value can be 1-64 alphanumeric characters.

id specifies the userid to be used as a security ID on the **snode**.

pswd specifies the user password on the **snode**.

newpswd specifies the new password value. The user password is changed to the new value on the **snode** if the userid and old password are correct. The ***newpswd*** parameter is not valid if the **snode** is a Connect:Direct UNIX node.

If ***pswd*** is specified, ***id*** also must be specified. If ***newpswd*** is specified, ***pswd*** also must be specified. These values must be coded in the order of ***id***, ***pswd***, and ***newpswd***.

startt = ([*date* | *day*] [,*hh*:*mm*:ss [am | pm]])
specifies that the Process is executed at a specified date and/or time. The Process is placed in the Timer queue in WS status. The **date**, **day**, and **time** values are positional subparameters. If **date** or **day** is not specified, a comma must precede **time**.

Note: The **startt** parameter should not be coded when **retain=initial** is coded. This causes the **submit** command to fail.

---

date specifies that the Process will execute on a specific date. You can specify the ***date*** subparameter in the format ***mm*/*dd*/*yyyy*** or ***mm*-*dd*-*yyyy***, where:

- ***mm*** indicates month
- ***dd*** indicates day
- ***yyyy*** indicates year

If only ***date*** is specified, ***time*** defaults to 00:00. The current date is the default.

day specifies the day of the week a Process is released for execution. Values are today, tomorrow, monday, tuesday, wednesday, thursday, friday, saturday, and sunday.

If only day is specified, time defaults to 00:00:00. For example, if a Process is submitted on Monday, with **monday** as the only **startt** parameter, the Process does not run until the following Monday when the time reaches 00:00:00.

hh*:*mm*:*ss* [am | pm]** specifies the hour (hh***), minute (***mm***), and second (***ss***) the Process should start. You can specify hour in either 12- or 24-hour format. If you use 12-hour

format, you must specify **am** or **pm**. A space is required before **am** or **pm**. The default is 24-hour format. The default value is 00:00:00.

Note: When both **hold=yes** and a **startt** value are specified, the hold specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed on the Hold queue even if a start time is specified

&symbolic name1 = variable string 1
&symbolic name2 = variable string 2
.
.
.
&symbolic namen = variable string n

specifies the value for a symbolic parameter in the Process. This default can be overridden when submitting the Process. The value is substituted within the Process when the symbolic parameter is encountered. The value for the symbolic parameter must be in double quotation marks if it is a keyword or contains special characters. A symbolic name cannot exceed 32 characters.

The symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Example

This example shows a Process that uses symbolics to allow you to specify the file and data set names at the time of submission. Process accounting data is specified for the **pnode** and **snode**.

```
copyseq     process     snode=dallas
                        pacct="dept-59"
                        sacct="dept-62"
step01      copy  from  (file=&file)
                  to    (file=&dsn
                        snode)
            pend
```

The following **submit** command specifies the file and data set names to be used in a file transfer.

```
submit     file=copyseq
           &file=myfile
           &dsn=abc;
```

The following Process is generated by the previous input:

```
copyseq     process     snode=dallas
                       pacct="dept-59"
                        sacct="dept-62"
step01      copy  from  (file=myfile)
                  to    (file=abcsnode)
            pend
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 42

Connect:Direct UNIX Copy Statement

The **copy** statement is comprised of a **from** clause that includes the source file name and a **to** clause that includes the destination file name. Additional parameters can be specified to further customize the file transfer operation.

To copy from one Connect:Direct system environment to another, refer to the appropriate **copy** from and to sections in this manual for those environments.

The **copy** statement execution results in a return code. See page 362 for a list of standard return codes.

Format

The **copy** statement format includes the following parameters. Default values for parameters and subparameters are underlined. Required parameters are shown in bold characters.

Label	**Statement**	**Parameters**	
stepname	**copy**	**from**	(
			file=*filename* | dsn=*filename*
			pnode | snode
			sysopts=":datatype=text | binary | vb:" ":xlate=no | yes:" ":xlate.tbl=<*pathname/filename*>:" ":strip.blanks=yes | no: " ":permiss=*nnn:*" ":pipe=yes | no:" ":codepage=(*source codepage, destination codepage*):"
			)
			ckpt=no | *n*k |*n*m
			compress [[primechar = x'*xx*' | x'20' | c'*c*'] | extended]

Label	Statement	Parameters	
			(
		to	file=*filename* | dsn=*filename*
			pnode | snode
			sysopts=":datatype=text | binary | vb:" ":xlate=no | yes:" ":xlate.tbl=<*pathname/filename*>:" ":strip.blanks=yes | no:" ":permiss=*nnn*:" ":pipe=yes | no:" ":codepage=(*source codepage, destination codepage*):"
			disp=[(] new | mod | rpl [)]
			)

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

A label is not required on the **copy** statement.

Note: Although the Connect:Direct UNIX label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

copy

identifies the statement with all its parameters as the **copy** statement.

Required Parameters

from

specifies that the subsequent parameters and subparameters define the source file characteristics.

to

specifies that the subsequent parameters and subparameters define the destination file characteristics.

Optional Parameters

ckpt = no | *n* | *n*K | *n*M | *n*G

specifies the byte interval for checkpoint support, which allows restart of interrupted transmissions at the last valid checkpoint point and therefore reduces the time to retransmit the file. If the **ckpt** parameter and value are not specified, the default is the value specified by the **ckpt.interval** parameter in the Initialization Parameters file. Code this parameter between the from and to parameters.

The following table shows the maximum number of digits for each option.

Option	Byte Interval
no	no checkpointing
nnnnnnnnn	number of bytes
*nnnnnnn*K	number of kilobytes
*nnnn*M	specifies a number of megabytes
*n*G	specifies a number of gigabytes

compress [[primechar = x'*xx*' | x'20' | c'*c*'] |
extended=[(CMPrlevel = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
WINdowsize = 9 | 10 | 11| 12 | 13 | 14 | 15
MEMlevel = 1 | 2 3 | 4 | 5 | 6 | 7 | 8 | 9)]]

specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file will be automatically decompressed at the destination. The default subparameter for the **compress** parameter is **primechar=x'20'**. Use **compress** for text data or single-character repetitive data. Use **compress extended** for all other types of data. Code this parameter between the from and to parameters.

primechar specifies the primary compression character. The default value for primechar is **x'20**'.

The Connect:Direct software reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character will be compressed to one byte.
- Repetitive occurrences (ranging from 3-63) of any other character will be compressed to two bytes.

extended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited and data is repetitive. Use CMPrlevel, WINdowsize, and MEMlevel to refine the extended compression.

CMPrlevel is the compression level used. Level 1 is the fastest but offers the least degree of compression. Level 9 provides the greatest degree of compression but

the slowest rate of compression. The default is 1 unless overridden in the initparms.

WINdowsize specifies the size of the compression window or history buffer. The greater the window size is, the greater the degree of compression but at the cost of a greater amount of virtual memory that is used. The default is 13. The following table indicates the approximate amount of memory used for each window size.

Window Size	Memory Used
9	2K
10	4K
11	8K
12	16K
13	32K
14	64K
15	128K

MEMlevel specifies how much virtual memory should be allocated to maintain the internal compression state. Memory level 1 uses the least amount of memory, but slows processing and reduces the degree of compression. Memory level 9 provides the fastest speed but uses the most memory. The default is 4. The following table shows approximate memory usage for each memory level setting.

Memory Level	Memory Usage
1	1K
2	2K
3	4K
4	8K
5	16K
6	32K
7	64K
8	128K
9	256K

disp = [(] new | mod | rpl [)]

defines the state that the target file should be in when it is opened. The **disp** parameter also determines how the file should be opened, either **new**, **mod**, or **rpl**.

new indicates the file must not already exist.

mod indicates that if the file already exists, data will be appended to the end of the file. If the file does not exist, then **mod** behaves as if **new** is specified.

rpl indicates that **copy to** will act as if **disp=new** was specified if the file does not already exist. If the file exists, then it will be overwritten. The default is **rpl**.

file = *filename* | dsn = *filename* | file = "*unix command* [;*unix command* [;*unix command*...]]"

The **file** or **dsn** parameter specifies the optional path name and required file name. The file name can be enclosed in double quotation marks.

If **sysopts="pipe=yes"**, the **file** parameter specifies a UNIX command to execute. The command can be a command, program, or shell script including any options and arguments.

When specifying **file=*filename***, enclose the file name in double quotation marks when using a reserved word (statement name or keyword) for the file name.

Connect:Direct UNIX supports the string (*) and character (?) wildcards, allowing you to copy multiple files from a source directory to a target directory with a single copy command.

You can use a Connect:Direct UNIX node to perform a wildcard copy *send* to any other Connect:Direct platform. With a Connect:Direct UNIX node, you can *receive* a wildcard copy *only* from a Connect:Direct Windows node or from another Connect:Direct UNIX node.

pnode

When **pnode** is coded with the **from** parameter, the file to be copied resides on the primary node. When **pnode** is coded with the **to** parameter, the file is sent to the primary node. The **pnode** parameter is the default with the **from** parameter.

snode

When **snode** is coded with the **from** parameter, the file to be copied resides on the secondary node. When **snode** is coded with the **to** parameter, the file is sent to the secondary node. The default for the **to** parameter is **snode**.

sysopts=":datatype=text | binary:"
":xlate=no | yes:"
":xlate.tbl=<*pathname/filename*>:"
":strip.blanks=yes | no:"
":permiss=*nnn*:"
":pipe=yes | no:"
":codepage=(*source codepage*, *destination codepage*):"

specifies system-specific parameters on the **copy** statement. The subparameters are specified in the same format as fields within a Connect:Direct UNIX configuration file. They are a series of field names and values (***fldn=valn***), each of which is delimited by a colon. Enclose the string of subparameters in double quotes; for example:

```
":fld1=val1:fld2=val2:...:fldn=valn:"
```

datatype specifies the type of data contained within the file: **text**, **binary**, or **vb**.

text indicates it is a text file. The default is **text**. The following shows the default source file attributes assigned for UNIX text files, which are to be used if necessary by the remote Connect:Direct node:

- dsorg=ps
- recfm=vb
- lrecl=23036
- blksize=23040

binary indicates the file contains binary data. The following shows the default source file attributes assigned for UNIX binary files, which are to be used if necessary by the remote Connect:Direct node:

- dsorg=ps
- recfm=u
- lrecl=0
- blksize=23040

vb indicates the file is in variable block fomat. Varaible block format is structured as follows:

BDW | RAW | Data . . . | BDW | RAW | Data . . .

- BDW=block descriptor word containing 2 bytes for the length of the block (including 4 bytes of the BDW) and 2 bytes of zeros
- RDW=descriptor word containing 2 bytes for the length of the block (including 4 bytes of the RDW) and 2 bytes of zeros
- Data=record of user data equal in length to the RDW minus the 4 bytes of the RDW (for example, an RDW of 76 bytes is followed by a record of 72 bytes)

Note: The **vb** feature does not support ASCII/EBCDIC translation. If you specify **":datatype=vb:"**, specify **":xlate=no:"**. The copy will fail if you specify **":xlate=yes:"**.

---

xlate = no | yes indicates whether character translation should be performed using the default or user-supplied translation table. Typically, this translation is between ASCII and EBCDIC.

no specifies the translation is not performed. **no** is the default for binary files.

yes specifies the translation is performed. **yes** is the default for text files.

xlate.tbl =<*pathname*/*filename*> specifies that a translation table is to be used that is different from the default table used by the Connect:Direct software.

strip.blanks determines whether trailing blank characters are removed from a line of text before it is written to the UNIX text file.

yes specifies to remove trailing blank characters.

no specifies trailing blank characters are not removed. The default is **no**.

permiss=*nnn* specifies the UNIX file permissions for a file being created by the copy operation. The **permiss** subparameter is ignored if it is specified for a file that already exists.

nnn is a 3-digit octal number that defines privileges for users. Each type of user (owner, group, and others, respectively) can be assigned read, write, and execute privileges.

The following table shows valid octal numbers and associated permissions for each **n** based on the binary numbering system:

Octal	Binary	Permissions
0	000	No permissions
1	001	Execute permission
2	010	Write permission
3	011	Write and execute permissions
4	100	Read permissions
5	101	Read and execute permissions
6	110	Read and write permissions
7	111	Read, write, and execute permissions

For example, permiss=634 indicates that the owner has read and write permissions, the group has write and execute permissions, and others have read permissions.

pipe = yes | no specifies whether the pipe I/O function is activated. The pipe I/O function allows commands, programs, or shell scripts including any options and arguments to be copied to the destination file or from the source file

yes specifies the pipe I/O function is activated.

no specifies the pipe I/O function is not activated..

Note: Checkpoint/restart is not supported for the pipe I/O function.

codepage=(*source codepage, destination codepage*)
determines which codepage is used to translate a file.

Note: If you do not identify two parameters (source codepage, destination codepage) in the **from** statement or in the **to** statement, the codepage listed as the source is converted to UTF-8, sent, and then converted to the codepage identified in the **to** statement at the destination location. When this occurs, the file can be translated incorrectly.

Three methods can be used to translate files:

- **sysopts=":codepage=(*source codepage, destination codepage*):"**

  This definition can be used either in the **from** or **to** statement and identifies the codepages used for translating a file either before it is transferred (**from**) or after it is received at the destination location (**to**).

- **from sysopts=":codepage=source codepage:" to sysopts=":codepage=destination codepage:"**

  This definition translates the file using the source codepage defined in the **from** statement to UTF-8 format, transfers the file, and then translates it at the destination from UTF-8 to the codepage defined in the **to** statement. See the note above.

- **from sysopts=":codepage=(*source codepage, transitional codepage*):" to sysopts=":codepage=(*transitional codepage, destination codepage*):"**

  This definition translates the file using the source codepage defined in the **from** statement to the transitional codepage, transfers the file, and then translates it at the destination location from the transitional codepage to the destination codepage defined in the **to** statement. If UTF-8 is used as the transitional codepage, then this translation method performs the same as the second translation method described above.

Codepage translation supports the translation of text or binary files. When translating text files, codepage translates one line at a time. The trailing line feed character is removed from the text line. If the **strip.blanks** parameter is set to **yes**, trailing blanks are removed from the file and a line feed is appended to the line of text.

Examples

In this example, the Process copies a binary file from a local UNIX node to an OS/390 node:

```
copyseq process snode=dallas
/* When copying, make sure the datatype is set to binary. */
step01  copy    from   (file=a.out
                       pnode
                       sysopts=":datatype=binary:")
                ckpt=64k
                compress extended= (CMP=1
                                    WIN=9
                                   MEM=1)
                to    (file=TESTAOUT
                        snode
                        disp=(rpl))
/*  If step01 succeeds, CD will copy the same file back to UNIX.  */
step02  if     (step01 > 4 ) then
         exit
         eif
/*  Before copying the file back, delete it first.  */
step03    run task
                     pnode
                     sysopts="rm -f a.out"
step04  if     (step03 > 4 ) then
         exit
         eif
/*  Copy the file from OS390 to UNIX.  */
/*  When copying, make sure the datatype is set to binary.  */
step05   copy  from (file=TESTAOUT
                     snode)
                  ckpt=64k
                to  (file=a.out
                     pnode
                     disp=(rpl)
                     sysopts=":datatype=binary:")
         pend
```

The following Process copies a file from a local Connect:Direct node to a node named Atlanta:

```
copyseq  process      snode=atlanta
step01   copy  from   (file=myfile)
                      sysopts=":datatype=binary:")
               to     (file=yourfile
          pend
```

Sample Wildcard Copy Statements

In the following example, a Connect:Direct UNIX PNODE directory */financial/accounts* contains the files *customer1*, *customer2*, *customer3*, *supplier1*, and *supplier2*. A Connect:Direct Windows SNODE has the directory *J:/financial/accounts*. The following wildcard copy command copies the files called *customer1*, *customer2*, and *customer3* from the PNODE to the *J:/financial/accounts* directory on the SNODE. The source file names and the destination file names are identical.

```
WILDCOPY COPY
      FROM (FILE=/financial/accounts/customer?)
      TO   (FILE=J:\financial\accounts\
             DISP=RPL)
```

Note: You must include the ending backslash (\) for the destination directory.

The following wildcard copy step copies *customer1*, *customer2*, *customer3*, *supplier1*, and *supplier2* into the *J:\financial\accounts* directory on the SNODE. The source file names and the destination file names are identical.

```
WILDCOPY COPY
      FROM (FILE=/financial/accounts/*)
      TO   (FILE=J:\financial\accounts\
            DISP=RPL)
```

Note: You must include the ending backslash (\) for the destination directory.

Wildcard Copy Send to a Connect:Direct UNIX Node

To copy send to a Connect:Direct UNIX node, you must include an ending forward slash (/) in the **TO FILE=** parameter. Following is an example:

```
WILDCOPY COPY
      FROM (FILE="/financial/accounts/customer?")
      TO   (FILE=/financial/accounts/
            DISP=RPL)
```

Wildcard Copy Send to a Connect:Direct OS/390 Node

To copy send to sequential files on a Connect:Direct OS/390 node, you must include an ending period (.) in the **TO FILE**= parameter. Following is an example:

```
WILDCOPY COPY
      FROM (FILE=/financial/accounts/*)
      TO   (FILE=FINANCIAL.ACCOUNTS.
            DISP=RPL)
```

To copy send to a PDS on a Connect:Direct OS/390 node, you must use an asterisk (*) for the PDS member name. Following is an example:

```
WILDCOPY COPY
      FROM (FILE=/financial/records/*)
      TO   (FILE=FINANCIAL.RECORDS(*)
            DISP=RPL)
```

Wildcard Copy Send to a Connect:Direct Node with Download Restrictions

To copy send to a Connect:Direct node that enforces download restrictions, use an asterisk (*) for the **TO FILE**= parameter if the destination directory is the download directory. Following is an example:

```
WILDCOPY COPY
      FROM (FILE=/financial/records/*)
      TO   (FILE=*
            DISP=RPL)
```

Chapter 43

Connect:Direct UNIX Run Job Statement

The Connect:Direct UNIX **run job** statement allows you to run UNIX commands in a Process. A UNIX command shell is invoked to execute one or more UNIX commands. The Process does not wait until the UNIX commands have finished running before executing the next step in the Process.

The execution of the **run job** statement produces a return code. See the **nn** parameter on page 362 for a list of standard return codes.

Note: The return code *does not* indicate whether the UNIX command was successfully executed. The return code only indicates whether Connect:Direct successfully submitted the **run job** statement.

Format

The Connect:Direct UNIX **run job** statement format includes the following parameters. Required parameters are shown in bold characters. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**run job**	**dsn=dsn[(member)]**
		pnode | snode
		sysopts="*unix command* [;*unix command* [;*unix command*...]]"

Field Descriptions

Label
Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

Note: Although the Connect:Direct UNIX label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the **run job** statement.

run job
identifies the statement with all its parameters as the **run job** statement.

Required Parameters

dsn = dsn[(member)]
specifies the name of the data set containing the job to be submitted. If the file is a PDS, the member containing the job must be specified. The data set containing the job must already exist on the node where the job will be submitted.

sysopts = "*unix command* [;*unix command* [;*unix command*...]]"
specifies a UNIX command to execute under a B shell. The command can be a command, program, or shell script including any options and arguments. Specify the command as if it were being issued at a UNIX terminal.

The **run job** statement does not wait until the UNIX command is completed to complete execution. This is the distinguishing feature between **run job** and **run task**.

Once started, the UNIX command or commands in a **run job** statement will be permitted to continue execution. Connect:Direct UNIX does not have control over the UNIX commands.

Optional Parameters

pnode | snode
specifies on which Connect:Direct node the **run job** statement is executed. The default is **pnode**.

Example

In this example, the application myjob is submitted to the **pnode** system. See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

```
jobstep  run job    sysopts="myjob param1"
```

Chapter 44

Connect:Direct UNIX Run Task Statement

The Connect:Direct UNIX **run task** statement allows you to run UNIX commands in a Process. A UNIX command shell is invoked to execute one or more UNIX commands. The Process waits until the UNIX commands have finished running before executing the next step in the Process.

Caution: Do not specify programs in the **run task** statement that cannot complete without user intervention because they will not be completed.

Process statements that follow the **run task** statement do not execute until the submitted job completes. The execution of the **run task** statement results in a return code which is the exit code for the program executed using **run task**.

Format

The **run task** statement format includes the following parameters. Required parameters are shown in bold characters. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**run task**	**(pgm=program-name)**
		pnode | snode
		sysopts="*unix command* [;*unix command* [;*unix command*...]]"

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the **run task** statement.

Note: Although the Connect:Direct UNIX label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

run task
identifies the statement with all its parameters as the **run task** statement.

Required Parameters

pgm = program-name
specifies the name of the program to be attached as the subtask. The program runs on the node specified and has access to the DD cards allocated on that node only.

sysopts = "*unix command* [;*unix command* [;*unix command*...]]"
specifies one or more UNIX commands to execute. The command can be a UNIX command, program, or shell script including any options and arguments. Specify the command as if it were being issued at a UNIX terminal.

The **run task** statement waits until the UNIX command has completed execution.

The UNIX command or commands executed in a **run task** step can be terminated by a **flush process force** or a **stop force** or **stop immediate** command.

Optional Parameters

pnode | snode
specifies on which Connect:Direct node the **run task** will be executed. The default is **pnode**.

Example

In this example, the user-supplied program is specified in a **run task** statement. The program requires two parameters. The program is located on the primary node. The **run task** statement is contained within a Process that is also located on the **pnode**.

```
taskstep run task sysopts="grep -e CTRC s19931215.001 > stat.txt"
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 45

Connect:Direct UNIX Submit Statement

During execution of a Connect:Direct Process, the **submit** statement causes another Connect:Direct Process to be submitted to either the **pnode**, which is the node with Process control or the **snode**, which is the secondary node that participates in the Process.

The Process submitted must reside in a file on the node where the **submit** statement executes; this node is referred to as the **subnode**.

Note: The **submit** statement described in this chapter is not the same as the **submit** command. The **submit** statement is used within a Process to submit another Process. See the *Connect:Direct UNIX User's Guide* for **submit** command syntax and parameters.

Any parameter values specified in the **submit** statement override the parameter values contained in a **process** statement named by the **dsn** | **file** parameter.

The execution of the **submit** statement produces a return code. Refer to page 362 for a list of standard return codes.

Format

The **submit** statement format includes the following parameters. Required parameters are shown in bold characters. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**submit**	**file=*filename***
		class=*n*
		hold=yes | no | call
		newname=*new process name*
		notify=*username@hostname*
		pacct="*pnode accounting data*"
		plexclass=*remote_plexclass*

Label	Statement	Parameters
		pnodeid=(*id* [,*pswd*])
		prty=*nn*
		retain=yes | no | initial
		sacct="*snode accounting data*"
		snode=*nodename* | *hostname* | *nnn.nnn.nnn.nnn* [;*portname* | ;*nnnnn*]
		snodeid=(*id* [,*pswd*[,*newpswd*]])
		startt=([*date* | *day*][,*hh*:*mm*:*ss*[am | pm]])
		subnode=pnode |snode
		&symbolic name1=variable string 1 *&symbolic name2=variable string 2* *.* *.* *.* *&symbolic namen=variable string n*

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the **submit** statement.

Note: Although the Connect:Direct UNIX label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

submit

identifies the statement with all its parameters as the **submit** statement.

Required Parameters

file = *filename*
specifies the name of the file that contains the Process. The file name may include a pathname indicating the location of the Process. There is no practical limit on the character length of the **file** parameter.

Note: A pathname recognized in *cli* is relative to the current working directory. An absolute path (starting from root "/") will always work.

Enclose the file name in double quotation marks when using a reserved word (statement name or keyword) for the file name.

Optional Parameters

class = *n*
determines the node-to-node session on which a Process can execute. A Process may execute on the class specified or any higher session class. The default class is specified as the sess.default parameter in the Initialization Parameters file. The **class** default is 1.

hold = yes | no | call
specifies how the Process is handled in the Hold queue.

yes specifies that the Process is held in the Hold queue in HI status until it is released by a **change process** command. When **hold=yes** and a **startt** value is specified, the **hold** parameter takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

Note: When a Process is submitted with **retain=yes** and **hold=no** or **hold=call,** the **hold** parameter is ignored.

---

no specifies that the Process is not held in the Hold queue. It is executed as soon as resources are available. The default is **no**.

Note: Setting **hold** to no value, such as hold or hold=, does not produce an error. It is another way of placing the Process in the Hold queue similar to setting **Hold=yes**.

---

call specifies that the Process is held (no prompt returns) until the remote node (**snode**) connects to the local node (**pnode**). At that time, the Process is released for execution. The Process is also released when another Process on the local node connects to the **snode**.

newname = *new process name*
specifies the new name to be given to the Process. This value overrides the label on the **process** statement. There is no practical limit on the character length of the **newname** parameter.

notify = *username@hostname*
specifies a user name and host computer name to use in alerting a user whenever a step completes within a Process. Notification is accomplished through the UNIX System V mail facility.

pacct = "*pnode accounting data*"
permits the user to specify an arbitrary string to be used as accounting information for the **pnode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

plexclass=*remote_plexclass*
Specifies the 1–8 character name of a valid plexclass on the remote server. This parameter gives you control over which Connect:Direct/Plex Server is selected by the Connect:Direct/Plex Manager to execute a Process. For example, you can specify plexclass=tape in a Process to direct a Process to a Connect:Direct/Plex Server that has a tape drive.

pnodeid = (*id* [,*pswd*])
specifies security information for the Process on the PNODE. Each **pnodeid** parameter value can be 1-64 alphanumeric characters long:

id specifies the userid to be used on the **pnode**.

pswd specifies a user password on the **pnode**.

If ***pswd*** is specified, ***id*** also must be specified. These values must be coded in the order of ***id*** and ***pswd***.

prty = *nn*
specifies the priority of the Process on the Transmission Control Queue (TCQ). The **prty** parameter is used only for Process selection. A Process with a higher priority will be selected for execution before a Process with a lower priority. The value specified for the **prty** parameter does not affect the priority during transmission. Values range from 0-15. The highest priority is 15. If **prty** is not specified, the default is 10.

retain = yes | no | initial
specifies whether the Process is retained on the TCQ in the Wait queue for re-execution after execution has completed.

yes specifies that the Process is retained on the Hold queue in HR status after execution.

When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

no specifies that the Process is not retained. The default is **no**.

initial specifies that the Process is retained on the Hold queue in HR status and automatically executed each time the Process Manager initializes. The **startt** parameter should not be coded when **retain=initial** is coded. This causes the **submit** statement to fail.

sacct = "*snode accounting data*"
specifies accounting data for the **snode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

The **sacct** parameter overrides any accounting data specified on the **process** statement of the submitted Process.

snode = *nodename* | *hostname* | *nnn.nnn.nnn.nnn* [;*portname* | ;*nnnnn*]
is a 1-256 alphanumeric character string that specifies the node name of the secondary node. This parameter overrides the value specified in the **process** statement. The **snode** default value is the value coded in the **process** statement. It is required either on the **submit** command or **process** statement.

nodename is the name of the remote Connect:Direct node. The secondary node name corresponds to an entry in the Network Map file.

hostname is the name of the host machine where the remote Connect:Direct software is running. This is applicable only for TCP/IP.

nnn.nnn.nnn.nnn is the IP address of the remote Connect:Direct node. Each **nnn** is a decimal number from 0-255, inclusive. This is applicable only for TCP/IP.

[;*portname*|;*nnnnn*] identifies the communications port for the Connect:Direct software. The ***nnnnn*** value is a decimal number from 1024 to 65535.

snodeid = (*id* [,*pswd*[,*newpswd*]])
specifies security userids and security passwords for the Process on the **snode**. Each **snodeid** parameter can contain 1-64 alphanumeric characters.

id specifies the security ID that is passed to a security exit.

pswd specifies the user password on the **snode**.

newpswd is the new password value. The user password is changed to the new value on the **snode** if the userid and old password are correct. The ***newpswd*** parameter is not valid if the **snode** is a Connect:Direct UNIX node.

If ***pswd*** is specified, ***id*** must also be specified. If ***newpswd*** is specified, ***pswd*** must also be specified. These values must be coded in the order of ***id***, ***pswd***, and ***newpswd***.

startt = ([*date* | *day*] [,*hh*:*mm*:ss [am | pm]])
specifies the date and/or time to execute the Process. The Process is placed in the Timer queue in WS status. The ***date***, ***day***, and ***time*** are positional parameters. If ***date*** or ***day*** is not specified, a comma must precede ***time***.

Note: The **startt** parameter must not be coded with **retain=initial** or **retain=yes**.

date specifies to hold Process until the desired date. The ***date*** value can be specified in the format ***mm/dd/yyyy*** or ***mm-dd-yyyy***, where:

- ***mm*** indicates month.
- ***dd*** indicates day.
- ***yyyy*** indicates year.

If only ***date*** is specified, ***time*** defaults to 00:00:00. The current date is the default.

day specifies the day of the week a Process is to be released for execution. Values are today, tomorrow, monday, tuesday, wednesday, thursday, friday, saturday, and sunday.

Note: A process submitted with **startt=(today's day of the week)** executes immediately. For example, if today is Tuesday, setting **startt=tuesday** causes the Process to execute immediately if no value is set for the time.

hh*:*mm*:*ss* [am | pm]** specifies the hour (hh***), minute (***mm***), and second (***ss***) that the Process should start. Hour can be specified in either 12- or 24-hour format. If the 12-hour format is used, then **am** or **pm** must be specified. A space is required before **am** or **pm**. The default is 24-hour format. The default value is 00:00:00, which indicates midnight.

Note: When both **hold=yes** and a **startt** value are specified, the hold specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

subnode = pnode | snode

specifies the node on which the Process will be submitted. The Process must reside on the node on which it is being submitted.

pnode specifies that the Process is submitted on the node that has Process control. The default value is **pnode**.

snode specifies that the Process is submitted on the node participating in, but not controlling, Process execution.

&symbolic name1 = variable string 1
&symbolic name2 = variable string 2
.
.
.
&symbolic namen = variable string n

specifies the value for a symbolic parameter in the Process. This default can be overridden when submitting the Process. The value is substituted within the Process when the symbolic parameter is encountered. The value for the symbolic parameter must be in double quotation marks if it is a keyword or contains special characters. A symbolic name cannot exceed 32 characters.

The symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Example

This example shows how you can copy a file from your local node to a node in Dallas, which will then copy it to a node in Tampa.

A user issues a **submit** command to initiate a Process.

```
submit file=process1
```

The **submit** statement contained within process1 specifies the Process name of process2. As a result, process2 will also be submitted.

```
process1    process     snode=dallas
copystep    copy  from  (file=myfile)
                  to    (dsn=yourfile
                        disp=(new,keep))
substep     submit      file=process2
                        subnode=snode
                        &dsn=yourfile
             pend
```

The **copy** statement in process2 will copy yourfile to the file newfile in Tampa.

```
process2     process     snode=tampa
copystep     copy  from  (dsn=&dsn)
                   to    (dsn=newfile
                         disp=(new,keep))
             pend
```

The symbolic parameter &dsn is converted in process2 to the value of yourfile, which was specified in the **submit** statement in process1.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 46

Connect:Direct UNIX Conditional Statements

The conditional statements (**if then**, **else**, **eif**, **goto**, and **exit**) permit you to alter the sequence of Process execution based on the completion of the previous step in the Process.

Formats

Formats for Connect:Direct UNIX conditional statements follow. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	**if**	**(*label condition nn*) then**
		(*process steps*)
	else	
		(*alternative process steps*)
	eif	
optional	goto	label
	exit	

Statement Descriptions

if then

specifies that the Connect:Direct system executes a block of Connect:Direct statements based on the completion code of a Process step. An **eif** statement must be used in conjunction with an IF THEN statement. See the following *Field Descriptions* section for details.

else
designates a block of Connect:Direct statements that execute when the **if then** condition is not satisfied. No parameters exist.

eif
is required for specifying the end of the **if then** or **if then else** block of statements. No parameters exist.

goto
moves to a specific step within a Process. See the following *Field Descriptions* section for details.

exit
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

Required Parameters

condition
The ***condition*** parameter specifies the type of comparison that will be performed. This condition checking can be based on such comparisons as *equal to*, *greater than*, or *less than*. The valid conditions are as follows:

== | **=** | **eq** specifies that the return code must be equal to the value **nn** for the condition to be satisfied.

<> | **!=** | **ne** specifies that the return code must not equal the value **nn** for the condition to be satisfied.

>= | **=>** | **ge** specifies that the return code must be greater than or equal to the value **nn** for the condition to be satisfied.

> | **gt** specifies that the return code must be greater than the value **nn** for the condition to be satisfied.

<= | **=<** | **le** specifies that the return code must be less than or equal to the value **nn** for the condition to be satisfied.

< | **lt** specifies that the return code must be less than the value **nn** for the condition to be satisfied.

nn
This parameter specifies the numeric value to be used for return code checking. If coded as **x'*nn*'**, it is a hexadecimal value. Any other coding indicates it is decimal. Standard return codes are as follows:

- 0 indicates successful completion.
- 4 indicates warning.
- 8 indicates error.
- 16 indicates catastrophic error.

then
specifies subsequent processing based on the other specified parameters.

Optional Parameters

label
Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

A label is not required on the **if** statement.

Note: Although the Connect:Direct UNIX label may be up to 256 characters, labels for many of the Connect:Direct platforms cannot exceed eight characters.

Example

In this example, the Connect:Direct UNIX Process contains all of the conditional statements. A description of the Process follows.

```
copy1     process     snode=chicago
step01    copy  from  (file=abc pnode)
                to    (file=def snode)
step02    if          (step01 gt 4) then
                     goto step07
          else
step03    runjob      sysopts="myjob"
          eif
step04    if          (step03 >= 8) then
          exit
step05    if          (step03 lt 4) then
step06    copy  from  (file=xyz pnode)
                to    (file=uvw snode)
          eif
          exit
step07    run task    sysopts=“verify”
          exit
          pend
```

copy01 is the **process** statement defining the secondary node as chicago.

step01 copies file abc on the **pnode** to file def on the **snode**.

step02 checks the completion code of step01. If step01 fails (return code greater than 4), step07 executes. If step01 completes with a return code of 4 or less, step03 executes.

step03 submits a job, myjob, on the **pnode**.

step04 checks the completion code of step03. If step03 fails with a code of 8 or greater, the Process terminates. Otherwise, step05 executes.

step05 checks the completion code from step03. If less than 4, indicating the step completed without error, step06 executes.

step06 copies file xyz on the **pnode** to file uvw on the **snode**. The Process will then exit.

step07 only executes if step01 fails. The program verify runs, sending an Operation Failed Message to the console operator.

pend marks the end of a Process.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 47

Connect:Direct UNIX Pend Statement

The **pend** statement is optional and marks the end of an Connect:Direct UNIX Process. There are no parameters associated with the **pend** statement.

An example of **pend** statement usage follows.

Example

The following Process copies a file from a local node to a node named Atlanta. The **pend** statement indicates the end of the Process.

```
copyseq    process     snode=atlanta
step01     copy  from  (file=myfile)
                 to    (dsn=yourfile)
            pend
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 48

Connect:Direct OpenVME Process Statement

The Connect:Direct OpenVME **process** statement defines the beginning of a series of one or more statements that specify functions to be performed. A Process is made up of a required **process** statement, followed by one or more other statements, such as **copy**, **run job**, **run task**, **submit**, and **pend**.

A Connect:Direct OpenVME **process** statement defines the attributes of a Connect:Direct Process and is always the first statement in a Process.

Note: The maximum storage area allowed for a Connect:Direct OpenVME Process statement is 60K. To accommodate a larger Process, split the Process into two separate Processes. Include a SUBMIT statement in the first Process to run the second Process.

Format

The Connect:Direct OpenVME **process** statement format includes the following parameters. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and keywords are in bold print. Default values for parameters and subparameters are underlined. A description of each parameter and subparameter follows the **process** statement format.

Label	Statement	Parameters
process name	**process**	class=*n*
		hold=yes | no | call
		notify=*username@hostname*
		pacct="*pnode accounting data*"
		pnodeid=(*id* [,*pswd*])
		prty=*nn*
		retain=yes | no | initial
		sacct="*snode accounting data*"

Label	Statement	Parameters
		snode=[*nodename*] | [(*hostname* | *nnn.nnn.nnn.nnn*);(*portnumber* | *portname*)] The **snode** parameter must be specified on the **process** statement or **submit** command
		snodeid=(*id* [,*pswd*[,*newpswd*]])
		startt=([*date* | *day*][,*hh*:*mm*:ss[am | pm]])
		&symbolic name1=variable string 1 *&symbolic name2=variable string 2* . . . *&symbolic namen=variable string n*

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

The **process** statement is the only statement that requires a label. The label is used to identify the Process in any messages or statistics relating to this Process.

Note: Although the Connect:Direct OpenVME label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

The label field defines a Process name (**pname**) that may be used as the **pname** parameter value on the **change process, delete process, flush process, select process,** and **select statistics** commands as the name of the target Process.

Required Parameters

There are no required parameters for the **process** statement. However, **snode** must be specified on the **submit** command or the **process** statement.

Optional Parameters

class = *n*
determines the node-to-node session on which a Process can execute. A Process may execute on the class specified or any higher session class. The default class is specified as the sess.default parameter in the Initialization Parameters file. The **class** default is 1.

hold = yes | no | call
specifies how the Process is handled in the Hold queue.

yes indicates that the Process is held in the Hold queue until it is explicitly released by a **change process** command. When both **hold=yes** and **startt** values are specified, the **hold** specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed on the Hold queue even if a start time is specified.

no indicates that the Process is not held on the Hold queue. It is executed as soon as resources are available. The default is **no**.

call indicates that the Process is held (no prompt returns) until the remote node (**snode**) connects to the local node (**pnode**). At that time, the Process is released for execution. The Process is also released when another Process on the local node connects to the **snode**.

Note: When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

---

notify = *username@hostname*
specifies a user name and host computer name to use in notifying a user when a step within the Process completes. Notification is accomplished through the UNIX mail facility.

pacct = "*pnode accounting data*"
specifies accounting data for the **pnode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

pnodeid = (*id* [,*pswd*])
specifies security information for the Process on the **pnode**. Each **pnodeid** parameter value can be 1-64 alphanumeric characters long.

id specifies the userid to be used as a security ID on the **pnode**.

pswd specifies a user password on the **pnode**.

If ***pswd*** is specified, ***id*** also must be specified. These values must be coded in the order of ***id*** and ***pswd***.

prty = *nn*
specifies the priority of the Process on the Transmission Control Queue (TCQ). The **prty** parameter is used only for Process selection. A Process with a higher priority is selected for execution before a Process with a lower priority. The value specified for the **prty** parameter does not affect the priority during transmission. Values range from 0-15. The highest priority is 15. If **prty** is not specified, the default is 10.

retain = yes | no | initial
specifies whether the Process is retained on the TCQ in the Hold queue for re-execution after execution has completed.

yes specifies that the Process is retained on the Hold queue in HR status after execution. The Process must then be released manually through the **change process** command to cause it to be executed, or explicitly deleted through the **delete process** command. When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

no specifies that the Process is not retained. The default is **no**.

initial specifies that the Process is retained on the Hold queue in HR status and automatically executed each time the Process Manager initializes. The **startt** parameter should not be coded when **retain=initial** is coded. This causes the SUBMIT command to fail.

sacct = "*snode accounting data*"
specifies accounting data for the **snode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

The **sacct** parameter overrides any accounting data specified on the **process** statement of the submitted Process.

snode = [*nodename*] | [(*hostname* | *nnn.nnn.nnn.nnn*);(*port number* | *portname*)]
is a 1-256 alphanumeric character string that specifies the node name of the secondary node. The **snode** default value is the value coded in the **process** statement. It is required either on the **submit** command or **process** statement.

nodename is the node name of the adjacent Connect:Direct node. The secondary node name corresponds to an entry in the network map file.

hostname is the name of the host machine where the remote Connect:Direct is running. This is applicable only for TCP/IP.

nnn.nnn.nnn.nnn is the IP address of the remote Connect:Direct node. Each **nnn** is a decimal number from 0-255, inclusive. This is applicable only for TCP/IP.

portnumber | ***portname*** identifies the communications port for the Connect:Direct software. The ***portnumber*** is a decimal number from 1024-5535. This is applicable only for TCP/IP.

snodeid = (*id* [,*pswd*[,n*ewpswd*]])
specifies security user IDs and security passwords for the Process on the SNODE. Each **snodeid** parameter value can be 1-64 alphanumeric characters.

id specifies the userid to be used as a security ID on the **snode**.

pswd specifies the user password on the **snode**.

newpswd specifies the new password value. The user password is changed to the new value on the **snode** if the userid and old password are correct. The ***newpswd*** parameter is not valid if the **snode** is a Connect:Direct OpenVME node.

If ***pswd*** is specified, ***id*** also must be specified. If ***newpswd*** is specified, ***pswd*** also must be specified. These values must be coded in the order of ***id***, ***pswd***, and ***newpswd***.

startt = ([*date* | *day*] [,*hh*:*mm*:*ss* [am | pm]])

specifies that the Process is executed at a specified date and/or time. The Process is placed in the Timer queue in WS status. The **date**, **day**, and **time** values are positional subparameters. If **date** or **day** is not specified, a comma must precede **time**.

Note: The **startt** parameter should not be coded when **retain=initial** is coded. This causes the **submit** command to fail.

---

date specifies that the Process will execute on a specific date. You can specify the ***date*** subparameter in the format ***mm*/*dd*/*yyyy*** or ***mm*-*dd*-*yyyy***, where:

- ***mm*** indicates month
- ***dd*** indicates day
- ***yyyy*** indicates year

If only ***date*** is specified, ***time*** defaults to 00:00. The current date is the default.

day specifies the day of the week a Process is released for execution. Values are today, tomorrow, monday, tuesday, wednesday, thursday, friday, saturday, and sunday.

If only day is specified, time defaults to 00:00:00. For example, if a Process is submitted on Monday, with **monday** as the only **startt** parameter, the Process does not run until the following Monday when the time reaches 00:00:00.

hh*:*mm*:*ss* [am | pm]** specifies the hour (hh***), minute (***mm***), and second (***ss***) the Process should start. You can specify hour in either 12- or 24-hour format. If you use 12-hour format, you must specify **am** or **pm**. A space is required before **am** or **pm**. The default is 24-hour format. The default value is 00:00:00.

Note: When both **hold=yes** and a **startt** value are specified, the hold specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed on the Hold queue even if a start time is specified.

&symbolic name1 = variable string 1
&symbolic name2 = variable string 2
.
.
.
&symbolic namen = variable string n

is a symbolic parameter assigned a value. The value is substituted within the Process when the symbolic parameter is encountered. The value for the symbolic parameter must be in double quotation marks if it is a keyword or contains special characters.

Example

This example shows a Process that uses symbolics to allow you to specify the file and data set names at the time of submission. Process accounting data is specified for the **pnode** and **snode**.

```
copyseq     process     snode=dallas
                        pacct="dept-59"
                        sacct="dept-62"
step01      copy  from  (file=&file)
                  to    (file=&dsn
                        snode)
            pend
```

The following **submit** command specifies the file and data set names to be used in a file transfer.

```
submit     file=copyseq
           &file=myfile
           &dsn=abc;
```

The following Process is generated by the above input:

```
copyseq     process     snode=dallas
step01      copy  from  (pnode file=myfile)
                  to    (snode dsn=abc)
            pend
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 49

Connect:Direct OpenVME Copy Statement

The **copy** statement is comprised of a **from** clause that includes the source file name and a **to** clause that includes the destination file name. Additional parameters can be specified to further customize the file transfer operation.

To copy from one Connect:Direct system environment to another, refer to the appropriate **copy** from and to sections in this manual for those environments.

The **copy** statement execution results in a return code. See the **nn** parameter on page 396 for a list of standard return codes.

Format

The **copy** statement format includes the following parameters. Default values for parameters and subparameters are underlined. Required parameters are shown in bold characters.

Label	Statement	Parameters	
stepname	**copy**	**from**	(
			file=*filename* | dsn=*filename*
			pnode | snode
			sysopts="':datatype=text | binary:" "':xlate=no | yes:" "':xlate.tbl=<*pathname/filename*>:" "':strip.blanks=yes | no:" "':permiss=*nnn:*" "':pipe=yes:"
			)
			ckpt=no | *n*k |*n*m
			compress [[primechar = x'*xx*' | x'20' | c'*c*'] | extended]

Label	Statement	Parameters	
		to	(
			file=*filename* | dsn=*filename*
			pnode | snode
			sysopts=":datatype=text | binary:" ":xlate=no | yes:" ":xlate.tbl=<*pathname/filename*>:" ":strip.blanks=yes | no:" ":permiss=*nnn*:" ":pipe=yes:"
			disp=[(] new | mod | rpl [)]
			)

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

A label is not required on the **copy** statement.

Note: Although the Connect:Direct OpenVME label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

copy

identifies the statement with all its parameters as the **copy** statement.

Required Parameters

from

specifies that the subsequent parameters and subparameters define the source file characteristics.

to

specifies that the subsequent parameters and subparameters define the destination file characteristics.

Optional Parameters

ckpt = no | *n*k |*n*m

specifies the byte interval for checkpoint support, which allows restart of interrupted transmissions at the last valid checkpoint point and therefore reduces the time to retransmit the file. If the **ckpt** parameter and value are not specified, the default is the value specified by the ckpt.interval parameter in the Initialization Parameters file. Code this parameter between the **from** and **to** parameters.

no specifies no checkpointing.

***n*k** specifies a number of kilobytes.

***n*m** specifies a number of megabytes.

compress [[primechar = x'*xx*' | x'20' | c'*c*'] | extended]

specifies that the data is to be compressed, which reduces the amount of data transmitted as the file is copied from one node to another. The file will be automatically decompressed at the destination. The default subparameter for the **compress** parameter is **primechar=x'20'**. Use **compress** for text data or single-character repetitive data. Use **compress extended** for all other types of data. Code this parameter between the **from** and **to** parameters.

primechar specifies the primary compression character. The default value for primechar is **x'20**'.

The Connect:Direct software reduces the amount of data transmitted based on the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character will be compressed to one byte.
- Repetitive occurrences (ranging from 3-63) of any other character will be compressed to two bytes.

extended is used to search for repetitive strings of characters in data and compress them to codes that are transmitted and converted back to the original string during decompression. It is advantageous to specify this parameter when line transmission speeds are limited and data is repetitive.

disp = [(] new | mod | rpl [)]

defines the state that the target file should be in when it is opened. The **disp** parameter also determines how the file should be opened, either **new**, **mod**, or **rpl**.

new indicates the file must not already exist.

mod indicates that if the file already exists, data will be appended to the end of the file. If the file does not exist, then **mod** behaves as if **new** is specified.

rpl indicates that **copy to** will act as if **disp=new** was specified if the file does not already exist. If the file exists, then it will be overwritten. The default is **rpl**.

file = *filename* | dsn = *filename* | file = "*unix command* [;*unix command* [;*unix command*...]]"

The **file** or **dsn** parameter specifies the optional pathname and required file name. The file name can be enclosed in double quotation marks.

If **sysopts="pipe=yes"**, the **file** parameter specifies a UNIX command to execute. The command can be a command, program, or shell script including any options and arguments.

When specifying **file=*filename***, enclose the file name in double quotation marks when using a reserved word (statement name or keyword) for the file name.

pnode

When **pnode** is coded with the **from** parameter, the file to be copied resides on the primary node. When **pnode** is coded with the **to** parameter, the file is sent to the primary node. The **pnode** parameter is the default with the **from** parameter.

snode

When **snode** is coded with the **from** parameter, the file to be copied resides on the secondary node. When **snode** is coded with the **to** parameter, the file is sent to the secondary node. The default for the **to** parameter is **snode**.

sysopt=":datatype=text | binary:"
":xlate=no | yes:"
":xlate.tbl=<*pathname/filename*>:"
":strip.blanks=yes | no:"
":permiss=*nnn*:"
":pipe=yes:"

specifies system-specific parameters on the **copy** statement. The subparameters are specified in the same format as fields within a Connect:Direct OpenVME configuration file. They are a series of field names and values (***fldn=valn***), each of which is delimited by a colon. Enclose the string of subparameters in double quotes; for example:

```
":fld1=val1:fld2=val2:...:fldn=valn:"
```

datatype specifies the type of data contained within the file, either **text** or **binary**.

text indicates it is a text file. The default is **text**. The following shows the default source file attributes assigned for OpenVME text files, which are to be used if necessary by the remote Connect:Direct node:

- dsorg=ps
- recfm=vb
- lrecl=23036
- blksize=23040

binary indicates the file contains binary data. The following shows the default source file attributes assigned for OpenVME binary files, which are to be used if necessary by the remote Connect:Direct node:

- dsorg=ps
- recfm=u
- lrecl=0
- blksize=23040

xlate = no | yes indicates whether character translation should be performed using the default or user-supplied translation table. Typically, this translation is between ASCII and EBCDIC.

no specifies the translation is not performed. **no** is the default for binary files.

yes specifies the translation is performed. **yes** is the default for text files.

xlate.tbl =<*pathname*/*filename*> specifies that a translation table is to be used that is different from the default table used by the Connect:Direct software.

strip.blanks determines whether trailing blank characters will be removed from a line of text before it is written to the OpenVME text file.

yes specifies to remove trailing blank characters. The default is **yes**.

no specifies trailing blank characters will not be removed.

permiss=*nnn* specifies the OpenVME file permissions for a file being created by the copy operation. The **permiss** subparameter is ignored if it is specified for a file that already exists.

nnn is a 3-digit octal number that defines privileges for users. Each type of user (owner, group, and others, respectively) can be assigned read, write, and execute privileges.

The following table shows valid octal numbers and associated permissions for each **n** based on the binary numbering system:

Octal	Binary	Permissions
0	000	No permissions
1	001	Execute permission
2	010	Write permission
3	011	Write and execute permissions
4	100	Read permissions
5	101	Read and execute permissions
6	110	Read and write permissions
7	111	Read, write, and execute permissions

For example, permiss=634 indicates that the owner has read and write permissions, the group has write and execute permissions, and others have read permissions.

pipe = yes specifies that the pipe I/O function is activated. The pipe I/O function allows commands, programs, or shell scripts including any options and arguments to be copied to the destination file or from the source file.

Note: Checkpoint/restart is not supported for the pipe I/O function.

Example

In this example, the Process copies a binary file from a local OpenVME node to an OS/390 node:

```
* When copying, make sure the datatype is set to binary.  */
step01     copy  from  (file=a.out
                       pnode
                       sysopts=":datatype=binary:")
                 ckpt=lk
                 compress extended
                 to    (file=TESTAOUT
                       snode
                       disp=(rpl))
/*  If step01 succeeds, CD will copy the same file back to OpenVME.  */
step02     if    (step01 > 4 ) then
           exit
           eif
/*  Before copying the file back, delete it first.  */
step03     run task
                       pnode
                       sysopts="rm -f a.out"
step04     if          (step03 > 4 ) then
           exit
           eif
/*  Copy the file from OS390 to OpenVME.  */
/*  When copying, make sure the datatype is set to binary.  */
step05     copy  from  (file=TESTAOUT
                       snode)
                 ckpt=lk
                 to    (file=a.out
                       pnode
                       disp=(rpl)
                       sysopts=":datatype=binary:")
                       pend
```

The following Process copies a file from a local Connect:Direct node to a node named Atlanta:

```
copyseq     process     snode=atlanta
step01      copy  from  (file=myfile)
                  to    (file=yourfile
                        sysopts=":datatype=binary:")
            pend
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 50

Connect:Direct OpenVME Run Job Statement

The Connect:Direct OpenVME **run job** statement allows you to run UNIX commands in a Process. A UNIX command shell is invoked to execute one or more UNIX commands. The Process does not wait until the UNIX commands have finished running before executing the next step in the Process.

The execution of the **run job** statement produces a return code. See the **nn** parameter on page 396 for a list of standard return codes.

Note: The return code *does not* indicate whether the UNIX command was successfully executed. The return code only indicates whether Connect:Direct successfully submitted the **run job** statement.

Format

The Connect:Direct OpenVME **run job** statement format includes the following parameters. Required parameters are shown in bold characters. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**run job**	**dsn=dsn**[(member)]
		pnode | snode
		sysopts="*unix command* [;*unix command* [;*unix command*...]]"

Field Descriptions

Label
Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

Note: Although the Connect:Direct OpenVME label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the **run job** statement.

run job
identifies the statement with all its parameters as the **run job** statement.

Required Parameters

dsn = dsn[(member)]
specifies the name of the data set containing the job to be submitted. If the file is a PDS, the member containing the job must be specified. The data set containing the job must already exist on the node where the job will be submitted.

sysopts = "*unix command* [;*unix command* [;*unix command*...]]"
specifies a UNIX command to execute under a B shell. The command can be a command, program, or shell script including any options and arguments. Specify the command as if it were being issued at an OpenVME terminal.

The **run job** statement does not wait until the UNIX command is completed to complete execution. This is the distinguishing feature between **run job** and **run task**.

Once started, the UNIX command or commands in a **run job** statement will be permitted to continue execution. Connect:Direct OpenVME does not have control over the UNIX commands.

Optional Parameters

pnode | snode
specifies on which Connect:Direct node the **run job** statement is executed. The default is **pnode**.

Example

In this example, the application myjob is submitted to the **pnode** system. See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

```
jobstep  run job    sysopts="myjob param1"
```

Chapter 51

Connect:Direct OpenVME Run Task Statement

The Connect:Direct OpenVME **run task** statement allows you to run OpenVME commands in a Process. A UNIX command shell is invoked to execute one or more UNIX commands. The Process waits until the UNIX commands have finished running before executing the next step in the Process.

Caution: Do not specify programs in the **run task** statement that cannot complete without user intervention because they will not be completed.

Process statements that follow the **run task** statement do not execute until the submitted job completes. The execution of the **run task** statement results in a return code which is the exit code for the program executed using **run task**.

Format

The **run task** statement format includes the following parameters. Required parameters are shown in bold characters. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**run task**	(**pgm=program-name)**
		pnode | snode
		sysopts="*unix command* [;*unix command* [;*unix command*...]]"

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the **run task** statement.

Note: Although the Connect:Direct OpenVME label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

run task

identifies the statement with all its parameters as the **run task** statement.

Required Parameters

pgm = program-name

specifies the name of the program to be attached as the subtask. The program runs on the node specified and has access to the DD cards allocated on that node only.

sysopts = "*unix command* [;*unix command* [;*unix command*...]]"

specifies one or more UNIX commands to execute. The command can be a UNIX command, program, or shell script including any options and arguments. Specify the command as if it were being issued at an OpenVME terminal.

The **run task** statement waits until the UNIX command has completed execution.

The UNIX command or commands executed in a **run task** step can be terminated by a **flush process force** or a **stop force** or **stop immediate** command.

Optional Parameters

pnode | snode

specifies on which Connect:Direct node the **run task** will be executed. The default is **pnode**.

Example

In this example, the user-supplied program is specified in a **run task** statement. The program requires two parameters. The program is located on the primary node. The **run task** statement is contained within a Process that is also located on the **pnode**.

```
taskstep run task sysopts="grep -e CTRC s19931215.001 > stat.txt"
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 52

Connect:Direct OpenVME Submit Statement

During execution of a Connect:Direct Process, the **submit** statement causes another Connect:Direct Process to be submitted to either the **pnode**, which is the node with Process control or the **snode**, which is the secondary node that participates in the Process.

The Process submitted must reside in a file on the node where the **submit** statement executes; this node is referred to as the **subnode**.

Note: The **submit** statement described in this chapter is not the same as the **submit** command. The **submit** statement is used within a Process to submit another Process. See the *Connect:Direct OpenVME User's Guide* for **submit** command syntax and parameters.

Any parameter values specified in the **submit** statement override the parameter values contained in a **process** statement named by the **dsn** | **file** parameter.

The execution of the **submit** statement produces a return code. See page 396 for a list of standard return codes.

Format

The **submit** statement format includes the following parameters. Required parameters are shown in bold characters. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**submit**	**file=*filename***
		class=*n*
		hold=yes | no | call
		newname=*new process name*
		notify=*username@hostname*
		pacct="*pnode accounting data*"
		pnodeid=(*id* [,*pswd*])

Label	Statement	Parameters
		prty=*nn*
		retain=yes | no | initial
		sacct="*snode accounting data*"
		snode=*nodename* | *hostname* | *nnn.nnn.nnn.nnn* [;*portname* | ;*nnnnn*]
		snodeid=(*id* [,*pswd*[,*newpswd*]])
		startt=([*date* | *day*][,*hh*:*mm*:*ss*[am | pm]])
		subnode=pnode |snode
		&symbolic name1=variable string 1 &symbolic name2=variable string 2 . . . &symbolic namen=variable string n

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the **submit** statement

Note: Although the Connect:Direct OpenVME label may be up to 256 characters long, labels for many of the Connect:Direct platforms cannot exceed eight characters.

submit

identifies the statement with all its parameters as the **submit** statement.

Required Parameters

file = *filename*
specifies the name of the file that contains the Process. The file name may include a pathname indicating the location of the Process. There is no practical limit on the character length of the **file** parameter.

Note: A pathname recognized in *cli* is relative to the current working directory. An absolute path (starting from root "/") will always work.

Enclose the file name in double quotation marks when using a reserved word (statement name or keyword) for the file name.

Optional Parameters

class = *n*
determines the node-to-node session on which a Process can execute. A Process may execute on the class specified or any higher session class. The default class is specified as the **sess.default** parameter in the Initialization Parameters file. The **class** default is 1.

hold = yes | no | call
specifies how the Process is handled in the Hold queue.

yes specifies that the Process is held in the Hold queue in HI status until it is released by a **change process** command. When both **hold=yes** and a **startt value** are specified, the **hold** parameter takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

When a Process is submitted with **retain=yes** and **hold=no** or **hold=call,** the **hold** parameter is ignored.

no specifies that the Process is not held on the Hold queue. It is executed as soon as resources are available. The default is **no**.

call specifies that the Process is held (no prompt returns) until the remote node (**snode**) connects to the local node (**pnode**). At that time, the Process is released for execution. The Process is also released when another Process on the local node connects to the **snode**.

newname = *new process name*
specifies the new name to be given to the Process. This value overrides the label on the **process** statement. There is no practical limit on the character length of the **newname** parameter.

notify = *username@hostname*
specifies a user name and host computer name to use in alerting a user whenever a step completes within a Process. Notification is accomplished through the normal UNIX mail facility.

pacct = "*pnode accounting data*"
permits the user to specify an arbitrary string to be used as accounting information for the **pnode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

pnodeid = (*id* [,*pswd*])
specifies security information for the Process on the PNODE. Each **pnodeid** parameter value can be 1-64 alphanumeric characters long:

id specifies the userid to be used on the **pnode**.

pswd specifies a user password on the **pnode**.

If ***pswd*** is specified, ***id*** also must be specified. These values must be coded in the order of ***id*** and ***pswd***.

prty = *nn*
specifies the priority of the Process on the Transmission Control Queue (TCQ). The **prty** parameter is used only for Process selection. A Process with a higher priority will be selected for execution before a Process with a lower priority. The value specified for the **prty** parameter does not affect the priority during transmission. Values range from 0-15. The highest priority is 15. If **prty** is not specified, the default is 10.

retain = yes | no | initial
specifies whether the Process is retained on the TCQ in the Wait queue for re-execution after execution has completed.

yes specifies that the Process is retained on the Hold queue in HR status after execution.

When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

no specifies that the Process is not retained. The default is **no**.

initial specifies that the Process is retained on the Hold queue in HR status and automatically executed each time the Process Manager initializes. The **startt** parameter should not be coded when **retain=initial** is coded. This causes the **submit** statement to fail.

sacct = "*snode accounting data*"
specifies accounting data for the **snode**. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

The **sacct** parameter overrides any accounting data specified on the **process** statement of the submitted Process.

snode = *nodename* | *hostname* | *nnn.nnn.nnn.nnn* [;*portname* | ;*nnnnn*]
is a 1-256 alphanumeric character string that specifies the node name of the secondary node. This parameter overrides the value specified in the **process** statement. The **snode** default value is the value coded in the **process** statement. It is required either on the **submit** command or **process** statement.

nodename is the name of the remote Connect:Direct node. The secondary node name corresponds to an entry in the Network Map file.

hostname is the name of the host machine where the remote Connect:Direct software is running. This is applicable only for TCP/IP.

nnn.nnn.nnn.nnn is the IP address of the remote Connect:Direct node. Each **nnn** is a decimal number from 0-255, inclusive. This is applicable only for TCP/IP.

[;*portname*|;*nnnnn*] identifies the communications port for the Connect:Direct software. The ***nnnnn*** value is a decimal number from 1024 to 65535.

snodeid = (*id* [,*pswd*[,*newpswd*]])
specifies security userids and security passwords for the Process on the **snode**. Each **snodeid** parameter can contain 1-64 alphanumeric characters.

id specifies the security ID that is passed to a security exit.

pswd specifies the user password on the **snode**.

newpswd is the new password value. The user password is changed to the new value on the **snode** if the userid and old password are correct. The ***newpswd*** parameter is not valid if the **snode** is a Connect:Direct OpenVME node.

If ***pswd*** is specified, ***id*** must also be specified. If ***newpswd*** is specified, ***pswd*** must also be specified. These values must be coded in the order of ***id***, ***pswd***, and ***newpswd***.

startt = ([*date* | *day*] [,*hh*:*mm*:*ss* [am | pm]])
specifies that the Process will be executed at the specified date and/or time. The Process is placed in the Timer queue in WS status. The ***date***, ***day***, and ***time*** are positional parameters. If ***date*** or ***day*** is not specified, a comma must precede ***time***.

Note: The **startt** parameter must not be coded with **retain=initial** or **retain=yes**.

date specifies that the Process is to be held until the desired date. The ***date*** value may be specified in the format ***mm*/*dd*/*yyyy*** or ***mm*-*dd*-*yyyy***, where:

- ***mm*** indicates month.
- ***dd*** indicates day.
- ***yyyy*** indicates year.

If only ***date*** is specified, ***time*** defaults to 00:00:00. The current date is the default.

day specifies the day of the week a Process is to be released for execution. Values are today, tomorrow, monday, tuesday, wednesday, thursday, friday, saturday, and sunday.

hh*:*mm*:*ss* [am | pm]** specifies the hour (hh***), minute (***mm***), and second (***ss***) that the Process should start. Hour may be specified in either 12- or 24-hour format. If the 12-hour format is used, then **am** or **pm** must be specified. A space is required before **am** or **pm**. The default is 24-hour format. The default value is 00:00:00 which indicates midnight.

If only ***day*** is specified, time defaults to 00:00:00. For example, if a Process is submitted on Monday, with **monday** as the only **startt** parameter, the Process does not run until the following Monday when the time reaches 00:00:00.

Note: When both **hold=yes** and a **startt** value are specified, the hold specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed on the Hold queue even if a start time is specified.

subnode = pnode | snode

specifies the node on which the Process will be submitted. The Process must reside on the node on which it is being submitted.

pnode specifies that the Process is submitted on the node that has Process control. The default value is **pnode**.

snode specifies that the Process is submitted on the node participating in, but not controlling, Process execution.

&symbolic name1 = variable string 1
&symbolic name2 = variable string 2
.
.
.
&symbolic namen = variable string n

specifies the value for a symbolic parameter in the Process. This default can be overridden when submitting the Process.

The value for the symbolic parameter must be in double quotation marks if it is a keyword or contains special characters.

The symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Example

This example shows how you can copy a file from your local node to a node in Dallas, which will then copy it to a node in Tampa.

A user issues a **submit** command to initiate a Process.

```
submit file=process1
```

The **submit** statement contained within process1 specifies the Process name of process2. As a result, process2 will also be submitted.

```
process1     process     snode=dallas
copystep     copy  from  (file=myfile)
                   to    (dsn=yourfile)
                         disp=(new,keep)
substep      submit      file=process2
                         subnode=snode
                         &dsn=yourfile
             pend
```

The **copy** statement in process2 will copy yourfile to the file newfile in Tampa.

```
process2     process     snode=tampa
copystep     copy  from  (dsn=&dsn)
                   to    (dsn=newfile)
                         disp=(new,keep)
             pend
```

The symbolic parameter &dsn is converted in process2 to the value of yourfile, which was specified in the **submit** statement in process1.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 53

Connect:Direct OpenVME Conditional Statements

The conditional statements (**if then**, **else**, **eif**, **goto**, and **exit**) permit you to alter the sequence of Process execution based on the completion of the previous step in the Process.

Formats

Formats for Connect:Direct OpenVME conditional statements follow. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	if	**(*label condition nn*) then**
		(*process steps*)
	else	
		(*alternative process steps*)
	eif	
optional	goto	label
	exit	

Statement Descriptions

if then

specifies that the Connect:Direct system executes a block of Connect:Direct statements based on the completion code of a Process step. An **eif** statement must be used in conjunction with an IF THEN statement. See the following *Field Descriptions* section for details.

else
designates a block of Connect:Direct statements that execute when the **if then** condition is not satisfied. No parameters exist.

eif
is required for specifying the end of the **if then** or **if then else** block of statements. No parameters exist.

goto
moves to a specific step within a Process. See the following *Field Descriptions* section for details.

exit
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

Required Parameters

condition
The ***condition*** parameter specifies the type of comparison that will be performed. This condition checking can be based on such comparisons as *equal to*, *greater than*, or *less than*. The valid conditions are as follows:

== | **=** | **eq** specifies that the return code must be equal to the value **nn** for the condition to be satisfied.

<> | **!=** | **ne** specifies that the return code must not equal the value **nn** for the condition to be satisfied.

>= | **=>** | **ge** specifies that the return code must be greater than or equal to the value **nn** for the condition to be satisfied.

> | **gt** specifies that the return code must be greater than the value **nn** for the condition to be satisfied.

<= | **=<** | **le** specifies that the return code must be less than or equal to the value **nn** for the condition to be satisfied.

< | **lt** specifies that the return code must be less than the value **nn** for the condition to be satisfied.

nn
This parameter specifies the numeric value to be used for return code checking. If coded as **x'*nn*'**, it is a hexadecimal value. Any other coding indicates it is decimal. Standard return codes are as follows:

- 0 indicates successful completion.
- 4 indicates warning.

- 8 indicates error.
- 16 indicates catastrophic error.

then
specifies subsequent processing based on the other specified parameters.

Optional Parameters

label
Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in column one. The label consists of a 1-256 character alphanumeric string.

Statement names and keywords are reserved and cannot be used as labels.

A label is not required on the **if** statement.

Note: Although the Connect:Direct OpenVME label may be up to 256 characters, labels for many of the Connect:Direct platforms cannot exceed eight characters.

Example

In this example, the Connect:Direct OpenVME Process contains all of the conditional statements. A description of the Process follows.

```
copy1     process     snode=chicago
step01    copy  from  (file=abc pnode)
                to    (file=def snode)
step02    if          (step01 gt 4) then
                      goto step07
          else
step03    runjob      sysopts="myjob"
          eif
step04    if          (step03 >= 8) then
          exit
step05    if          (step03 lt 4) then
step06    copy  from  (file=xyz pnode)
                to    (file=uvw snode)
          eif
          exit
step07    run task    sysopts=“verify”
          exit
          pend
```

copy01 is the **process** statement defining the secondary node as chicago.

step01 copies file abc on the **pnode** to file def on the **snode**.

step02 checks the completion code of step01. If step01 fails (return code greater than 4), step07 executes. If step01 completes with a return code of 4 or less, step03 executes.

step03 submits a job, myjob, on the **pnode**.

step04 checks the completion code of step03. If step03 fails with a code of 8 or greater, the Process terminates. Otherwise, step05 executes.

step05 checks the completion code from step03. If less than 4, indicating the step completed without error, step06 executes.

step06 copies file xyz on the **pnode** to file uvw on the **snode**. The Process will then exit.

step07 only executes if step01 fails. The program verify runs, sending an Operation Failed Message to the console operator.

pend marks the end of a Process.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 54

Connect:Direct OpenVME Pend Statement

The **pend** statement marks the end of a Connect:Direct OpenVME Process. There are no parameters associated with the **pend** statement.

And example of the **pend** statement usage follows.

Example

The following Process copies a file from a local node to a node named Atlanta. The **pend** statement indicates the end of the Process.

```
copyseq     process     snode=atlanta
step01      copy  from  (file=myfile)
                  to    (dsn=yourfile)
            pend
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 55

Connect:Direct Windows Process Statement

The Connect:Direct Windows **process** statement defines the beginning of a series of one or more statements that specify functions to be performed. A Process is made up of a required **process** statement, followed by one or more other statements, such as **copy**, **run job**, **run task**, and **submit**.

A Connect:Direct Windows **process** statement defines the attributes of a Connect:Direct Process and must be the first statement in a Process.

Format

The Connect:Direct Windows **process** statement format includes the following parameters. All platform-specific parameters and subparameters are listed along with their possible values. The required parameters and subparameters are in bold print. Default values for parameters and subparameters are underlined.

Label	Statement	Parameters
process name	**process**	class=*n*
		execprty=*nn*
		hold=yes | no | call
		pnode=*nodename*
		localacct=*"pnode node accounting data"*
		pnodeid=(id [,*pswd*])
		notify=*username*
		prty=*nn*
		snode=[*nodename]*|*[hostname*|*IPaddress;portnumber*|*servicename]* The **snode** parameter must be specified on the **process** statement or **submit** command.

Label	Statement	Parameters
		remoteacct=*"snode accounting data"*
		snodeid=(id [,*pswd*[,*newpswd*]])
		retain=yes | no | initial
		startt=([*date* | *day*][,*time* [am | pm]])
		&symbolic name1=variable string 1 *&symbolic name2=variable string 2* . . . *&symbolic namen=variable string n*

Field Descriptions

Process name

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in position one. The label consists of a 1-8 character alphanumeric string.

Note: Statement names, parameter names, and subparameter names are reserved words and cannot be used as labels.

The **Process** Statement Is The Only Statement That Requires A Label. The Label Is Used To Identify The Process In Any Messages Or Statistics Relating To This Process.

The label field defines a Process name that may be used as the **pname** parameter value on the **change process, delete process, select process,** and **select statistics** commands as the name of the target Process.

Required Parameters

The **process** statement has no required parameters. However, the **snode** keyword must be specified on either the **submit** command when submitting the Process or the **process** statement.

Optional Parameters

class = *n*

determines the node-to-node session on which a Process can execute. A Process may execute in the class specified or any higher session class up to the maximum allowed

for an snode (**sess.pnode.max**). The default class is specified as the **sess.default** parameter in the initialization parameters. The **class** default in the initialization parameters is **1**.

execprty = *nn*
specifies the relative operating system priority to be assigned to the Process. The **execprty** keyword determines how Windows schedules the Session Manager, which runs the Process, for execution relative to other tasks in the system. A Process with a higher priority receives more opportunities for execution from the operating system than a Process with a lower priority. Values range from 1-15. If **execprty** is not specified, the default is **7**.

Note: Scheduling Processes to run with a high priority may have an adverse effect on the execution of other applications in the system.

---

hold = yes | no | call
specifies that a Process is to be held and the circumstances or methods under which it may be released.

yes indicates that the Process is held in the Hold queue until it is explicitly released by a **change process** command. When both **hold=yes** and **startt** values are specified, the **hold** specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

no indicates that the Process is not held and is executed as soon as resources are available. The default is **no**.

call indicates that the Process is held until receipt of an incoming session from the Connect:Direct snode. At that time, the Process is released for execution. The Process is also released when another Process on the pnode connects to the snode.

Note: When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

---

pnode = *nodename*
is an alphanumeric character string that specifies the Connect:Direct pnode name.

nodename is the node name of the Connect:Direct pnode. The pnode name corresponds to the name parameter in the initialization parameters. This parameter is for Process documentation purposes only. See the *Connect:Direct Windows Administration Guide* for more information on initialization parameters.

localacct = "***pnode node accounting data***"
specifies accounting data for the pnode. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

pnodeid = (*id* [, *pswd*])
specifies security information to be used on the pnode for the Process. Each **pnodeid** parameter value can be 1-20 alphanumeric characters long.

id specifies the userid to be used as a security ID on the pnode. This must be the name of an existing user account. It is not case-sensitive.

pswd specifies a user password on the pnode. It is case-sensitive.

If ***pswd*** is specified, ***id*** must be specified also. These values must be coded in the order shown.

notify = *username*
specifies a username (valid mail ID) to notify when a step within the Process is complete.

prty = *nn*
specifies the selection priority of the Process in the Transmission Control Queue (TCQ). The **prty** keyword influences the order in which Processes are selected for execution. A Process with a higher priority is selected for execution before a Process with a lower priority. This value does not affect the priority during execution.

If **prty** is not specified, the value from **proc.prio.default** in the initialization parameters is used. If no value is specified in the initialization parameters, the default is 10.

snode = [*nodename]* | *[hostname* | *IPaddress;portnumber*|*servicename*]
is a 1-16 alphanumeric character string that specifies the Connect:Direct snode name. This parameter is required on either the **submit** command or the **process** statement.

nodename is the node name of the Connect:Direct snode. The snode name corresponds to a Connect:Direct **remote.node** object in the network map.

hostname is the name of the host machine where the Connect:Direct snode is running. This is applicable only for TCP/IP.

IPaddress is the IP address of the Connect:Direct snode. The IP format is nnn.nnn.nnn.nnn. Each **nnn** is a decimal number from 0-255, inclusive. This is applicable only for TCP/IP.

portnumber*|*servicename identifies the communications port for the Connect:Direct software. The ***portnumber*** is a decimal number from 1024-65535. This is applicable only for TCP/IP.

remoteacct = "*snode accounting data*"
specifies accounting data for the snode. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

snodeid = (*id* [,*pswd*[,*newpswd*]])
specifies security userids and security passwords for the Process on the snode. Each **snodeid** parameter value can be 1-20 alphanumeric characters. When copying to a file server other than the one containing Connect:Direct, **snodeid** is required.

id specifies the userid used as a security ID on the snode.

***For Connect:Direct Tandem nodes*:** The **snodeid** parameter specifies the Tandem group number and user number. The valid range for these numbers is 0-255.

pswd specifies the user password on the snode.

newpswd specifies a new password value. The user password is changed to the new value on the snode if the userid and old password are correct and the snode supports this optional parameter. The ***newpswd*** parameter is not valid if the snode is a Connect:Direct Windows node.

If **snodeid** is specified, ***id*** must be specified also. If ***newpswd*** is specified, ***pswd*** must be specified also. These values must be coded in the order of ***id***, ***pswd***, and ***newpswd***.

retain = yes | no | initial

specifies whether the Process is retained in the TCQ after its initial execution is complete. Using **startt** with **retain=yes** allows you to have Processes automatically release at the interval defined by **startt**.

yes specifies that the Process is retained in the Hold queue in HR status after execution. Processes submitted with **retain=yes** remain in the queue unless explicitly released if **startt** is not coded. When released, it is again cloned, and the clone runs immediately. The Process must then be released manually through the **change process** command to cause it to be executed again or explicitly deleted through the **delete process** command.

Processes submitted with both **retain=yes** and **startt** are cloned. If **startt** specifies both day of the week and a time, the Process is cloned and runs every week on that day and at that time. If **startt** specifies day only, the Process is cloned and runs every week on that day at 12:00 a.m. If **startt** specifies time only, the Process is cloned and runs every day at that time.

no specifies that the Process is deleted immediately after execution. The default is **no**.

initial specifies that the Process is retained in the Hold queue in HR status and automatically executed each time the Connect:Direct Windows server initializes.

Processes submitted with **retain=initial** are only searched for when the server initially starts. At that time, they are cloned, and the clone executes immediately.

Caution: Setting the **startt** parameter and the **retain=initial** parameter causes the **submit** command to fail.

Note: When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

---

startt = ([*date* | *day*] [,*time* [am | pm]])

specifies that the Process will be executed at a specific date and/or time. The Process is placed in the Timer queue in WS status. The ***date***, ***day***, and ***time*** values are positional subparameters. If ***date*** or ***day*** is not specified, a comma must precede ***time***.

Caution: The **startt** parameter should not be coded when **retain=initial** is coded. This causes the **submit** command to fail.

---

date specifies that the Process is to be held until the desired date. The current date is the default. You can specify the ***date*** subparameter in the format ***mm/dd/yyyy*** or ***mm-dd-yyyy***, where:

- ***mm*** indicates month and can be a one- or two-digit number
- ***dd*** indicates day and can be a one- or two-digit number
- ***yyyy*** indicates year and can be a two- or four-digit number

If *date* is specified in **startt**, the Process runs only on the date specified. If *day* is specified, the Process runs every week on that day. If *time* is specified, the Process runs at that time on the days indicated by *day* or *date*, or every day if a day or date is not specified.

day specifies the day of the week the Process will be released for execution. Values are **today, tomorrow, monday, tuesday, wednesday, thursday, friday, saturday,** and **sunday**.

time specifies the time the Process should start. The format is ***hh:mm:ss*** **[am|pm]**, specifies the hour (***hh***), minute (***mm***), and second (***ss***) the Process will start. You can specify hour in either 12- or 24-hour format. If you use 12-hour format, you must specify **am** or **pm**. A space is required before **am** or **pm**.

Seconds (***ss***) are optional. The default is 24-hour format. The default value is 00:00:00, which indicates midnight.

If only ***day*** or ***date*** is specified, ***time*** defaults to 00:00:00 (midnight). For example, if a Process is submitted on Monday, with **monday** as the only **startt** parameter, the Process does not run until the following Monday at midnight.

Note: When both **hold=yes** and a **startt** value are specified, the hold specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

&symbolic name1 = variable string 1
&symbolic name2 = variable string 2
.
.
.
&symbolic namen = variable string n

is a symbolic parameter assigned a value. The value is substituted within the Process when the symbolic parameter is encountered. The value for the symbolic parameter must be in double quotation marks if it is a parameter, subparameter, or statement name or if it contains special characters.

Note: Statement names, parameter names, and subparameter names are reserved and cannot be used as symbol names.

Special Characters

Symbol names begin with an ampersand and are terminated by delimiters. Symbol names and their values cannot exceed 126 characters. The following table lists the delimiters recognized by Connect:Direct Windows.

Delimiters	Description
	blank
	tab

Delimiters	Description
	carriage return
	line feed
<	less than sign
>	greater than sign
*	asterisk
(	open parenthesis
)	closed parenthesis
/	slash
\	backslash
:	colon
;	semicolon
,	comma
.	period
'	single quotation mark
"	double quotation mark
=	equal sign
{	opening brace
}	closing brace
[	opening bracket
]	closing bracket
&	ampersand

Single or Double Quotation Marks

The rules for using single or double quotation marks are as follows:

- Substitution is performed only when the string is enclosed in double quotes. If you are substituting in a string that is already in double quotes, such as the **sysopts** value, no additional double quotes are needed. Double quotes added for substitution purposes, but not needed, are stripped off.
- In substitution variable definitions, such as the **process** statement or **submit** command, enclose the value in double quotes if it contains any special characters. Substitution values that do not contain special characters do not need to be enclosed in double quotes.
- If the string is enclosed in single quotes, no substitution is done.

The following example shows the use of quotation marks to allow special characters or blanks to be embedded within a symbolic parameter or subparameter value:

```
&NAME="C:\MYFILE.TXT"
FILE="&NAME"
```

The previous example converts to the following string:

```
FILE=C:\MYFILE.TXT
```

Concatenation

The rules for using periods are as follows:

- Single periods after variable names can be used to concatenate variables to constants and are dropped when the variable is resolved to its value.
- If two or more periods are found after variable names, the variable is resolved, one period is dropped, and the rest are retained.

The following example shows the use of periods to concatenate variables to constants:

```
&NAME=MYFILE
FILE="C:\&NAME..TXT"
```

The previous example converts to the following string:

```
FILE=C:\MYFILE.TXT
```

Spanning Lines

Strings that span multiple lines and contain substitution variables are not allowed.

Example

In this example, the Process uses symbolics to allow you to specify the file names at the time of submission. Process accounting data is specified for the pnode and snode.

```
copyseq     process     snode=OS390.dallas
                        localacct="dept-59"
                        remoteacct="dept-62"
step01      copy  from  (file="&homefile")
                  to    (file="dalfile"
                        snode)
```

The following **submit** command specifies the file names to be used in a file transfer using the Process in the previous example.

```
submit     file=copyseq
           &homefile="c:\mydir\myfile.txt"
           &dalfile="user01.newfile"
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 56

Connect:Direct Windows Copy Statement

The **copy** statement allows you to exchange text or binary files with remote Connect:Direct nodes.

The **copy** statement is comprised of a **from** clause that includes the source file name and a **to** clause that includes the destination file name. Additional parameters can be specified to further customize the file transfer operation.

To copy from one Connect:Direct system environment to another, refer to the appropriate *Copy Statement* chapters for those environments in this guide.

The **copy** statement execution produces a return code. See the ***nn*** parameter definition on page 441 for a list of standard return codes.

Format

The **copy** statement format includes the following parameters. Default values for parameters and subparameters are underlined. Required parameters are shown in bold print.

Label	Statement	Parameters	
stepname	**copy**	**from**	(
			file=*filename* | dsn=*filename*
			pnode | snode
			sysopts="datatype(text | binary) user_data xlate(no | yes) xlate.tbl(*pathname/filename*) strip.blanks(yes | no | i) strip.oneable(yes | no) codepage=(*source codepage*, *destination codepage*)"
			)

Label	Statement	Parameters	
		to	(
			file=*filename* | dsn=*filename*
			pnode | snode
			sysopts="datatype(text | binary) user_data xlate(no | yes) xlate.tbl(*pathname/filename*) strip.blanks(yes | no) strip.oneable(yes | no) codepage=(*source codepage*, *destination codepage*) acl(*operation,rightslist,accountname* *[;operation,rightslist,accountname* *[operation,rightslist,accountname [;...]]*) attributes(*physattr*)"
			disp=[{]*initial*,*normal*,*abnormal*[}]
			)
			ckpt=no | *nnnnnnnn*k | *nnnnnnnn*m
			compress [[primechar = x'*xx*' | x'20' | c'*c*'] | extended]

Note: Connect:Direct Windows does not support, within Process statements, filenames that contain the equal sign (=) or ampersand (&) characters.

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels (***stepname***). A label is any character or character string beginning in position one. The label consists of a 1-8 character alphanumeric string.

Note: Statement names and keywords are reserved and cannot be used as labels.

A label is not required on the **copy** statement.

Statement

copy identifies the statement with all its parameters as the **copy** statement.

Required Parameters

from
specifies that the subsequent parameters and subparameters define the source file characteristics.

to
specifies that the subsequent parameters and subparameters define the destination file characteristics.

file=*filename* | *dsn=filename*
specifies the 1-256 character required file name. This parameter must be specified for both the from and to parameters. A path specification and file server name are optional. If a remote file server name is specified as part of the file name, the remote server must be a Windows version 3.51 or later system.

Connect:Direct Windows supports the string (*) and character (?) wildcards, allowing you to copy multiple files from a source directory to a target directory with a single copy statement. See page 419 for wildcard copy examples.

Specify the UNC form of the filename if the file is not on a drive directly connected to the Windows server where Connect:Direct resides. If the file is on the Windows server where Connect:Direct is installed, you can specify the drive letter.

The UNC name format is as follows:

```
\\servername\share point name\filename
```

Replace the ***servername*** with the name of the Windows server where data resides. The ***share point name*** represents the name under which the remote Windows server shares the directory you want to access. The ***filename*** specifies the name of the file and includes any subdirectories, if appropriate.

The file name can be enclosed in double quotation marks when you use a reserved word (statement name, parameter name, or subparameter name) for the file name.

Optional Parameters

ckpt=no | *nnnnnnnn*k | *nnnnnnnn*m
specifies the byte interval for checkpoint support. Checkpoint support allows restart of interrupted transmissions at the last valid checkpoint, reducing the time necessary to retransmit the file. If the **ckpt** parameter and value are not specified, the default is the value specified by the **ckpt.interval** parameter in the initialization parameters.

no specifies no checkpointing.

***nnnnnnnn*k** specifies the number of kilobytes.

***nnnnnnnn*m** specifies the number of megabytes.

codepage=(*source codepage, destination codepage*)
determines which codepage is used to translate a file.

Note: If you do not identify two parameters (source codepage, destination codepage) in the **from** statement or in the **to** statement, the codepage listed as the source is converted to UTF-8, sent, and then converted to the codepage identified in the **to** statement at the destination location. When this occurs, the file can be translated incorrectly.

Three methods can be used to translate files:

- **sysopts=codepage(*source codepage, destination codepage*)**

  This definition can be used either in the **from** or **to** statement and identifies the codepages used for translating a file either before it is transferred (**from**) or after it is received at the destination location (**to**).

- **from sysopts=codepage=*source codepage***
  to sysopts=codepage=*destination codepage*

  This definition translates the file using the source codepage defined in the **from** statement to UTF-8 format, transfers the file, and then translates it at the destination from UTF-8 to the codepage defined in the **to** statement. See the preceding note.

- **from sysopts=codepage(*source codepage, transitional codepage*)**
  to sysopts=codepage(*transitional codepage, destination codepage*)

  This definition translates the file using the source codepage defined in the **from** statement to the transitional codepage, transfers the file, and then translates it at the destination location from the transitional codepage to the destination codepage defined in the **to** statement. If UTF-8 is used as the transitional codepage, then this translation method performs the same as the second translation method described above.

Codepage translation supports the translation of text or binary files. When translating text files, codepage translates one line at a time. The trailing line feed character is removed from the text line. If the **strip.blanks** parameter is set to **yes**, trailing blanks are removed from the file and a line feed is appended to the line of text.

compress [[primechar = *x'xx'* | x'20' | c*'c'*]| extended]
specifies that the data is compressed to reduce the amount of data transmitted as the file is copied from one node to another. The file is automatically decompressed at the destination. Use prime-character compression for text data or single-character repetitive data. Use **compress extended** for all other types of data.
primechar specifies the primary compression character. This character is specified as a hexadecimal constant (**x'*xx*'**) or a character constant (**c'*c*'**). The default value for primechar is **x'20**'.

Connect:Direct Windows reduces the amount of data transmitted according to the following rules:

- Repetitive occurrences (ranging from 2-63) of the primary compression character are compressed to one byte.

- Repetitive occurrences (ranging from 3-63) of any other character are compressed to two bytes.

extended searches for repetitive strings of characters in data and compresses them to codes that are transmitted and converted back to the original string during decompression. This method is advantageous when line transmission speeds are limited and data is repetitive.

disp = (*initial, normal, abnormal*)

specifies the status of the file on the receiving node. This parameter is only valid when copying to a Connect:Direct OS/390 system. Subparameters are:

initial specifies the status of the file before the Process executes. This subparameter is required. Valid options for this subparameter are as follows:

new specifies that the Process step will create the destination file. The **copy** step fails if the file exists.

old specifies that the destination file already exists. The Process has exclusive control of the file. The **copy** step fails if the file does not exist.

mod specifies that the Process step modifies the file by appending data at the end of the file. If a system failure occurs when **mod** is specified, the system is designed not to restart even if CKPT is specified; data loss or duplication would be difficult to detect. The **copy** step fails if the file is not found.

rpl specifies that the destination file replaces any existing file or, if none exists, allocates a new file. **rpl** is the default.

shr specifies that the destination file already exists. The file can be used simultaneously by another job or Process. The **copy** step fails if the file is not found.

normal specifies the disposition of the file if the **copy** step is successful. This subparameter is optional. Valid destination file dispositions are as follows:

keep specifies that the system keeps the file after the Process step completes.

abnormal specifies the disposition of the file after an abnormal Process step termination resulting in a nonzero completion code. This subparameter is optional. Valid destination file dispositions are as follows:

keep specifies that the system keeps the file after the Process step terminates abnormally or with a nonzero return code. **keep** is the default.

delete specifies the system deletes the file if the Process step terminates abnormally.

pnode | snode

When **pnode** is coded with the **from** parameter, the file to be copied is sent from the pnode. When **pnode** is coded with the **to** parameter, the file is sent from the snode. The **pnode** parameter is the default value for the **from** parameter.

When **snode** is coded with the **from** parameter, the file to be copied is sent from the snode. When **snode** is coded with the **to** parameter, the file is sent to the snode. The **snode** parameter is the default with the **to** parameter.

sysopts=*fld1(val1) fld2(val2) ...fldn(valn)*

specifies system-specific parameters on the **copy** statement. The parameters are a series of field names and values ***fldn(valn)***, each of which is delimited by a space. The entire string is enclosed in double quotes. For example:

```
"xlate(yes) xlate.tbl(tbl1)"
```

Specifying Sysopts for from and to Attributes

The **sysopts** subparameters valid for all file copies are:

datatype(text|binary) specifies the type of data contained within the file.

text indicates it is a text file and that trailing blanks will be stripped unless you specify **strip.blanks(no).** Translation of the data is performed unless you also specify **xlate(no).** The default is **text** for all nodes, except Windows nodes. The following list shows the default source file attributes assigned for Windows text files, which are used if necessary by the remote Connect:Direct node:

- dsorg=ps (physical sequential organization)
- recfm=vb (variable-length records, blocked if the destination operating system supports it)
- lrecl=23036 (maximum record length of 23036)
- blksize=23040 (maximum block size of 23040, if the destination operating system supports record blocking)
- **binary** indicates the file contains binary data. The default is **binary** for Windows nodes. The following shows the default source file attributes assigned for Windows binary files, which are used if necessary by the remote Connect:Direct node:
- dsorg=ps (physical sequential organization)
- recfm=u (undefined record format)
- lrecl=0 (undefined record length)
- blksize=23040 (maximum block size and record size of 23040)

user_data specifies any user-defined data to be passed to the open exit dynamic link library (DLL) specified by the user, when the file is opened. The value defined for **user_data** is limited to a maximum of 511characters and cannot contain a closing parenthesis ‘)’ character.

Note: The file open exit DLL is defined in the Connect:Direct initialization parameters.

strip.blanks(yes | no) determines whether trailing blank characters at the end of each record are removed from a line of text before it is written to the Windows text file.

Note: The **strip.blanks** parameter is ignored when **datatype(binary)** is specified.

yes removes trailing blank characters. The default is **yes** for OS/390, VM, VSE, and AS/400 nodes.

no does not remove trailing blank characters. The default is **no** for Windows and all other nodes with the exception of OS/390, VM, VSE, and AS/400.

strip.oneable(yes | no) strips ^Z from the end of old DOS files. If this character is not stripped from old DOS files, Connect:Direct Windows translates ^Z into 3F when files are sent to the snode.

yes removes ^Z from the end of old DOS files.

no does not remove ^Z from the end of old DOS files. The default is **no.**

Note: Because this parameter applies to text file transfers only, be sure to specify **datatype (text)** and **strip.oneable (yes)** in **sysopts** to strip the character from the end of the DOS file.

xlate(no | yes) indicates whether character translation should be performed using the default or user-supplied translation table. Typically, this translation is between ASCII and EBCDIC.

no specifies the translation should not be performed. The default for Windows and all other nodes with the exception of OS/390, VM, VSE, and AS/400 is **no**.

yes specifies the translation should be performed. The default for OS/390, VM, VSE, and AS/400 nodes is **yes**.

xlate.tbl(*pathname/filename*) specifies that a different translation table from the default table used by Connect:Direct is used. ***pathname/filename*** should identify a translation table created using the translation table editor of the Connect:Direct Requester for Windows. The path specification is optional. If the path is not provided, the path specified by the **xlate.dir** initialization parameter is assumed for the filename specified.

The **sysopts** subparameters valid only in the **to** clause are:

acl(*operation,rightslist,accountname*
[;*operation,rightslist,accountname*[;*operation,rightslist,accountname*[;...]]) specifies the users or groups allowed or denied access to a file being created by the **copy** operation. This subparameter is ignored if specified for a file that already exists.

operation specifies whether the rights are allowed or denied. Valid values are:

- allow
- deny

rightslist specifies the type of rights allowed or denied. Valid values are:

- read
- write
- execute
- delete
- all

The rights values can be combined by linking them with the + sign.

accountname specifies the name of the user or group to which access is allowed or denied. Multiple-user rights may be specified with each group enclosed in parentheses and delimited by a comma.

attributes(*physattr*) specifies the physical attributes of the file when it is created.

physattr can include the following attributes (case sensitive):

- **A** (Archive Needed)
- **H** (Hidden)
- **R** (Read Only)
- **S** (System)

Example

In the following example, the Process copies from a Windows node to an OS/390 node, specifying that the data type is text and blanks are to be left in the file. The comments document the node. The file name on the from side uses the UNC format.

```
copyseq     process     snode=OS390.v4200 /*$OS390$*/
                        hold=no
                        class=1
                        prty=10
                        execprty=10
                        retain=no
                        pnodeid=(user01,pw01)
                        snodeid=(user01,pw01)
step01      copy  from  (
                        file=\\srv-one\c_drive\temp\ntfile.txt
                        pnode/*$WindowsNT$*/
                        sysopts="datatype(text) strip.blanks(no)
                        xlate(yes)")
                    ckpt=1k
                    compress extended
                    to    (
                          snode /*$OS390$*/
                          file=user01.newfile2
                          disp=(rpl,catlg))
```

Note: If you specify the pnode parameter as pnode/*$Windows$*/, the **copy** statement in the previous example copies from a Windows node to an OS/390 node.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Wildcard Copy Examples

Connect:Direct Windows supports the string (*) and character (?) wildcards, allowing you to copy multiple files from a source directory to a target directory with a single copy statement.

Note: The list of files to be copied is generated at the start of a wildcard copy. When a Process restarts, in a wildcard copy step, the step restarts with the first file that is not completely copied. If you are using checkpointing, the step restarts at the past checkpoint of the file that is not completely copied.

You can use a Connect:Direct Windows version 3.3.02 node to perform a wildcard copy send to any other Connect:Direct platform. With a Connect:Direct Windows version 3.3.02 node, you can only receive a wildcard copy from a Connect:Direct UNIX version 3.4.00 or higher node or from another Connect:Direct Windows version 3.3.02 or higher node.

Example Windows to Windows Wildcard Copy

The following wildcard copy step copies any .dat file beginning with Customer from the Connect:Direct Windows PNODE directory I:/Financial/Accounts to a Connect:Direct Windows SNODE directory J:/Financial/Accounts. The source and destination file names are identical.

```
WILDCOPY COPY
      FROM ( FILE=I:\Financial\Accounts\Customer?.dat )
      TO (FILE=J:\financial\accounts\
            DISP=RPL )
```

You must include the ending backslash (\) for the destination directory.

The following wildcard copy step copies all .dat files from the Connect:Direct Windows PNODE directory I:/Financial/Accounts to a Connect:Direct Windows SNODE directory J:/Financial/Accounts. The source and destination file names are identical.

```
WILDCOPY COPY
      FROM (FILE=I:\Financial\Accounts\*.dat )
      TO ( FILE=J:\financial\accounts\
            DISP=RPL )
```

Example Windows to UNIX Wildcard Copy

To copy files to a Connect:Direct UNIX node, you must include an ending forward slash (/) in the **TO FILE**= parameter, as shown in the following:

```
WILDCOPY COPY
      FROM ( FILE="I:\Financial\Accounts\Customer?.dat" )
      TO (FILE=/financial/accounts/
            DISP=RPL )
```

Example Windows to OS/390 Wildcard Copy

To copy files to sequential files on a Connect:Direct OS/390 node, you must include an ending period (.) in the **TO FILE**= parameter, as shown in the following:

```
WILDCOPY COPY
      FROM ( FILE=/financial/accounts/* )
      TO (   FILE=FINANCIAL.ACCOUNTS.
            DISP=RPL )
```

To copy files to a PDS on a Connect:Direct OS/390 node, you must use an asterisk (*) for the PDS member name, as shown in the following:

```
WILDCOPY COPY
      FROM ( FILE=I:\Financial\Records\* )
      TO (FILE=FINANCIAL.RECORDS(*)
            DISP=RPL )
```

Example Wildcard Copy to a Connect:Direct Node with Download Restrictions

To copy to a Connect:Direct node that enforces download restrictions, use an asterisk (*) for the **TO FILE**= parameter of the destination download directory. Following is an example.

```
WILDCOPY COPY
      FROM (FILE=I:\Financial\Records\* )
      TO (FILE=*
          DISP=RPL )
```

Chapter 57

Connect:Direct Windows Run Job Statement

The Connect:Direct Windows **run job** statement allows you to start programs that will run on the same Windows system as Connect:Direct. The command or program you specify in the **run job** statement executes on the system where Connect:Direct is located. The program may be any of the following:

- Windows XP or 2000 program
- Windows 95/NT program
- Windows 95/NT command file
- Windows 3.1 program in a Windows-on-Windows 95/NT VDM

The new program runs as a separate Windows process. The Connect:Direct Process continues execution immediately after the new Windows process is started.

Note: Process statements that follow the **run job** statement are executed without waiting for the submitted job to complete. Connect:Direct does not verify the validity of the submitted job statements or commands. The Connect:Direct return code indicates the success or failure of the **run job** statement, not the success or failure of the submitted job.

Format

The Connect:Direct Windows **run job** statement format includes the following parameters. Required parameters are shown in bold print. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**run job**	pnode | snode
		dsn=WINxx (either WIN95 or WINNT)
		sysopts="pgm(*filename*) **cmd(*command* | *parms*)** **args(*arguments*)** **desktop(*yes*|*no)*"**

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in position one. The label consists of a 1-8 character alphanumeric string.

stepname specifies the optional step name to be associated with this **run job** statement.

Note: Statement names, parameter names, and subparameter names are reserved and cannot be used as labels.

The label may be referenced by conditional statements to change the flow of instructions within a Process. A label is not required on the **run job** statement.

Statement

run job identifies the statement with all its parameters as the **run job** statement.

Required Parameters

dsn=WINxx

specifies that the job as either a WIN95 or WINNT program or command.

sysopts="pgm(*filespec*) cmd(*command* | *parms*) args(*arguments*) | desktop(yes|no)"

specifies the parameters appropriate for the operation being performed. Possible values are:

pgm(*filespec*) specifies which .EXE or .BAT file will be run.

Note: You must use either the **pgm** or **cmd** parameter. Do not specify both.

cmd(*command* | *parms*) specifies a system command and any arguments that this command requires.

args(*arguments*) specifies the arguments passed to the program when it is started. These arguments are in the same format as they would be specified from the command prompt. This optional parameter is only valid when you specify **pgm**.

desktop(*yes* | *no*) specifies whether you want to interact with the desktop. The default for the **desktop** parameter is **no** when used with the **run job** statement. This optional parameter is only valid when specified with **pgm** or **cmd**.

Optional Parameters

pnode | snode

specifies the Connect:Direct node where the **run job** statement executes. The default is **pnode**.

Example

In this example, the **run job** statement executes the program testwin.exe on a remote Windows system.

```
jobstep   run job   snode (dsn=WINNT)
                    sysopts="pgm(c:\winnt35\system32\testwin.exe)"
```

Note: If you change the dsn parameter to dsn=WIN95, the **run job** statement in the previous example executes testwin.exe on a remote Windows 95 system.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 58

Connect:Direct Windows Run Task Statement

The Connect:Direct Windows **run task** statement allows you to start programs that run on the same Windows system as Connect:Direct. The command or program you specify in the **run task** statement executes on the system where Connect:Direct is located. The program may be any of the following:

- Windows XP or 2000 program
- Windows 95/NT program
- Windows 95/NT command file
- Windows 3.1 program in a Windows-on-Windows 95/NT VDM
- MS-DOS application in a DOS subsystem

The new program runs as a separate Windows process. The Connect:Direct Process waits for the completion of the command or program specified before continuing.

Caution: Do not specify programs in the **run task** statement that cannot complete without user intervention. Unless you explicitly allow Connect:Direct to interact with the desktop through the Control Panel's Services applet, you will not have the opportunity to provide input to the program and it will not complete execution.

Process statements that follow the **run task** statement do not execute until the specified command or program completes. The execution of the **run task** statement results in a return code which is the exit code for the program executed using **run task**. See the ***nn*** parameter on page 441 for a list of standard return codes.

Caution: When using an external program executed by the Connect:Direct **run task** statement, do not use a return code of **16** in the external program or the Connect:Direct Process will fail.

Format

The Connect:Direct Windows **run task** statement format includes the following parameters. Required parameters are shown in bold. Default values for parameters are underlined.

Label	Statement	Parameters
stepname	**run task**	pnode | snode
		pgm=WINxx (WINxx represents either WIN95 or WINNT)
		sysopts="pgm(*filename*) **cmd(*command*|*parms*)** **args(*arguments*)** **desktop(*yes*|*no*)"**
		restart = y|n

Label
Connect:Direct statements are identified by user-defined labels (***stepname***). A label is any character or character string beginning in position one. The label consists of a 1-8 character alphanumeric string. Statement names, parameter names, and subparameter names cannot be used as a label.

Statement
run task identifies the statement with all its parameters as the **run task** statement.

Required Parameters

pgm=WINxx
specifies that the task is either a WIN95 or WINNT program or command.

sysopts="pgm(*filename*) cmd(*command*|*parms*) args(*arguments*)|desktop(*yes*|*no)*"
specifies the parameters appropriate for the operation being performed. Possible values are:

pgm(*filename*) specifies which .EXE or .BAT file will be run.

Note: You must specify either the **pgm** or **cmd** subparameter. Do not specify both.

cmd(*command* | *parms*) specifies a system command and any arguments the command requires.

args(*arguments*) specifies the arguments passed to the program when it is started. These arguments are in the same format as they would be specified from the command prompt. This optional parameter is only valid when you specify **pgm**.

desktop(*yes* | *no*) specifies whether you want to interact with the desktop. The default for the **desktop** parameter is **no** when used with the **run task** statement. The optional parameter is only valid when specified with **pgm** or **cmd**.

Optional Parameters

pnode | snode
specifies the Connect:Direct node where the **run task** statement executes. The default is **pnode**.

restart = y|n
specifies whether a program runs again after a session failure. This value overrides the value set in the intialization parameter.

Example

The Process in the following example starts testwin.exe and waits for it to finish::

```
jobstep   run task   snode (pgm=WINNT
                     sysopts="pgm(C:\winnt35\system32\testwin.exe)"
```

Note: If you change the dsn parameter to pgm=WIN95, the **run task** statement in the previous example executes testwin.exe on a remote Windows 95 system.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 59

Connect:Direct Windows Submit Statement

During execution of a Connect:Direct Process, the **submit** statement causes another Connect:Direct Process to be submitted to either the pnode, the node with Process control, or the snode, the node that participates in the Process.

The Process submitted must reside in a file on the node where the **submit** statement executes; this node is referred to as the **subnode**.

Note: The **submit** statement described in this chapter is not the same as the **submit** command. The **submit** statement is used within a Process to submit another Process. See the *Connect:Direct Applications Programming Interface for Windows User's Guide* for **submit** command syntax and parameters.

Any parameter values specified in the **submit** statement override the parameter values contained in a **process** statement named by the **file** parameter.

The execution of the **submit** statement produces a return code. See the **nn** parameter definition on page 441 for a list of standard return codes.

Note: The return code indicates the success of the **submit** statement, not the success of the Process started.

Format

The **submit** statement format includes the following parameters. Required parameters are shown in bold print. Default values for parameters are underlined.

Label	Statement	Parameters
label	**submit**	**file=*filename***
		class=*n*
		execprty=*nn*
		hold=yes |<u>no</u> | call
		localacct=*"pnode node accounting data"*

Label	Statement	Parameters
		pnodeid=(id[,pswd])
		newname=*new process name*
		notify=*username*
		prty=*nn*
		snode=*nodename*
		remoteacct=*"snode accounting data"*
		snodeid=(*id* [,*pswd*[,*newpswd*]])
		retain=yes | no | initial
		startt=([*date* | *day*][,*time* [am | pm]])
		subnode=pnode | snode
		&symbolic name1=variable string 1 *&symbolic name2=variable string 2* . . . *&symbolic namen=variable string n*

Field Descriptions

Label

Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in position one.

> **Note:** Statement names, parameter names, and subparameter names are reserved words and cannot be used as labels.

The label, in conjunction with other commands, may be used to change the flow of instructions within a Process. A label is not required on the **submit** statement.

Statement

submit identifies the statement with all its parameters as the **submit** statement.

Required Parameters

file = *filename*
specifies the 1-256 character file name of the file containing the Process. The **file** parameter can include a pathname indicating the location of the Process. If the **file** parameter does not include a pathname indicating the location of the Process, the directory specified in the **process.dir** initialization parameter is searched.

Note: If you specify the **file** parameter, it must be specified before any other parameter. If you do not specify the **file** parameter, the text of the Process must follow the **submit** command.

Enclose the file name in double quotation marks when using a reserved word (statement name, parameter name, or subparameter name) for the file name.

Optional Parameters

class = *n*
determines the node-to-node session on which a Process can execute. A Process may execute in the class specified or any higher session class up to the maximum allowed for the snode (**sess.remote.max**). The default class is specified as the **sess.default** parameter of the **local.node** record in the initialization parameters file. The **class** default is **1**.

execprty = *nn*
specifies the relative operating system priority assigned to the Process. The **execprty** keyword determines how Windows schedules the Session Manager, which runs the Process, for execution relative to other tasks in the system. A Process with a higher priority receives more opportunities for execution from the operating system than a Process with a lower priority. Values range from 1-15. If **execprty** is not specified, the default is **7**.

Note: Scheduling Processes to run with a high priority may have an adverse effect on the execution of other applications in the system.

---

hold = yes | no | call
specifies how the Process is handled in the Hold queue.

yes specifies that the Process is held in the Hold queue in HI status until it is explicitly released by a **change process** command. When both **hold=yes** and a **startt** value are specified, the **hold** specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

no specifies that the Process is not held in the Hold queue. It is executed as soon as resources are available. The default is **no**.

call specifies that the Process is held until the snode connects to the pnode. At that time, the Process is released for execution. The Process is also released when another Process on the pnode connects to the snode.

Note: When a Process is submitted with **retain=yes** and **hold=no** or **hold=call,** the **hold** parameter is ignored.

---

localacct = "*pnode node accounting data*"
specifies an arbitrary string used as accounting information for the pnode. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

pnodeid = (*id* [,*pswd*])
specifies security userids and passwords at the pnode. The **pnodeid** parameter value can be contain 1-20 alphanumeric characters.

When copying to a file server other than the one containing Connect:Direct, **pnodeid** is required.

id specifies the userid to be used on the pnode.

pswd specifies a user password on the pnode.

If ***pswd*** is specified, ***id*** must be specified also. These values must be coded in the order of ***id*** and ***pswd***.

newname = *new process name*
specifies the new name to be given to the Process. This value overrides the label on the **process** statement. The maximum length for the **newname** parameter is eight characters. The default is the label on the **process** statement.

notify = *username*
specifies the user name to receive Process completion messages.

prty = *nn*
specifies the selection priority of the Process in the Transmission Control Queue (TCQ). The **prty** keyword influences the order of Process selection. A Process with a higher priority is selected for execution before a Process with a lower priority. The value specified for the **prty** parameter does not affect the priority during execution. Values range from 0-15. The highest priority is 15. If **prty** is not specified, the default is 10.

snode = *nodename*
is an alphanumeric character string that specifies the snode name. This parameter overrides the value specified in the **process** statement. The snode default value is the value coded in the **process** statement. This parameter is required either on the **submit** command or **process** statement.

nodename is the node name of the Connect:Direct snode.

remoteacct = "*snode accounting data*"
specifies accounting data for the snode. The maximum length of the string is 256 characters. The string must be enclosed in double quotation marks.

The **remoteacct** parameter overrides any accounting data specified on the **process** statement of the submitted Process.

snodeid = (*id* [,*pswd*[,*newpswd*]]
specifies security userids and security passwords for the Process on the snode. Each **snodeid** parameter can contain 1-20 alphanumeric characters. When copying to a file server other than the one Connect:Direct is on, **snodeid** is required.

id specifies the security ID that is passed to a security exit.

pswd specifies the user password on the snode.

newpswd is the new password value. The user password is changed to the new value on the snode if the userid and old password are correct and the snode supports this optional parameter. The ***newpswd*** parameter is not valid if the snode is a Connect:Direct Windows node.

If ***pswd*** is specified, ***id*** must be specified also. If ***newpswd*** is specified, ***pswd*** must be specified also. These values must be coded in the order of ***id***, ***pswd***, and ***newpswd***.

***For Connect:Direct Tandem nodes*:** The **snodeid** parameter specifies the Tandem group number and user number. The valid range for these numbers is 0-255.

retain = yes | no | initial

specifies whether Connect:Direct retains a copy of the Process in the TCQ for re-execution after execution has completed.

yes specifies that the Process is retained in the Hold queue in HR status after execution. The Process must be released manually through the **change process** command to cause it to be executed or explicitly deleted through the **delete process** command.

Processes submitted with both **retain=yes** and **startt** are cloned, and the clone runs immediately. If **startt** specifies both day of the week and a time, the job will be cloned and will run every week on that day and at that time. If **startt** specifies a day only, the job is cloned and runs every week on that day at 12:00 a.m. If **startt** specifies a time only, the job will be cloned and run every day at that time.

no specifies that the Process is not retained. The default is **no**.

Processes submitted with **retain=yes** but without **startt** are submitted the first time. Two copies of the job are submitted. One is submitted with **retain=yes** and one with **retain=no**. The **retain=no** job runs immediately (or when its **startt** expires); the **retain=yes** job remains in the queue unless explicitly released. When it is released, it is again cloned, and the clone runs immediately.

initial specifies that the Process is retained in the Hold queue in HR status and is automatically executed each time the Process Manager initializes.

Processes submitted with **retain=initial** are only searched for when the server initially starts. At that time, they are cloned, and the clone executes immediately.

Caution: The **startt** parameter should not be coded when **retain=initial** is coded. This causes the **submit** statement to fail.

Note: When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

---

startt = ([*date* | *day*] [,*time* [am | pm]])

specifies that the Process will be executed at a specific date and/or time. The Process is placed in the Timer queue in WS (Wait for Start Time) status. The ***date***, ***day***, and ***time*** are positional subparameters. If ***date*** or ***day*** is not specified, a comma must precede ***time***.

date specifies that the Process will execute on a specific date. The ***date*** value may be specified in the format ***mm/dd/yyyy*** or ***mm-dd-yyyy***, where:

- ***mm*** indicates month and can be a one- or two-digit number
- ***dd*** indicates day and can be a one- or two-digit number
- ***yyyy*** indicates year and can be a two- or four-digit number

If only ***date*** is specified, ***time*** defaults to 00:00:00. The current date is the default.

day specifies the day of the week a Process is to be released for execution. Values are **today**, **tomorrow**, **monday**, **tuesday**, **wednesday**, **thursday**, **friday**, **saturday**, and **sunday**.

time specifies the time the Process will start. The format is ***hh:mm:ss*** **[am|pm]** where (***hh***) specifies the hour, (***mm***) the minute, and (***ss***) the second the Process will start. Hour may be specified in either 12- or 24-hour format. If the 12-hour format is used, then **am** or **pm** must be specified. Seconds (***ss***) are optional. A space is required before **am** or **pm**. The default is 24-hour format. The default value is 00:00:00, which indicates midnight.

If only ***day*** is specified, time defaults to 00:00:00. For example, if a Process is submitted on Monday, with **monday** as the only **startt** parameter, the Process does not run until the following Monday at midnight.

Note: When both **hold=yes** and a **startt** value are specified, the hold specification takes precedence. Therefore, a Process submitted with **hold=yes** is placed in the Hold queue even if a start time is specified.

The **startt** parameter must not be coded with **retain=initial**. This causes the **submit** command to fail.

---

subnode = pnode | snode

specifies whether the Process should be submitted to the **pnode**, the node where the Process currently executing was submitted, or to the **snode**, the current executing session partner of the Process.

pnode specifies that the Process is submitted on the node that has Process control. The default value is **pnode**.

snode specifies that the Process is submitted on the node participating in, but not controlling, Process execution.

&symbolic name1 = variable string 1
&symbolic name2 = variable string 2
.
.
.
&symbolic namen = variable string n

specifies the value for a symbolic parameter in the Process. This default can be overridden when submitting the Process.

Note: Statement names, parameter names, and subparameter names are reserved and cannot be used as symbol names.

The value for the symbolic parameter must be in double quotation marks if it is a statement name, parameter name, subparameter name or if it contains special characters.

The symbolic parameter can be set to a single ampersand symbolic parameter that was resolved during the first Process submission. Do not use identical symbolic names.

Special Characters

Variable names begin with an ampersand and are terminated by delimiters. Variable names and their values cannot exceed 126 characters. The following table lists the delimiters recognized by Connect:Direct.

Delimiters	Description
	blank
	tab
	carriage return
	line feed
<	less than sign
>	greater than sign
*	asterisk
(	open parenthesis
)	closed parenthesis
/	slash
\	backslash
:	colon
;	semicolon
,	comma
.	period
'	single quotation mark
"	double quotation mark
=	equal sign
{	opening brace
}	closing brace
[	opening bracket
]	closing bracket
&	ampersand

Single or Double Quotation Marks

The rules for using single or double quotation marks are as follows:

- Substitution occurs only if the string is enclosed in double quotes. If you are substituting into a string that is already in double quotes, such as the **sysopts** value, no additional double quotes are needed. Double quotes added for substitution purposes, but not needed, are stripped off.
- In substitution variable definitions, such as the **process** statement or **submit** command, enclose the value in double quotes if it contains any special characters. Substitution values that do not contain special characters do not need to be enclosed in double quotes.
- If the string is enclosed in single quotes, no substitution is done.

The following example shows the use of quotation marks to allow special characters or blanks to be embedded within a symbolic parameter or subparameter value.

```
&NAME="C:\MYFILE.TXT"
FILE="&NAME"
```

The previous example converts to the following string:

```
FILE=C:\MYFILE.TXT
```

Concatenation

The rules for using periods are as follows:

- Single periods after variable names can be used to concatenate variables to constants and are dropped when the variable is resolved to its value.
- If two or more periods are found after variable names, the variable is resolved, one period is dropped, and the rest are retained.

The following example shows the use of periods to concatenate variables to constants:

```
&NAME=MYFILE
FILE="C:\&NAME..TXT"
```

The result is the following string:

```
FILE=C:\MYFILE.TXT
```

Spanning Lines

Strings that span multiple lines and contain substitution variables are not allowed.

Example

In this example, the Process uses symbolics to allow you to specify the file names at the time of submission. Process accounting data is specified for the pnode and snode.

```
copyseq     process     snode=OS390.dallas
                        localacct="dept-59"
                        remoteacct="dept-62"
step01      copy  from  (file="&homefile")
                  to    (file="&dfwfile"
                        snode)
```

The following submit command specifies the file names to be used in a file transfer using the previous Process.

```
  submit  file=copyseq
          &homefile="c:\mydir\myfile.txt"
          $dfwfile="user01.newfile."
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 60

Connect:Direct Windows Conditional Statements

The conditional statements (**if then**, **else**, **eif**, **goto**, and **exit**) permit you to alter the sequence of Process step execution based on the completion of one or more previous steps in the Process.

Format

The Connect:Direct Windows conditional statements include the following. The required parameters and keywords are in bold print.

Label	Statement	Parameters
optional	if	**(*label condition nn*)** then
		(*process steps executed if condition true*)
	else	
		(*process steps executed if condition false*)
	eif	
optional	goto	label
	exit	

Statement Descriptions

if then
specifies that the Connect:Direct system executes a block of Connect:Direct statements based on the completion code of a Process step. An **eif** statement must be used in conjunction with an **if/then** statement. See the following *Field Descriptions* section for details.

else
designates a block of Connect:Direct statements that execute when the **if/then** condition is not satisfied. No parameters exist.

eif
is required for specifying the end of the **if then** or **if then else** block of statements. No parameters exist.

goto
moves to a specific step within a Process. See the following *Field Descriptions* section for details.

exit
is used to bypass all remaining steps within a Process. No parameters exist.

Field Descriptions

Label
Connect:Direct statements are identified by user-defined labels. A label is any character or character string beginning in position one. The label in conditional statements is optional.

The label may be referenced to change the flow of instructions within a Process. A label is not required on the **conditional** statements.

Statement
Identify the statement with a **conditional** statement.

Required Parameters

The following parameters are required in an **if** conditional statement.

label
The ***label*** parameter specifies the step to which the condition applies.

condition

The ***condition*** parameter specifies the type of comparison that will be performed. This condition checking can be based on such comparisons as equal to, greater than, or less than. The valid conditions are as follows:

== |=| eq specifies that the return code must be equal to the value ***nn*** for the condition to be satisfied.

<>|!=|ne specifies that the return code must not equal the value ***nn*** for the condition to be satisfied.

>=|=>|ge specifies that the return code must be greater than or equal to the value ***nn*** for the condition to be satisfied.

>|gt specifies that the return code must be greater than the value ***nn*** for the condition to be satisfied.

<=|=<|le specifies that the return code must be less than or equal to the value ***nn*** for the condition to be satisfied.

<|lt specifies that the return code must be less than the value ***nn*** for the condition to be satisfied.

nn

This parameter specifies the numeric value used for return code checking of the step named in the ***label***. If coded as **x'*nn*'**, it is a hexadecimal value. Any other coding indicates it is decimal. Standard return codes are as follows:

- 0 indicates successful completion.
- 4 indicates warning.
- 8 indicates error.
- 16 indicates catastrophic error.

Example

In this example, the Connect:Direct Windows Process contains all of the conditional statements.

```
copy01     process     snode=cdchicago
step01     copy  from  (file=myfile1 pnode)
                 to    (file=yourfile1 snode)
step02     if          (step01 gt 4) then
                       goto step07
           else
step03     runjob      dsn=WINxx
                       sysopts="pgm(testwin.exe)"
           eif
step04     if          (step03 >= 8) then
           exit
           eif
step05     if          (step03 lt 4) then
step06     copy  from  (file=myfile2 pnode)
                 to    (file=yourfile2 snode)
           eif
           exit
step07     run task     pgm=WINxx
                        sysopts="pgm(failjob.exe)"
           exit
```

Note: In the previous example, you must specify the dsn and pgm parameters as either WIN95 or WINNT.

copy01 is the **process** statement defining the Process name as copy01 and the snode as cdchicago.

step01 copies file myfile1 on the pnode to file yourfile1 on the snode.

step02 checks the completion code of step01. If step01 fails (return code greater than 4), step07 executes. If step01 completes with a return code of **4** or less, step03 executes.

step03 starts the program testwin.exe.

step04 checks the completion code of step03. If step03, not the program testwin.exe, fails with a code of **8** or greater, the Process terminates. Otherwise, step05 executes.

step05 checks the completion code from step03. If less than **4**, indicating the step completed without error, not the program testwin.exe, step06 executes.

step06 copies file myfile2 on the pnode to file yourfile2 on the snode. The Process will then exit.

step07 only executes if step01 fails. The **run task** step executes the program goodjob.exe.

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Chapter 61

Connect:Direct Windows Pend Statement

The **pend** statement marks the end of the Connect:Direct Windows Process. No parameters are associated with the **pend** statement.

Example

The following Process copies a file from a pnode to a node named denver. The **pend** statement indicates the end of the Process.

```
process2     process     snode=denver
copystep     copy  from  (dsn="&homedsn")
                   to    (dsn=newfile)
                         disp=(new)
             pend
```

See the *Connect:Direct Process Concepts and Examples Guide* for additional examples.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X