

IBM Sterling Connect:Direct Process
Language
6.2

Documentation



This edition applies to Version 6 Release 2 of IBM® Connect:Direct and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 1993, 2021.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Understanding Sterling Connect:Direct Processes.....	1
What is a IBM Connect:Direct Process?.....	1
Process Language.....	2
Process Components.....	3
Concept of Nodes.....	4
How Do Processes Work?.....	5
How to Build Processes.....	7
Building Sterling Connect:Direct Processes With the Process Builder.....	8
Build IBM Connect:Direct Processes With the Process Builder.....	8
Signing on to the Connect:Direct Browser User Interface.....	10
Creating a Process Statement.....	10
Creating a Copy Statement.....	11
Creating a Run Task Statement.....	13
Creating a Run Job Statement.....	13
Creating Conditional Statements.....	14
Creating a Submit Statement.....	14
Validating Process Syntax.....	15
Saving and Submitting a Process.....	15
Viewing a Process in Text Format.....	15
Editing a Process.....	15
Creating Processes with a Text Editor.....	16
Create Processes with a Text Editor.....	16
Creating a Process That Executes at the Same Time Every Day.....	16
Creating a Process That Copies a File.....	17
Creating Processes with Sterling Connect:Direct Requester.....	19
Create Processes with IBM Connect:Direct Requester.....	19
Creating a Process Statement.....	20
Adding a Copy Statement.....	21
Adding Conditional Statements.....	22
Adding a Run Task Statement.....	22
Validating Process Content.....	22
Saving a Process.....	23
Editing a Process.....	23
Process Language Syntax.....	23
Process Language Syntax.....	23
Punctuation.....	23
Comments.....	24
Concatenation.....	25
HFS File Name Considerations.....	26
Special Character Strings.....	27
Symbolic Substitution.....	29
Termination.....	31
SYSOPTS Syntax.....	31
Chapter 2. Sterling Connect:Direct Process Statements and Parameters.....	35
HP Non-Stop Statements and Parameters.....	35
Connect:Direct for HP NonStop Process Statement.....	35
Connect:Direct for HP NonStop Copy Statement for Spooler Jobs.....	35
Connect:Direct for HP NonStop Copy Statement for Guardian Disk and Tape Files.....	36
Connect:Direct for HP NonStop Copy Statement for OSS Disk and Tape Files.....	39

Connect:Direct for HP NonStop Run Job Statement.....	41
Connect:Direct for HP NonStop Run Task Statement.....	41
Connect:Direct for HP NonStop Submit Statement.....	41
Connect:Direct for HP NonStop Conditional Statements.....	42
Connect:Direct for HP NonStop Symbol Statement.....	42
Connect:Direct for HP NonStop Process Parameters.....	42
i5/OS Statements and Parameters.....	59
Connect:Direct for i5/OS Process Statements.....	59
Connect:Direct for i5/OS Copy Files Statement.....	59
Connect:Direct for i5/OS Copy Members Statement.....	61
Connect:Direct for i5/OS Copy Objects Statement.....	63
Connect:Direct for i5/OS Copy Spooled Files Statement.....	64
Connect:Direct for i5/OS File Support.....	65
Member Name Length in Connect:Direct for i5/OS Copy to z/OS.....	65
Connect:Direct for i5/OS Run Job Statement.....	66
Connect:Direct for i5/OS Run Task Statement.....	66
Connect:Direct for i5/OS Process Parameters.....	66
Microsoft Windows Statements and Parameters.....	85
Connect:Direct for Microsoft Windows Process Statement.....	85
Connect:Direct for Microsoft Windows Copy Statement.....	86
Connect:Direct for Microsoft Windows Run Job Statement.....	87
Connect:Direct for Microsoft Windows Run Task Statement.....	88
Connect:Direct for Microsoft Windows Submit Statement.....	88
Connect:Direct for Microsoft Windows Conditional Statements.....	89
Connect:Direct for Microsoft Windows Pend Statement.....	90
Connect:Direct for Microsoft Windows Process Parameters.....	90
OpenVMS Statements and Parameters.....	101
Connect:Direct for OpenVMS Process Statement.....	101
Connect:Direct for OpenVMS Copy Statement.....	101
Connect:Direct for OpenVMS Run Job Statement.....	103
Connect:Direct for OpenVMS Run Task Statement.....	104
Connect:Direct for OpenVMS Submit Statement.....	104
Connect:Direct for OpenVMS Conditional Statements.....	105
Connect:Direct for OpenVMS Symbol Statement.....	105
Connect:Direct for OpenVMS Process Parameters.....	105
UNIX Statements and Parameters.....	117
Connect:Direct for UNIX Process Statement.....	117
Connect:Direct for UNIX Copy Statement.....	118
Connect:Direct for UNIX Run Job Statement.....	120
Connect:Direct for UNIX Run Task Statement.....	120
Connect:Direct for UNIX Submit Statement.....	120
Connect:Direct for UNIX Conditional Statements.....	121
Connect:Direct for UNIX Pend Statement.....	122
IBM Connect:Direct for UNIX Process Parameters.....	122
VM/ESA Statements and Parameters.....	132
Connect:Direct for VM/ESA Process Statement.....	132
Connect:Direct for VM/ESA Copy Statement.....	133
Connect:Direct for VM/ESA Run Job Statement.....	135
Connect:Direct for VM/ESA Run Task Statement.....	135
Connect:Direct for VM/ESA Submit Statement.....	136
Connect:Direct for VM/ESA Conditional Statements.....	137
Connect:Direct for VM/ESA Symbol Statement.....	137
Connect:Direct for VM/ESA Process Parameters.....	137
VSE/ESA Statements and Parameters.....	151
Connect:Direct for VSE Process Statement.....	151
Connect:Direct for VSE Copy Statement.....	152
Connect:Direct for VSE Run Job Statement.....	157
Connect:Direct for VSE Run Task Statement.....	157

Connect:Direct for VSE Submit Statement.....	157
Connect:Direct for VSE/ESA Conditional Statements.....	158
Connect:Direct for VSE Symbol Statement.....	159
Connect:Direct for VSE/ESA Process Parameters.....	159
z/OS Statements and Parameters.....	183
Connect:Direct for z/OS Process Statement.....	183
Connect:Direct for z/OS Copy Statement.....	184
PDS Support.....	188
VSAM Support.....	189
SMS Support.....	189
GDG Support.....	191
MBCS Support.....	191
Connect:Direct for z/OS Run Job Statement.....	192
Connect:Direct for z/OS Run Task Statement.....	192
Connect:Direct for z/OS Submit Statement.....	192
Connect:Direct for z/OS Conditional Statements.....	193
Connect:Direct for z/OS Symbol Statement.....	194
Creating Processes from z/OS Statement Models.....	194
Connect:Direct for z/OS Process Parameters.....	195

Chapter 3. Sterling B2B Integrator - Connect:Direct Server Interoperability..... 227

Sterling B2B Integrator-Connect:Direct Server Adapter Syntax.....	227
Transmit Files between IBM Connect:Direct and B2B Integrator.....	227
IBM Connect:Direct Parameters for Communicating with B2B Integrator.....	231
Sterling B2B Integrator-Connect:Direct Server Adapter Examples.....	232
B2B Integrator Connect:Direct Server Adapter Examples.....	232
Copy a File from Connect:Direct for z/OS to a B2B Integrator Mailbox.....	232
Copy a File from Connect:Direct for UNIX to a B2B Integrator Mailbox.....	232
Copy a File from Connect:Direct for Microsoft Windows to a B2B Integrator Mailbox.....	232
Copy a File from Connect:Direct for HP NonStop to a B2B Integrator Mailbox.....	233
Copy a File from Connect:Direct for i5/OS to a B2B Integrator Mailbox.....	233
Copy a File from Connect:Direct for z/OS to a B2B Integrator Business Process.....	233
Copy a File from Connect:Direct for UNIX to a B2B Integrator Business Process.....	234
Copy a File from Connect:Direct for Microsoft Windows to a B2B Integrator Business Process....	234
Copy a File from Connect:Direct for HP NonStop to a B2B Integrator Business Process.....	234
Copy a File from Connect:Direct for i5/OS to a B2B Integrator Business Process.....	235
Retrieve a File from a B2B Integrator Business Process to Connect:Direct for z/OS.....	235
Retrieve a File from a B2B Integrator Business Process to Connect:Direct for Microsoft Windows.....	235
Retrieve a File from a B2B Integrator Business Process to Connect:Direct for UNIX.....	235
Retrieve a File from a B2B Integrator Business Process to Connect:Direct for HP NonStop.....	236
Retrieve a File from a B2B Integrator Business Process to Connect:Direct for i5/OS.....	236
Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for z/OS.....	236
Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for Microsoft Windows.....	237
Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for UNIX.....	237
Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for HP NonStop.....	237
Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for i5/OS.....	238
Submit a B2B Integrator Business Process from Connect:Direct for z/OS.....	238
Submit a B2B Integrator Business Process from Connect:Direct for UNIX.....	238
Submit a B2B Integrator Business Process from Connect:Direct for Microsoft Windows.....	238
Submit a B2B Integrator Business Process from Connect:Direct for HP NonStop.....	239
Submit a B2B Integrator Business Process from Connect:Direct for i5/OS.....	239

Chapter 4. Example Processes..... 241

PROCESS Statement Examples.....	241
Detailed PROCESS Statement Using the RETAIN Parameter.....	241
Detailed PROCESS Statement Using the NOTIFY Parameter.....	241

Detailed PROCESS Statement Using Long Passphrases.....	241
Use %PNODE.....	242
Specify a List of Ciphers in a Particular Process.....	242
Encrypt Only Control Block Information—No Data Being Sent.....	242
Use %DD2DSN to Pass DSN from JCL to a Process.....	243
Use a TCP/IP Address for the SNODE Keyword (IPv4).....	243
Use a TCP/IP Address for the SNODE Keyword (IPv6).....	243
Use TCPNAME to Identify the PNODE/SNODE Sites.....	243
HP NonStop PROCESS Statement.....	243
Example OpenVMS PROCESS Statement.....	244
Example VMESA Process.....	244
VSE PROCESS Statement.....	244
Use Symbolics in a UNIX Process.....	245
COPY Statement Examples.....	245
HP NonStop.....	246
i5/OS.....	254
Microsoft Windows.....	255
OpenVMS.....	258
UNIX.....	265
VM/ESA.....	271
VSE.....	285
z/OS.....	310
SUBMIT Statement Examples.....	332
Use SUBMIT with Symbolic Substitution (z/OS to z/OS).....	332
Use SUBMIT with the DSN Parameter (z/OS to z/OS).....	333
Submit a Microsoft Windows Process from a UNIX Node.....	334
Use SUBMIT at the Connect:Direct for HP NonStop Node.....	334
Use submit with the hold Parameter (UNIX to UNIX).....	334
Use submit with the startt Parameter (UNIX to UNIX).....	335
Submit a Process to Run Upon Successful Completion of a Copy (HP NonStop to HP NonStop)...	335
Submit a Process on OpenVMS.....	335
Submit a Process That Copies a File from One UNIX Node to Another, Then to a Third UNIX Node.....	335
Submit a Process with Symbolics on VSE.....	336
Submit a Process Using Symbolics on VMESA.....	337
Submit a Process from UNIX to z/OS to Connect:Enterprise Using BATCHID.....	337
RUN JOB Statement Examples.....	340
HP NonStop.....	340
Microsoft Windows.....	341
OpenVMS.....	341
UNIX.....	343
VM/ESA.....	344
VSE.....	344
z/OS.....	344
RUN TASK Statement Examples.....	346
HP NonStop.....	347
i5/OS.....	348
Microsoft Windows.....	350
OpenVMS.....	352
UNIX.....	353
VM/ESA.....	354
VSE.....	356
z/OS.....	357
Conditional Statement Examples.....	365
Use of Conditional Statements (OpenVMS to z/OS).....	365
Use of Conditional Logic (z/OS to VSE).....	366
Use of Conditional Logic (i5/OS to z/OS).....	366
Use of Conditional Logic (HP NonStop to z/OS).....	367

Use of Conditional Statements to Test for Process Completion on OpenVMS.....	368
Use of Conditional Statement on OpenVMS.....	368
Use of Conditional Statements in a UNIX Process.....	369
Use of Conditional Statements in VMESA.....	369
Use of Conditional Statements in a VSE Process.....	370
Use of Conditional Statements in a Microsoft Windows Process.....	371
Use of Conditional Logic to Test Message ID in z/OS.....	371
Use of Conditional Logic to Test Highest Return Code with %MAXRC in z/OS.....	371
Use Conditional Logic to Test Variables in z/OS.....	372
Example UNIX pend Statement.....	372
Notices.....	373
Trademarks.....	374
Terms and conditions for product documentation.....	375

Chapter 1. What is a IBM Connect:Direct Process?

An IBM® Connect:Direct® Process is a series of statements and parameters that perform data movement and manipulation activities such as:

- moving files between different IBM Connect:Direct servers
- running jobs, programs, and commands on the IBM Connect:Direct server
- starting other Processes
- monitoring and controlling Processes
- handling processing errors

Processes can be linked to network and application activities, generating a continuous cycle of processing. For example, a network message can trigger a file transfer that is used by another application.

As a Process executes and after it completes, audit information is available to analyze and use for future processing.

Features that you can specify in a Process include:

- Scheduling—Setting a Process to run at a specific day and time. Processing can be scheduled to run automatically at a specified date or interval, without any operator intervention.
- Integration with native applications—Invoke native applications from within a IBM Connect:Direct Process.
- Integration with existing security systems—Specify user IDs and passwords within a Process that allow it to work within your existing network security system
- Data transmission integrity—Specifying checkpoint and restart intervals within a file transmission, so that if a transmission fails at some point, it can be restarted automatically from the most recent checkpoint.
- User notification—Automatically notify users of successful and unsuccessful transfers.

What is a IBM Connect:Direct Process?

An IBM Connect:Direct Process is a series of statements and parameters that perform data movement and manipulation activities such as:

- moving files between different IBM Connect:Direct servers
- running jobs, programs, and commands on the IBM Connect:Direct server
- starting other Processes
- monitoring and controlling Processes
- handling processing errors

Processes can be linked to network and application activities, generating a continuous cycle of processing. For example, a network message can trigger a file transfer that is used by another application.

As a Process executes and after it completes, audit information is available to analyze and use for future processing.

Features that you can specify in a Process include:

- Scheduling—Setting a Process to run at a specific day and time. Processing can be scheduled to run automatically at a specified date or interval, without any operator intervention.
- Integration with native applications—Invoke native applications from within a IBM Connect:Direct Process.
- Integration with existing security systems—Specify user IDs and passwords within a Process that allow it to work within your existing network security system

- Data transmission integrity—Specifying checkpoint and restart intervals within a file transmission, so that if a transmission fails at some point, it can be restarted automatically from the most recent checkpoint.
- User notification—Automatically notify users of successful and unsuccessful transfers.

Process Language

A IBM Connect:Direct Process uses its own scripting language that defines the work that you want the Process to do. The following are the statements used in IBM Connect:Direct Processes:

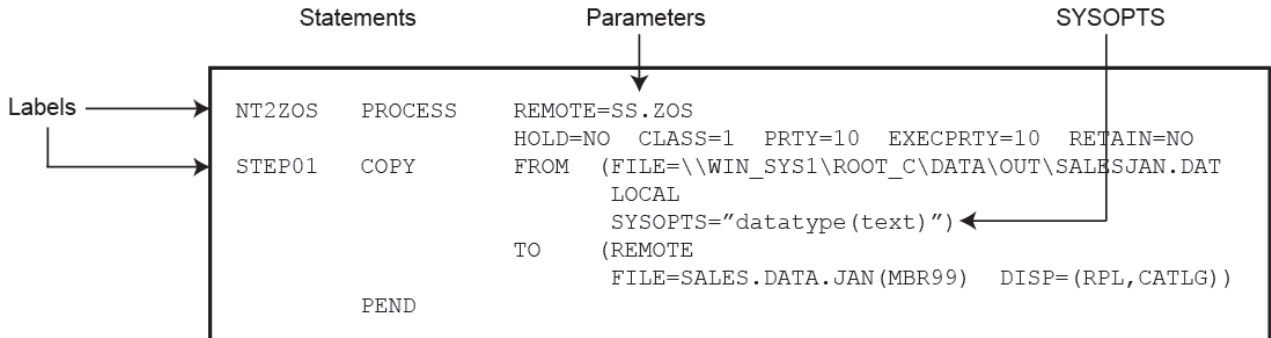
Statement	Description
PROCESS	Defines general Process characteristics. This statement is always the first statement in a Process. Among the items the Process statement specifies are: <ul style="list-style-type: none"> • The name of the secondary node in the Process • The Process priority • When to start the Process • Who to notify upon completion • Whether IBM Connect:Direct should keep a copy of the Process to execute in the future
COPY	performs a data transfer. The COPY statement also specifies various file transfer options, including: <ul style="list-style-type: none"> File allocation File disposition options File renaming Data compression options
RUN JOB	submits a job or application to the host operating system. The Process continues running and does not wait for the submitted job or application to complete. This is known as asynchronous processing.
RUN TASK	submits a job or application to the host operating system. The Process waits for the job or application to complete before continuing. If the job or application does not complete, the rest of the Process does not run. This is known as synchronous processing.
SUBMIT	submits a Process from within another Process. The SYMBOL statement enables Processes to be modular. This enhances processing flexibility, as you can modify Process modules as necessary without altering the master Process.
SYMBOL	replaces symbolic strings within a Process with parameter values. The SYMBOL statement eliminates the need to hard-code file names and values within a Process. Instead, the SYMBOL statement allows values to be substituted within a Process, enabling a Process to be reused for different file transfers.
Conditional (IF, EIF, ELSE, EXIT, GOTO)	controls Process execution by testing Process step return codes with conditional logic statements. For example, if a file transfer successfully completes, the Process can use the SUBMIT statement to initiate a second Process. If the Process transfers files, it can also send an error message to the operator.
pend	indicates the end of a Process. This statement is only valid for Connect:Direct for UNIX and Microsoft Windows.

The Process statement must be the first statement in a Process. The statements after the Process statement can follow in any sequence. Each statement uses parameters to control Process activities such

as execution start time, user notification, security, or accounting data. These parameters can be specified within the Process or you can specify them when you submit the Process. The parameters for a statement vary according to platform.

Process Components

The following example shows the components of a IBM Connect:Direct Process:



Note: The maximum size allowed for a Process is 1 MB.

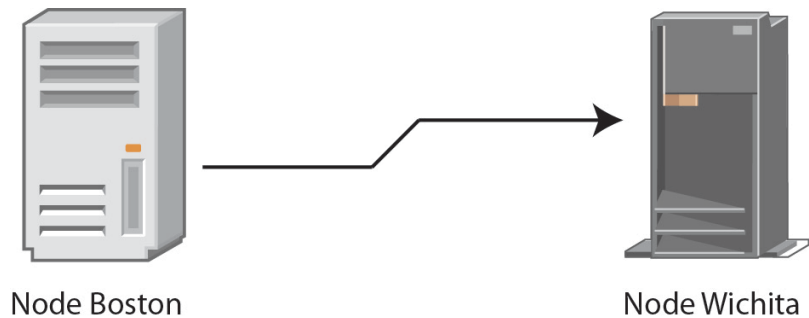
The following table describes these components:

Component	Description
Label	A user-defined 1-8 character alphanumeric string that identifies the IBM Connect:Direct statement. The label must begin in column one. The first character of the label must be alphabetic. The PROCESS statement is the only statement that requires a label; it is optional on all other statements. The PROCESS statement must be on the same line as the label.
Statement	The statement specifies the requested function. Statements must begin after column one. If a statement begins in column one, IBM Connect:Direct considers it a label. For more information on how to code IBM Connect:Direct statements to transfer files and submit business processes to IBM B2B Integrator, see <i>Transmitting Files between IBM Connect:Direct and B2B Integrator</i> .
Parameters and Subparameters	Parameters and subparameters specify further instructions for the statement. They must be separated from statements by one or more blanks or commas. Multiple symbolic substitutions must be separated by one or more spaces. There are two types of parameters: keyword and positional. <ul style="list-style-type: none"> • Keyword—Keyword parameters are usually followed by an equal sign and may have a set of subparameters. • Positional—Positional parameters must be entered in a specific order, with commas replacing any parameter omitted. These parameters are always to the right of the equal sign. Positional parameters must be enclosed in parentheses, with the parentheses optionally preceded and followed by blanks or commas. A positional parameter or the variable information in a keyword parameter sometimes is a list of subparameters. The list may contain both positional and keyword parameters. Positional subparameters must be enclosed in parentheses, with the parentheses optionally preceded and followed by blanks or commas.

Component	Description
SYSOPTS	<p>SYSOPTS (system operations) are a specialized type of parameter used by every IBM Connect:Direct platform. SYSOPTS specify platform-specific commands to perform during a Process. For example, when transferring a file from a mainframe system to a UNIX system, you use SYSOPTS to specify that the file be translated from EBCDIC to ASCII and that any trailing blanks be removed.</p> <p>See SYSOPTS Syntax for more information about using SYSOPTS.</p>

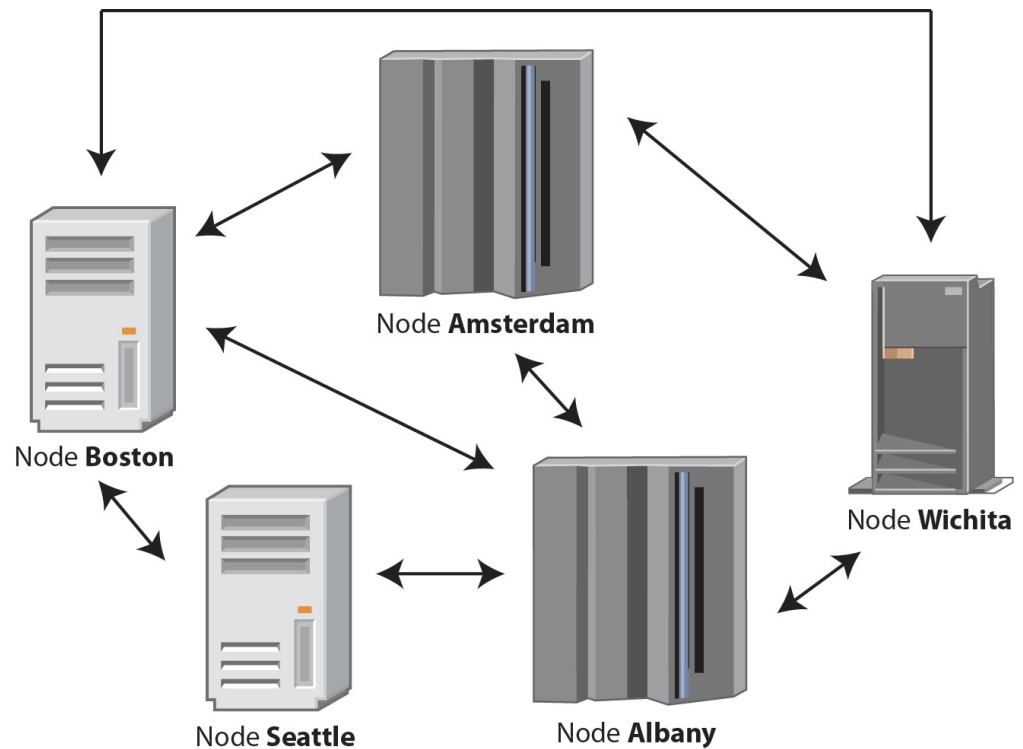
Concept of Nodes

Understanding the concept of nodes is critical to understanding how Processes work. A node is a computer in a network, as shown in the following example:



In this example, a computer in the node Boston communicates with a computer in the node Wichita.

A computer network can have numerous nodes, as shown in the following example:



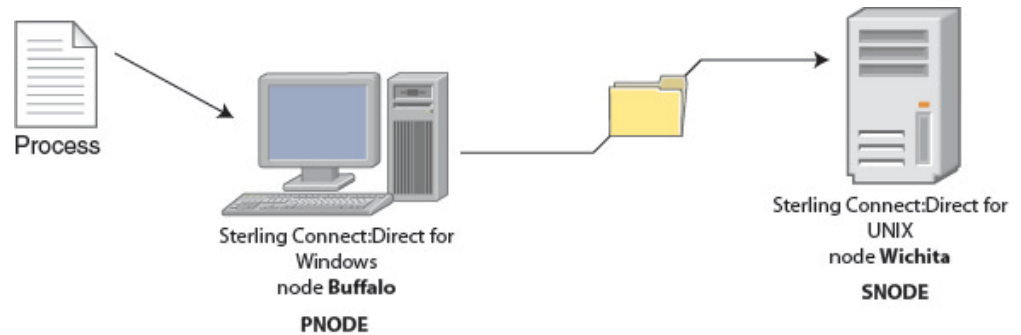
In a IBM Connect:Direct network, each IBM Connect:Direct server is considered a node. A IBM Connect:Direct server can be one of two node types:

- The primary node (PNODE). This is the IBM Connect:Direct server that initiates and controls the processing. It is the server where the IBM Connect:Direct Process is submitted.

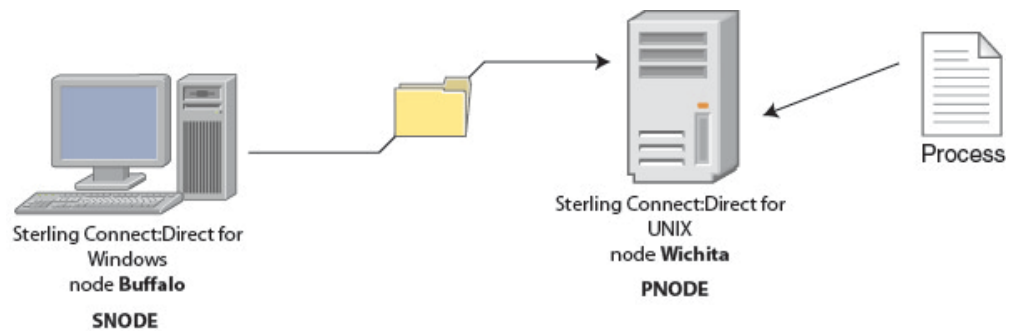
- The secondary node (SNODE). This is the IBM Connect:Direct server that works with the PNODE to accomplish the processing. For example, it may be the IBM Connect:Direct server that receives a file, or the IBM Connect:Direct server that sends a file to the PNODE.

The Process defines which node is the PNODE and which is the SNODE. A node can be a PNODE in one Process and an SNODE in another Process, as shown in the following examples.

The first example shows a simple file transfer. A Process is submitted to a Connect:Direct for Microsoft Windows node to send, or push, a file to a Connect:Direct for UNIX node. In this example, the Connect:Direct for Microsoft Windows node is the PNODE and the Connect:Direct for UNIX node is the SNODE.



In the next example a Process is submitted to a Connect:Direct for UNIX node to receive, or pull, a file from the Connect:Direct for Microsoft Windows node. In this example, the Connect:Direct for UNIX node is the PNODE and the Connect:Direct for Microsoft Windows node is the SNODE.

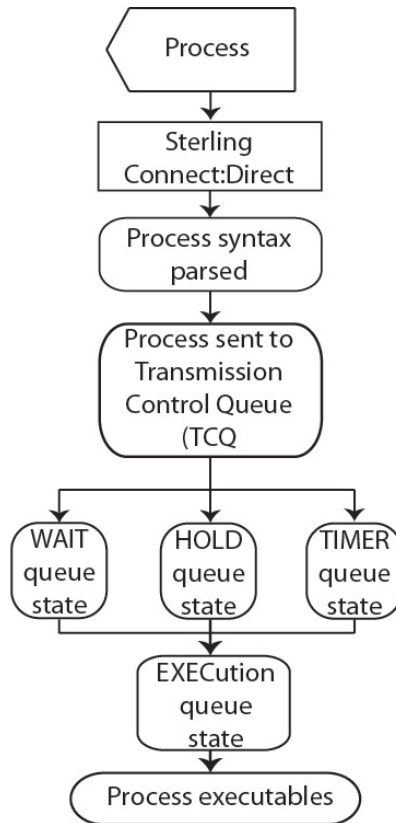


Note that the PNODE or SNODE designation has nothing to do with the file transfer direction. The PNODE and SNODE can both send or receive files.

For information about communicating with B2B Integrator, see *Transmitting Files between IBM Connect:Direct and B2B Integrator*.

How Do Processes Work?

The following illustration shows how a Process executes:



The following explains the Process steps:

Step	Description
Process submitted	A user submits a Process from a IBM Connect:Direct Process library or from the Connect:Direct Browser User Interface.
Process syntax parsed	The parser within IBM Connect:Direct verifies the Process syntax.

Step	Description
Process sent to Transmission Control Queue (TCQ)	<p>If the Process passes syntax checking, it is placed in the appropriate work queue according to Process parameters, such as priority, class, and start time. The IBM Connect:Direct work queues are jointly referred to as the TCQ. A Process is in one of the following states in the TCQ:</p> <ul style="list-style-type: none"> • EXECUTION–The Process is executing. • WAIT–The Process is waiting until a connection with the SNODE is established or available. Processes in the WAIT queue state may also be waiting for their turn to execute on an existing session. • HOLD–Process execution is on hold. The Process may have been submitted with a HOLD or RETAIN parameter. The Process is held on the queue until released by an operator or the SNODE connects with a request for held work. The HOLD queue state also applies to Processes that stop executing when an error occurs. • TIMER–The Process was submitted with a STARTT parameter that designates the time, date, or both that the Process should execute. Processes that initially failed due to inability to connect with the SNODE or because of a file allocation failure can also be in this queue state waiting for their retry interval to expire. Such Processes will retry automatically. <p>A queued Process can be queried and manipulated through IBM Connect:Direct commands such as SELECT, CHANGE, DELETE, FLUSH, and SUSPEND PROCESS. For complete information on the IBM Connect:Direct commands and the various queues, refer to the user guide for your platform.</p> <p>A message indicating that the Process was submitted successfully is created when the Process is placed into the TCQ. The Process statements have been checked for syntax, but the Process may not have been selected for execution.</p>
Process Executes	The Process is selected for execution based on Process parameters and the availability of the SNODE.

How to Build Processes

Building a Process involves analyzing the business task and creating and submitting a Process

Analyze the Business Task

The first and most important step in creating a Process is to analyze the business task you want to accomplish. Most IBM Connect:Direct Processes copy files from one location to another (although Processes can also call external programs or even other Processes).

Consider the following factors before creating a Process.

- What files do you want to copy?
- Where do you want to copy the files from, and where to?
- What IBM Connect:Direct platforms are involved in the transfer?
- Will the Process run at regular dates and times?
- What security is required to access the IBM Connect:Direct servers that will execute the Process?
- Should the files be compressed during transmission?
- Do you want to perform checkpoint/restart on the transmission, enabling the transmission to resume at a defined point in case of failure?
- Who needs to be notified of Process status and completion?
- Will the Process call an external program?

- Will the Process need branching to perform additional or alternate processing, depending on previous results?

You may find it helpful to create a flow chart of the Process before creating it.

Create and Submit a Process

After you define your business need, you can create and submit a Process for execution in the following ways:

- Through the Process Builder feature of the Sterling Connect:Direct Browser User Interface, a Web-based interface to a IBM Connect:Direct server. The Connect:Direct Browser User Interface is distributed with Connect:Direct for z/OS®, HP NonStop, UNIX, and Microsoft Windows.

The Process Builder is a Graphical User Interface that enables you to build, modify, and save Processes. The Process Builder handles IBM Connect:Direct Process syntax rules automatically. The Process Builder eliminates the typographical mistakes made when creating Processes with a text editor. You can also validate Process syntax and submit completed Processes from the Process Builder.

You can use the Process Builder to modify Processes created with a text editor. Likewise, Processes created with the Process Builder feature can be edited with a text editor.

See [Build Processes With the Process Builder](#) for procedures to create a Process with the Connect:Direct Browser User Interface Process Builder.

- Through the IBM Connect:Direct Requester for Microsoft Windows, which is a graphical interface to Connect:Direct for Microsoft Windows. See [Create Processes with Requester](#) for procedures to create a Process with the IBM Connect:Direct Requester for Microsoft Windows.
- A text file that is submitted to a IBM Connect:Direct server through a batch utility, command line, or a user written program through the Sterling Connect:Direct Application Program Interface (API). See [Create Processes with a Text Editor](#) for procedures to create a Process with a text editor.
- Through Connect:Direct for z/OS. See the *IBM Sterling Connect:Direct for z/OS User Guide* for information about the IUI.

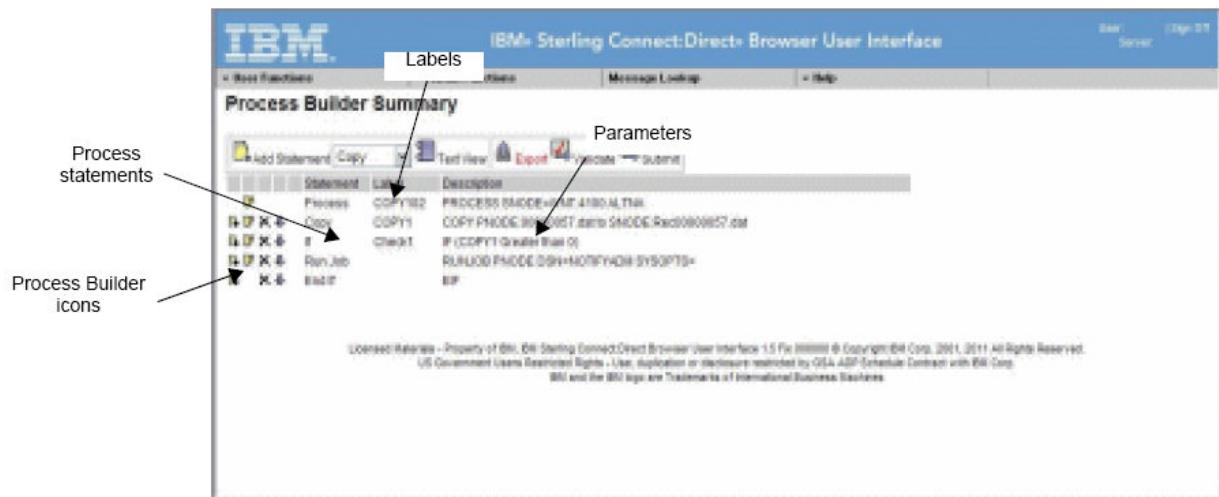
Building Sterling Connect:Direct Processes With the Process Builder

Build IBM Connect:Direct Processes With the Process Builder

The Process Builder is a graphical interface within the Connect:Direct Browser User Interface that enables you to build, modify, and save Processes. The Process Builder handles IBM Connect:Direct Process syntax rules automatically, such as inserting quotes in SYSOPTS statements. The Process Builder eliminates the typographical mistakes made when creating Processes with a text editor. You can also validate Process syntax and submit completed Processes from the Process Builder.

Note: The maximum size allowed for a Process is 1 MB.

The following is the Process Builder home page:



The following table explains the icons on the Process Builder Summary page:

Button	Description
*New	Displays the Process Builder Process Statement Main Options page to create a new Process.
Import	Imports the specified Process into the Process Builder.
+Add Statement	Displays a data entry page for the statement type specified in the list box.
Text View	Switches from a graphical view of the Process to a text view.
Export	Saves the Process in text format in a specified location.
Validate	Sends the Process to the IBM Connect:Direct server for Process syntax validation.
Submit	Submits the Process to a IBM Connect:Direct server.
	Adds a new Process statement using the parameters from the current statement.
	Edits the statement.
	Deletes the statement.
	Moves the statement down in the Process.
	Indicates that the Process statement contains invalid syntax.

Building a Process with the Process Builder consist of the following tasks:

- Signing on to the Connect:Direct Browser User Interface

- Creating a Process statement
- Adding statements for the tasks you want to accomplish
- Validating the Process
- Executing the Process


Signing on to the Connect:Direct Browser User Interface

1. Enter the URL for the Connect:Direct Browser User Interface through your browser. Acquire this URL from the system administrator if you do not know it.
2. Select the IBM Connect:Direct node to sign on to from the **Select Node** box, or type the node name. If this node is already configured in the Connect:Direct Browser User Interface, go to Step 5.
3. Type the IP address or host name of the IBM Connect:Direct system that you want to sign on to.
4. Type the port number of the IBM Connect:Direct system that you want to sign on to.


Note: If the node you are signing on to is already configured in the Connect:Direct Browser User Interface, you can leave the three previous fields blank.
5. Specify the protocol to use. Default specifies to use the value defined in the node. If there is no node definition, default specifies to use TCP/IP.
6. Type your user ID.
7. Type your password.
8. Click **Sign On**.

Creating a Process Statement

Every Process begins with a Process statement that defines general Process information. In the Connect:Direct Browser User Interface, the Process statement consists of five pages of options that define various processing options, such as when and where the Process is submitted, if a user is notified when a task completes, who can run the Process, and symbolic variables to be substituted in the Process.

1. From the **User Functions** menu, select **Process Builder** to display the Process Builder Summary page.
2. Click  ***New** to display the Process Builder Process Statement Main Options page.


Asterisks indicate required fields on this page.
3. Type a name for the Process.
4. Type the **PNODE** for the Process.

This field already contains the node that you are signed on to. You can change it to any defined node. You do not need to be signed onto the node to specify it.
5. Select the **PNODE platform**.
6. Type the **SNODE** for the Process.
7. Select the **SNODE platform**.
8. Select another Process Statement option, or click  to return to the Process Builder Summary page.

Setting Options to Control When a Process Executes and Notification

1. Click **Control** to display the Process Builder Process Statement Control Options page.


All fields on this page are optional.
2. Type the **Start Time** if you want to the process to execute at a particular time.
3. Select the **Hold Status**.
4. Select the **Priority** for the Process.
5. Select the **Retain Option**.
6. Select the **class**.

7. Type the **user ID** of the person to notify when the Process finishes. This feature is not available for Connect:Direct for HP NonStop.
8. Select another Process Statement option, or click  to return to the Process Builder Summary page.

Setting Process Security Options


1. Click **Security** to display the Process Builder Process Statement Security Options page.

All fields on this page are optional.

2. Type the **PNODE User ID**.
3. Type the **PNODE Password**.
4. Type the **SNODE User ID**.
5. Type the **SNODE password**.
6. Select another Process Statement option, or click  to return to the Process Builder Summary page.


Supplying Accounting Data

Accounting data is a free-form information that you define and use to track Process execution and data transfers. You can track data transfers by cost centers, department numbers, satellite locations, or any other type of code or identification that would benefit the management of data tracking.

1. Click **Accounting** to display the Process Builder Process Statement Accounting Data page.
All fields on this page are optional.
2. Type a text string to use as accounting information for the PNODE. You can enter up to 256 characters.
3. Type a text string to use as accounting information for the SNODE. You can enter up to 256 characters.
4. Select another Process Statement option, or click  to return to the Process Builder Summary page.


Defining Symbolic Variables

You use the Symbolic Variables page to specify or override symbolic variables when submitting a Process. IBM Connect:Direct substitutes the assigned value for the variable during Process execution.


1. Click **Variables** to display the Process Builder Process Statement Symbolic Variables page.
All fields on this page are optional.
2. Type the variable names and values you have created.
3. Select another Process Statement option, or click  to return to the Process Builder Summary page.

Creating a Copy Statement

The Copy statement copies a file from one IBM Connect:Direct node to another. The Copy Statement page contains several subpages that allow you to specify copy options. Copy options vary according to platform.

1. From the Process Builder Summary page, select **Copy** and  **Add Statement** to display the Process Builder Copy Statement Main Options page.


Asterisks indicate required fields on this page.

2. Type a **label** for the Copy step.
3. Select the **Copy Direction**.
4. Type the **Source File name**. If the file is on the PNODE, you can click  to browse to the file.

Note: The browse feature is only available on Connect:Direct for z/OS 4.4, Connect:Direct for UNIX 3.6, Connect:Direct for Microsoft Windows 4.1, and Connect:Direct for HP NonStop 3.4 or later

releases. Special characters (such as single quotes) in the directory name or file name are not supported for the browse feature.

5. Select the **Source DISP**.

6. Type the **Destination File name**. If the file is on the PNODE, you can click  to browse to the file.

7. Select the **Destination DISP**.

8. Select the **Compression** characteristics if you want to compress the file during transmission.

9. Select the **Checkpoint/Restart** characteristics if you want IBM Connect:Direct to set checkpoints when it transmits the file.

10. Select another Copy Statement option, or click  to return to the Process Builder Summary page.

Specifying Copy From Options

You use the Copy Statement From page to specify additional copy options for the platform that you are copying a file from. You are presented with options for the operating system you are copying from.

The following procedure describes the Copy From options for the z/OS platform. See the Connect:Direct Browser User Interface Help for information about other platforms

1. Type the **Source File Name**. If the file is on the PNODE, you can click  to browse to the file.

2. Type the **Data Control Block (DCB)**.

3. Select the disposition of the source file using the following three subfields:

- **Access**
- **Normal Term**
- **Abnormal Term**




4. Type the **Unit address**.

5. Type the **Volume number**.

6. Type the **Label**.

7. Type the **SYSOPTS**.


8. Do one of the following:

- Click  to return to the Process Builder Summary page
- Click  to return to the Process Builder Copy Statement Main Options page.
- Click  to display the Process Builder Copy Statement To Options page.

Specifying Copy To Options

Use the Copy Statement To page to specify options for the platform that you are copying a file to. You are presented with options for the operating system you are copying to.




The following procedure describes the Copy To options for the z/OS platform. See the Connect:Direct Browser User Interface Help for information about other platforms

1. Type the **Destination File Name**. If the file is on the PNODE, you can click  to browse to the file.

2. Type the **Data Control Block (DCB)**.

3. Select the destination file disposition using the following three subfields:

- **Access**

- **Normal Term**
 - **Abnormal Term**
4. Type the **Space** value.
 5. Type the **Unit address**.
 6. Type the **Volume number**.
 7. Type the **Label**.
 8. Type the **Typekey**.
 9. Type the **SYSOPTS**.
 10. Do one of the following:
 - Click  to return to the Process Builder Summary page
 - Click  to return to the Process Builder Copy Statement Main Options page.
 - Click  to display the Process Builder Copy Statement From Options page.




Creating a Run Task Statement

The Run Task statement calls external programs to run on a IBM Connect:Direct node. The Process waits until the external program completes before continuing. When the external program completes, the Run Task returns a completion code that indicates program success. This completion code can be used by subsequent Process conditional statements (see [Creating Conditional Statements](#)).

You can pass parameters to the external program as Run Task statement SYSOPTS.

The following are items to remember when writing programs called by Run Task statements:

- Do not specify programs in the Run Task statement that require user intervention.
- Do not use a completion code of 16 in the external program, or the Process will fail.


1. From the Process Builder Summary page, select **Run Task** and  **Add Statement** to display the Process Builder Run Task Statement page.
2. Type a label for the Run Task step.
3. Select whether the Run Task is submitted from the PNODE or on the SNODE.
4. Type the full path to the external program. If the program is on the PNODE, you can click  to browse to it.
5. Type the System Operations (SYSOPTS) or any optional parameters to pass to the program.
6. Click  to return to the Process Builder Summary page.

Creating a Run Job Statement


The Run Job statement executes external programs or commands to run on a IBM Connect:Direct node. These programs run concurrently with the Process. Unlike the Run Task statement (see [Creating a Run Task Statement](#)), the Process does not wait for the program to finish.

When the Run Job statement completes, it returns a completion code. This completion code indicates the success of the Run Job statement, and not the success of the program or command.


You can pass parameters to the external program as Run Job statement SYSOPTS.

1. From the Process Builder Summary page, select **Run Job** and  **Add Statement** to access the Process Builder Run Job Statement page.
2. Type a label for the Run Job step.

3. Select whether the Run Job is submitted from the PNODE or on the SNODE.

4. Type the full path to the external program. If the program is on the PNODE, you can click  to browse to it.

5. Type the System Operations (SYSOPTS) or any optional parameters to pass to the program.

6. Click  to return to the Process Builder Summary page.

Creating Conditional Statements


Conditional statements are used to branch processing within a Process, based on the result of a previous Process step. For example, when a Process performs a file copy, a conditional statement can test if the copy was successful. If the copy was successful, the Process continues processing subsequent statements. If the copy was unsuccessful, the Process can call a user-defined program that sends an error to the console operator and stops processing.

Conditional statements test against the completion code of a previous step. Conditional statements can be nested so that a Process can test for multiple results and react accordingly.

This procedure shows how to create a simple conditional statement that tests if a copy was successful. If the copy was successful, the Process executes the next statement. If the copy was not successful, the Process executes a Run Task that calls another program.

1. Use the Process Builder to create a new Process (see [Creating a Process Statement](#)).

2. Create a Copy statement and label it STEP1 (see [Creating a Copy Statement](#)).

3. From the Process Builder Summary page, select **If** and  **Add Statement** to access the Process Builder If Statement page.

4. Type a label for the If step.

5. Select the label that you want to test against from the list box (STEP1 in this example).


You can also type the name in the Label field, if you have not created the label yet.

6. Select **GT** (greater than) from the **Operator** field.

7. Select **4** from the **Value** field.

8. Click  to return to the Process Builder Summary page.

The If statement is added to the Process. Note that Process Builder appends Then to the end of the statement.

9. From the Process Builder Summary page, select **Run Task** and  **Add Statement** to access the Process Builder Run Task Statement page.

10. Create a Run Task statement that calls another program ([Creating a Run Task Statement](#)).




11. Click  to return to the Process Builder Summary page.

12. Select **Endif** and  **Add Statement**.

The conditional statement is completed. You can continue adding other statements as necessary to continue processing.

Creating a Submit Statement



The Submit statement submits another Process from within a Process. The submitted Process can execute on either the PNODE or SNODE.

1. From the Process Builder Summary page, select **Submit** and  **Add Statement** to display the Process Builder Submit Statement page.
2. Type a label for the Submit step.
3. Select whether the Process is submitted from the PNODE or on the SNODE.
4. Type the full path and name of the Process. If the Process is on the PNODE, you can click  to browse to it.
5. Type a new 1-8 character name for the submitted Process, if you want to rename it when it executes.
6. Click  to return to the Process Builder Summary page.

Validating Process Syntax

- To validate Process syntax, click  **Validate** from the Process Builder Summary page.
The Process is sent to the IBM Connect:Direct server for syntax validation. If the Process syntax is valid, the message **Validation completed successfully** is displayed.
If the Process syntax is not valid, an error message is displayed. Investigate and correct any errors before submitting the Process.

Saving and Submitting a Process

1. From the Process Builder Summary page, click  **Export**.
The Process Save page is displayed.
2. Click the file name **Link** to display the Save As window.
Note: On some Web servers, clicking the file name **Link** displays the Process instead of the Save As window. If this happens, click the browser's **Back** button to return to the Process Save page. Then right-click the file name **Link** and select **Save Target As**.
3. From the Save As window, select a location to save the Process file, rename the file if desired, and click **Save**.
4. To submit a Process, click  **Submit** from the Process Builder Summary page.

Viewing a Process in Text Format

1. From the Process Builder Summary page, click  **Text View**.
The Process is displayed in text format. This view is read-only. You cannot make changes in this view.
Note: Connect:Direct Browser User Interface adds a `/*BEGIN_REQUESTER_COMMENTS` comment block to the beginning of the Process. This block contains operating system and PNODE/SNODE information. This comment block maintains compatibility with Processes built with the IBM Connect:Direct Requester.
2. Click  to return to the Process Builder Summary page.





Editing a Process

1. From the Process Builder Summary page, click  to navigate to and select the Process.

- Click  **Import**.

The Process statements are displayed.

- Click one of the following icons next to a Process statement to edit that statement:

Icon	Description
	Adds a new Process statement using the parameters from the current statement.
	Edits the statement.
	Deletes the statement.
	Moves the statement down in the Process.

- When finished, click  **Export** to save the file.

Creating Processes with a Text Editor

Create Processes with a Text Editor

A IBM Connect:Direct Process is a series of statements and parameters that perform data movement and manipulation activities such as moving files, running jobs, programs, and commands, and handling processing errors.

A IBM Connect:Direct Process uses its own scripting language that defines the work that you want the Process to do. This topic describes how to create a Process using a text editor, such as Notepad (or an ISPF IBM Connect:Direct IUI panel on a mainframe). It provides several example Processes.

Note: The maximum size allowed for a Process is 1 MB.

Creating a Process That Executes at the Same Time Every Day

This example creates a simple Process that executes at 4:00 p.m. daily. The Process runs on a Connect:Direct for z/OS. The parameters are:

Parameter	Value
PNODE	CDZOS.NEW.YORK
SNODE	CDZOS.NEW.YORK

- Open a text editor (on a mainframe, create a new member in the PDS public process library dataset).
- Type a 1-8 character name for the Process. This name must begin with an alphabetic character in position 1. In this example, the Process is named DAILYPRC.

```
DAILYPRC
```

- Type a few spaces, then type the keyword PROCESS.

```
DAILYPRC  PROCESS
```


The number of spaces separating keywords does not matter. Use whatever helps you understand the statement.

4. Type a few spaces, then type the PNODE parameter followed by the PNODE value.

```
DAILYPRC  PROCESS  PNODE=CDZOS.NEW.YORK
```

5. Because we will add additional parameters for the PROCESS statement, we must use a continuation mark to show that parameters continue on a separate line. So, add a few more spaces, then type a hyphen and press RETURN to move to the next line.

```
DAILYPRC  PROCESS  PNODE=CDZOS.NEW.YORK  -
```

6. Type some spaces until you are under the PNODE parameter, then type the SNODE parameter and its value.

```
DAILYPRC  PROCESS  PNODE=CDZOS.NEW.YORK  -
                SNODE =CDZOS.NEW.YORK
```

7. Type some more spaces, then type a hyphen continuation mark and press RETURN.

```
DAILYPRC  PROCESS  PNODE=CDZOS.NEW.YORK  -
                SNODE =CDZOS.NEW.YORK  -
```

8. Type some spaces until you are under the SNODE parameter, then type the STARTT parameter and its value (4:00 p.m. in this example).

```
DAILYPRC  PROCESS  PNODE=CDZOS.NEW.YORK  -
                SNODE =CDZOS.NEW.YORK  -
                STARTT=(,4:00:00PM)
```

The value (*,4:00:00PM*) indicates that this Processes will run daily at 4:00 p.m.

9. Type some more spaces, then type a hyphen continuation mark and press RETURN.

```
DAILYPRC  PROCESS  PNODE=CDZOS.NEW.YORK  -
                SNODE =CDZOS.NEW.YORK  -
                STARTT=(,4:00:00PM)  -
```

10. Type some spaces until you are under the STARTT parameter, then type RETAIN=YES

```
DAILYPRC  PROCESS  PNODE=CDZOS.NEW.YORK  -
                SNODE =CDZOS.NEW.YORK  -
                STARTT=(,4:00:00PM)  -
                RETAIN=YES
```

11. The RETAIN parameter saves the Process on the Process queue after it executes, so that it can run again.
12. Save the file to a library that can be accessed by IBM Connect:Direct.
13. Use the IBM Connect:Direct SUBMIT command to submit the Process (or use the SB option on a Connect:Direct for z/OS for IUI panel.)

Creating a Process That Copies a File

This example creates a Process that copies a file from one z/OS node to another. The copy excludes all members that begin with DM. It allocates a data set on the destination node using the source file attributes. After the copy completes, a user notification is sent. The parameters are:

Parameter	Value
PNODE	CDZOS.NEW.YORK
SNODE	CDZOS.CHICAGO
User ID to notify	NYC6
File to copy	NYC1.LINKLIB

1. Open a text editor (on a mainframe, create a new member in the PDS public process library dataset).
2. Type a 1-8 character name for the Process. In this example, the Process is named PROCCOPY.

```
PROCCOPY
```

3. Type a few spaces, then type the PROCESS keyword and the PNODE parameter and values.

```
PROCCOPY  PROCESS  PNODE=CDZOS .NEW .YORK
```

4. Type some more spaces, then type a hyphen continuation mark and press RETURN.

```
PROCCOPY  PROCESS  PNODE=CDZOS .NEW .YORK  -
```

5. Type spaces until you are under the PNODE keyword, then type SNODE parameter and value, followed by a hyphen continuation mark. Press RETURN.

```
PROCCOPY  PROCESS  PNODE=CDZOS .NEW .YORK  -
                SNODE=CDZOS .CHICAGO      -
```

6. Under the SNODE keyword type the NOTIFY parameter followed by the user ID to notify upon Process completion. As this is the last parameter for the PROCESS keyword, do not add a continuation mark.

Press Enter.

```
PROCCOPY  PROCESS  PNODE=CDZOS .NEW .YORK  -
                SNODE=CDZOS .CHICAGO      -
                NOTIFY=NYC6
```

7. Next we will add the COPY statement. Type a 1-8 character label for the statement, beginning in position one. In this example, the label is called STEP1.

```
PROCCOPY  PROCESS  PNODE=CDZOS .NEW .YORK  -
                SNODE=CDZOS .CHICAGO      -
                NOTIFY=NYC6
STEP1
```

8. Type some spaces, then type the COPY FROM statement followed by an open parenthesis, the DSN parameter and value.

```
PROCCOPY  PROCESS  PNODE=CDZOS .NEW .YORK  -
                SNODE=CDZOS .CHICAGO      -
                NOTIFY=NYC6
STEP1    COPY FROM  (DSN=NYC1 .LINKLIB
```

9. Type some more spaces followed by a hyphen continuation mark and press RETURN.

PROCCOPY	PROCESS	PNODE=CDZOS.NEW.YORK	-
		SNODE=CDZOS.CHICAGO	-
		NOTIFY=NYC6	
STEP1	COPY FROM	(DSN=NYC1.LINKLIB	-

10. Type some spaces followed by the EXCLUDE parameter.

PROCCOPY	PROCESS	PNODE=CDZOS.NEW.YORK	-
		SNODE=CDZOS.CHICAGO	-
		NOTIFY=NYC6	
STEP1	COPY FROM	(DSN=NYC1.LINKLIB	-
		EXCLUDE=	

11. We want to exclude all members that begin with DM from the copy, so type (DM*) followed by a close parenthesis (to close the DSN parameter) and the hyphen continuation mark.

PROCCOPY	PROCESS	PNODE=CDZOS.NEW.YORK	-
		SNODE=CDZOS.CHICAGO	-
		NOTIFY=NYC6	
STEP1	COPY FROM	(DSN=NYC1.LINKLIB	-
		EXCLUDE=(DM*)	-

12. Press RETURN to move to the next line. Type spaces until you are under FROM, then TYPE the TO parameter followed by the DSN to assign the file on the destination node.

PROCCOPY	PROCESS	PNODE=CDZOS.NEW.YORK	-
		SNODE=CDZOS.CHICAGO	-
		NOTIFY=NYC6	
STEP1	COPY FROM	(DSN=NYC1.LINKLIB	-
		EXCLUDE=(DM*))	-
	TO	(DSN=CHI6.NEW.LINKLIB)	

13. Save the file to a library that can be accessed by IBM Connect:Direct.

14. Use the IBM Connect:Direct SUBMIT command to submit the Process.

Creating Processes with Sterling Connect:Direct Requester

Create Processes with IBM Connect:Direct Requester

IBM Connect:Direct Requester for Microsoft Windows provides a graphical interface to build, modify, and save Processes. You can select parameters from a drop-down list and automatically insert the correct syntax for each parameter. After you define a Process, you can validate the syntax. The connections between some clients and a Connect:Direct Server are unsecure. Passwords sent by one of these clients to a C:D Server are obfuscated, but the session is not encrypted. The clients are: the CD Requester, the Windows CLI, any user-written Windows SDK client and FileAgent.

Note: The maximum size allowed for a Process is 64 KB.

The following is a sample Process built using IBM Connect:Direct Requester for Microsoft Windows. The Process copies a binary file from a Microsoft Windows node to a UNIX node. If the copy is successful, a run task statement is performed on the Microsoft Windows node to delete the source file on the Microsoft Windows node.

Statement	Label	Description	Comment
Process	WDW2UNX	PROCESS SNODE=qa160sol3601	
Copy	COPY1	COPY PNODE:C:\Input\Binary\invoice1.dat t...	//Copying a binary file from a Wi
If	CHECK1	IF (COPY1 EQ 0)	
Run Task	DELFILE	RUN TASK PNODE PGM=Windows SYSOP...	//If copy is successful, delete
Endif		ENDIF	
End		PEND	

To make it easier to create a process using IBM Connect:Direct Requester for Microsoft Windows, first attach to the node where the Process will begin. This gives you access to the network map. Since the nodes you communicate with are defined in the network map, accessing a network map allows you to identify the node that communicate with in the Process you are defining.

Building a Process in IBM Connect:Direct Requester for Microsoft Windows requires the following tasks:

- Creating a Process Statement
- Adding other statements to perform the work
- Validating the Process
- Saving the Process file

The IBM Connect:Direct Requester for Microsoft Windows Help contains more detailed information about how to create a IBM Connect:Direct Process.

Creating a Process Statement

Every Process begins with a Process statement.

1. Select **Start > Programs > Sterling Commerce Connect Direct > CD Requester**.
2. From the **File** menu, select **New > Process**.
3. Type a Process name, from 1 to 8 alphanumeric characters, in the **Process Name** field.
4. Type or select the name of the PNODE in the **Pnode Name** field.
5. To issue a warning message if an attempt is made to submit the Process on a different PNODE, click **Warn if submitted to a different node**.
6. To issue a warning message if an attempt is made to submit the Process on a PNODE with a different operating system, click **Warn if submitted to a different operating system**.
7. Specify the SNODE by selecting the node from the drop-down menu, typing the name of the node, or specifying an IP address and port, using the following format:

```
hostname|IPaddress;portnumber|servicename
```

8. Click one of the tabs to further define the Process options.

Identifying When to Run a Process

1. Open the Process and click the **Control** tab.
2. Specify the date to run the Process: **Today**, select a specific **Date**, or identify a day to run the Process
3. Specify the time to run the task. Options include:
 - **Immediate** to run the Process as soon as it is activated. This option is only available if you selected **Today** or **Date** in the **Start Date** field.
 - **Time** to run the Process at a designated time. Type a time to start the Process in the format hh:mm:ss.
4. Click **OK** to save your changes and close the dialog box or click another tab to continue defining other control functions.

Placing the Process in the Hold or Retain Queue

To place the Process in the Hold or Retain queue to run at a later time:

1. Open the Process and click the **Control** tab.
2. To place the Process in the Hold queue for future execution, select one of the following options in the **Hold** field:
 - Select **Yes** to hold the Process in the queue in Held Initially status (HI) until explicitly released.
 - Select **Call** to hold the Process until a connection is established between the PNODE and the SNODE. The Process executes if another Process establishes connection between the nodes.
3. To place the Process in the Retain queue, select one of the following options in the **Hold** field:
 - Select **Yes** to retain the Process in the Hold queue in Hold Retain status (HR) after execution.
 - Select **Initial** to retain the Process in the Hold queue in HR status for automatic execution every time the Process Manager initializes.
4. Click **OK** to close the dialog box or click one of the other tabs to continue modifying Process options.

Note: If you select **Yes** as the value in the **Retain Execution Option** and you specify a start time, the value defined in the **Hold Execution Option** takes precedence. If you set the value of the **Hold Execution Option** to **Call**, and set **RETAIN** to **Yes**, the value in the **Hold Execution Option** is ignored.

Notifying a User when a Process Runs

1. Open the Process and click the **Control** tab.
2. Type the user ID in the **Notify Userid** field.
3. Click **OK** to close the dialog box or click one of the other tabs to continue modifying Process options.

Specifying User IDs and Passwords for PNODE and SNODE

1. Open the Process or command and click the **Security** tab.
2. Type the PNODE user ID in the **Pnode Userid** fields.
3. Type the PNODE password in the **Password** field.
4. Type the SNODE user ID in the **Snode Userid** field.
5. Type the SNODE password in the **Snode Password** fields.
6. To change the password for the user ID on the SNODE, type the new password in the **New Password** field.
7. Type the new password a second time in the **Verify New Password** field to validate the change.
8. Click **OK** to close the dialog box or click one of the other tabs to continue modifying Process options.

Adding a Copy Statement

A copy statement copies a file from one node to another.

1. From the **Process** menu, select **Insert > Copy**.
2. To identify the step within the Process, type a label of up to 8 alphanumeric characters in the **Copy Statement Label** box.
3. To copy the file to the SNODE, select **Send**.
4. To copy a file from the SNODE, select **Receive**.
5. Type the name of the source file in the **Source Filename** field.
6. Type the name of the file in the **Destination Filename** field.
7. Select the method to use to save the destination file: **NEW** to create a new file, **RPL** to replace information in an existing file or to create a new file, if the file does not exist, or **MOD** to append the transferred information to an existing file.
8. Click **OK**.

See the IBM Connect:Direct Requester for Microsoft Windows Help for all Copy statement parameters.

Adding Conditional Statements

Conditional statements branch processing within a Process, based on the result of a previous Process step. For example, when a Process copies a file, a conditional statement can test if the copy was successful. If the copy was successful, the Process continues processing additional steps. If the copy was unsuccessful, the Process can call a user-defined program that sends an error message to the console and stops processing.

Conditional processing tests against the completion code of the previous step. Conditional steps can be nested so that a Process can test for multiple results and react accordingly.

This procedure shows how to create a simple conditional statement that tests if a copy was successful. If the copy was successful, the Process executes the next statement.

1. From the **Process** menu, select **Insert > If** to add an If statement to a Process.
2. Type CHECK1 as the label for this step in the **If Statement Label** field.
3. Select COPY1 as the step label to test against from the list box.
4. Select EQ as the comparison statement in the **Operator** field.
5. Select 0 in the **Value** field to check if the copy process was successfully performed.
6. Click **OK** to save the statement. The If statement is displayed in the Process window.
7. To add the End If statement, from the **Process** menu, select **Insert > End If**.
8. Click **OK** to save the statement. The Endif statement is displayed in the Process window.

Adding a Run Task Statement

You can run programs and commands by adding the Run Task statement to a Process. The Run Task statement executes programs on the PNODE or the SNODE. In the example, the Run Task statement deletes the source file from the Microsoft Windows computer if the Copy step is successful. The Run Task statement must be added after the conditional If statement.

1. From the **Node** menu, select **Run Task**.
2. Type DELFILE as the label for the Run Task step.
3. Type Microsoft Windows as the name of the program that executes the command.
4. Type sysopts=cmd(del c:\data\out\invoice1.dat) as the command to execute if the copy step is successful.
5. Click **OK**.

Validating Process Content

After creating a Process, validate the content. Validating Process content checks the syntax for errors or missing information. Validation does not check the content of the statements, only that they

are formatted correctly. The Process validation sends messages to the Output window. A Validation Successful message means that the syntax is formatted correctly.

1. Open the Process file.
2. From the **Process** menu, select **Validate**.
3. View the messages displayed in the Output window. If messages indicate invalid statements, edit the statements and validate the content of the Process again.

Saving a Process

1. From the **File** menu, select **Save**.
2. Type a name for the Process, including the .CDP extension. Processes are saved in the Process directory.

Editing a Process

1. From the IBM Connect:Direct Requester for Microsoft Windows window, open the Process to edit.
2. Double-click the statement to edit.
3. Change the statement.
4. Click **OK**.
5. To delete a statement, highlight the statement and click **Delete**.
6. To add a new statement, right-click inside the Process and select **Insert**.

Process Language Syntax

Process Language Syntax

This topic describes the syntax used in IBM Connect:Direct Processes. If you use the Connect:Direct Browser User Interface Process Builder or the IBM Connect:Direct Requester to create Processes, some of these syntax rules are performed automatically by the software.

Those following symbols are defined as special characters, delimiters, and operators in IBM Connect:Direct:

-	hyphen		two vertical bars	&	ampersand
	(blank)	¬	(not sign)	'	(single quotation mark)
<	(less-than sign)	/	(slash)	"	(double quotation mark)
>	(greater-than sign)	\	(backslash)	[]	(brackets)
()	(parentheses)	,	(comma)	{ }	(braces)
=	(equal sign)	.	(period)	*	(asterisk)

Note: The EBCDIC Hex value for the slash (/) is x'61' and x'EO' for the backslash (\). The EBCDIC Hex value for the vertical bar (|) is X'4F.

Punctuation

This topic includes the following:

- [Commas](#)
- [Continuation Marks](#)
- [Parentheses](#)
- [Asterisks](#)

Commas

Commas have two functions:

- Separate items within a list
- Control the order of values specified as positional parameters

Use a comma to indicate omission of a positional parameter.

```
SIGNON USERID=(id, ,newpswd)
```

Do not use commas to separate multiple symbolics in a Process. Separate multiple symbolics with one or more spaces.

Continuation Marks

Use a hyphen as a continuation mark when a statement continues on multiple lines. Separate the hyphen from the preceding characters by at least one blank.

For Connect:Direct for HP NonStop: Both hyphen (-) and ampersand (&) are supported as continuation characters.

For Connect:Direct for UNIX, and OpenVMS: Continuation characters are not required.

Parentheses

Parentheses enclose lists and associate a group of values. For example, the FROM clause of the COPY statement is enclosed in one set of parentheses. Lists in the FROM clause are nested in subsequent pairs of parentheses.

Asterisks

Asterisks indicate generic specifications of parameters. With generics, you request information by specifying a single asterisk (*) or a character string plus an asterisk.

For example, the following FROM clause of a COPY statement selects all member names beginning with ACCT (the first four characters of the data set names) from the PDS named PDS.SOURCE.

```
COPY FROM (DSN=PDS.SOURCE SELECT=(ACCT*))
```

For Connect:Direct for i5/OS Processes: Asterisks precede some subparameters and must be typed as shown in the statement format.

Comments

Comments include additional information within a IBM Connect:Direct Process. Comments are allowed in the following formats:

- An asterisk (*) in column one, followed by the comment. You must use this format for statements processed by DMBATCH.
- Preceded by a slash-asterisk (/*) and followed by an asterisk-slash (*). This format can be used after a continuation mark as well as at the beginning of a line.
- Preceded by a slash-asterisk (/*), continuing over multiple lines, and terminated by an asterisk-slash (*). The terminating */ cannot begin in column one.

The following example shows every way that you can use comments.

```
/* This type of comment can be written on one line*/
/*
It can also continue across multiple lines. Remember that
the terminating asterisk-slash cannot begin in column one.
*/
COPY      FROM (
            DSN=&DSN1
            UNIT=SYSDA)
            - /* INPUT */
            - /* SYMBOLIC DATA SET */
* After submitting this Process,
* enable the Connect:Direct UNIX node.
```

Concatenation

Concatenation joins separate character strings into a single string. Different platforms use different characters to indicate concatenation.

Concatenation for Connect:Direct for z/OS, VM/ESA, VSE/ESA, and i5/OS

The operator blank-vertical bar-vertical bar-blank (||) indicates concatenation. For example:

```
DSN=CD || NODE
```

Resolves to:

```
DSN=CDNODE
```

Concatenation for Connect:Direct for HP NonStop

Connect:Direct for HP NonStop supports two concatenation operators:

- Vertical bars (||)
- Ampersand (&) for the PACCT and SACCT parameters

Typically, concatenation is not necessary for any other parameters.

The following example PROCESS statements illustrate the differences between the two types of concatenation for the SACCT parameter. In both examples, ampersands are used to indicate continuation of the PROCESS statement.

The following example shows an ampersand (&) used as a concatenation character. When used for concatenation, the ampersand (&) must be in column 80 and the remainder of the string must begin in column one of the next line to ensure that blanks are not added to the string. The entire string must be enclosed in single quotation marks.

```
PROC1      PROCESS          SNODE=CD.OS390.NODE
            SACCT='1234567890123456789012345678901234567890123456789012345678901234567890'
            SNOEID=(USERID,PASWRD)
            &
            &
```

The following example shows two vertical bars (||) used as a concatenation character. Do not use blanks before or after the two vertical bars (||) to prevent blanks from being added to the string. Each line of the SACCT string must begin and end with single quotation marks.

```
PROC1      PROCESS          SNODE=CD.OS390.NODE
            SACCT='12345678901234567890' ||
            &
```

```
'1234567890123456789'
SNODEID=(USERID,PASWRD) &
```

Concatenation for Connect:Direct for OpenVMS

Either double quotation marks (“ ”) or single quotation marks (‘ ’) can be used with the continuation character to concatenate a string spanning multiple records. The second and subsequent records must begin in column one. For example:

```
CONCAT          PROCESS          SNODE=CD.VMS.NODE
                SYMBOL &TO="$DISK1:" -
"<DIRECTORY>TEST.DAT"
STEP01         COPY      FROM      (DSN=IBMFIL SNODE) -
                TO        (DSN=&TO PNODE DISP=NEW)
```

Concatenation for Connect:Direct for UNIX and Connect:Direct for VM/ESA

Use stream input instead of concatenation or continuation symbols as required by other IBM Connect:Direct operating environments. Also, grammar is based on the sequence of parameters and arguments instead of position within the inline buffer. The exception is that comment identifiers (asterisks [*] and pound [#] signs) in column one are positional.

Concatenation is also used in conjunction with Special Purpose Bracketing and in Symbolic Substitution to join values that are represented as symbolic parameters. See the following sections for more information.

```
&USERID=BOB
DSN=CD || &USERID
```

Resolves to:

```
DSN=CDBOB
```

Symbolic resolution occurs before concatenation.

HFS File Name Considerations

When HFS file names contain internal Connect:Direct for z/OS keywords, single quote delimiters are required. For example, coding:

```
DSN=/u/myspace/DCB/test.file
```

would be rejected at submit time because DCB is an internal Connect:Direct for z/OS keyword that requires an equal sign.

Surrounding the entire file name within single quotes resolves this issue unless variable substitution is being used. If a Connect:Direct for z/OS keyword is being used along with variable substitution, then a more complex syntax is required. For example, coding:

```
SYMBOL &FILE="file"
:
DSN=/u/myspace/ -
'DCB/' -
test.&FILE
```

would produce:

```
DSN='/u/myspace/DCB/test.file'
```

All Connect:Direct for z/OS keywords used in DSN= or FILE= should be treated in this manner.

The following is a list of keywords that must receive special consideration if contained in an HFS file name:

ALIAS	AVGREC	BLKSIZE	BUFND	CASE	CKPT	CLASS
COMPRESS	COPY	CRC	DATACLAS	DATAEXIT	DATATYPE	DCB
DEBUG	DISP	DSN	DSNTYPE	DSORG	ESF	EXCLUDE
FOLD	FROM	HOLD	IOEXIT	JOB	KEYLEN	KEYOFF
LABEL	LIKE	LIMCT	LRECL	MGMTCLAS	MVSGP	NCP
NETMAP	NEWNAME	NODE	NOREPLACE	NOTIFY	OPTCD	PACCT
PARM	PDS.DIR	PDS.DIRECTORY	PGM	PNODE	PNODEID	PRECOMP
PRINT	REQUEUE	RESGDG	RETAIN	RKP	RUN	SACCT
SECMODEL	SELECT	SIGNON	SNODE	SPACE	SQL	STORCLAS
STRIP.BLANKS	SUB	SUBMIT	SUBNODE	SYSOPTS	TASK	TMPDD
TMPDSN	TO	TODAY	TOMORROW	TRTCH	TYPE	UNIT
USERID	VOL	VOLSER	WHERE	XLATE		

Special Character Strings

To maintain a special characters as part of a string, enclose the string within bracketing characters. The bracketing characters are:

- backslashes (\)
- single quotation marks (')
- double quotation marks (")

Backslashes

Backslashes indicate a character string and are not maintained as part of the string at its final resolution. The following table indicates the platforms that accept backslashes:

Processes Submitted from	Backslashes valid?
HP NonStop	No
OpenVMS	No
z/OS	Yes
i5/OS	Yes
UNIX	No
VM/ESA	Yes
VSE/ESA	Yes
Microsoft Windows	No

Use backslashes to:

- Continue a string containing special characters across multiple lines.
- Ensure that quotation marks in the string are maintained.

Both backslashes must be on the same line. If a string containing special characters continues across multiple lines, each line containing a special character must be enclosed in backslashes and concatenated. For example, the following SYSOPTS parameter for Connect:Direct for i5/OS is a quoted string and must be enclosed in backslashes when it continues across multiple lines:

```
SYSOPTS=  \ "CMD(\          ||          -  
          \SNDBRKMSG\      ||          -  
          \)"\
```

Resolves to:

```
SYSOPTS="CMD(SNDBRKMSG)"
```

If the character string includes a backslash, precede it with an additional backslash. For example:

```
PACCT=  \xd4 DEPT\MIS\      ||          -  
          \602'\
```

Resolves to:

```
PACCT='DEPT\MIS602'
```

Single and Double Quotation Marks

Use single and double quotation marks to embed special characters or blanks within a parameter or subparameter value. For example:

```
COPY TO (DSN='VMFILE FILETYPE')  
COPY TO (DSN=\"C:\PCDIR\BAT.EXE\" )
```

Strings within apostrophes (single quotes) allow the parsing of parameters as entered. Strings within quotes (double quotes) allow the resolution of &values in a quoted string.

For Connect:Direct for i5/OS and OpenVMS: Enclose the entire SYSOPTS string in double quotation marks (“ ”).

For Connect:Direct for z/OS: Enclose parameters in double quotation marks (“ ”).

For Connect:Direct for UNIX: Use double quotation marks, unless otherwise specified.

Special Character Parsing in the SUBMIT Command

Parsing of special bracketing and single and double quotes is performed differently in a SUBMIT command than in a SUBMIT statement within a Process.

For example, a SUBMIT command executed from DMBATCH, resolves:

```
SYMBOL &BATCHID2=\''\ || BATCHID || '\''\
```

to:

```
'''BATCHID'''
```

A SUBMIT statement within a Process resolves the same string to:

```
'BATCHID'
```

This is important to remember when a SUBMIT is performed between different platforms and products, such as from Connect:Direct for UNIX to Connect:Direct for z/OS to IBM Connect:Enterprise®.

Symbolic Substitution

Symbolic substitution substitutes information in a Process. When IBM Connect:Direct encounters an ampersand (&) followed by 1-8 alphanumeric characters, it substitutes a string represented by the ampersand and alphanumeric characters. For example:

```
&USERID=BOB  
DSN=CD || &USERID
```

Resolves to:

```
DSN=CDBOB
```

Separate multiple symbolics with spaces, as shown in the following:

```
SUBMIT PROC=TSTSEND &DSN1=TSTSEND.VAR0001.S200010 &RUNDATE=200012 &TSTDATE=200010
```

Symbolic resolution occurs before concatenation.

The following example encloses a string in double quotation marks to resolve the symbolic &FILTYF.

```
PROC2    PROCESS    SNODE=CD.VM &FILTYF=FT  
COPY    FROM      (DSN=OS390.DATA -  
DISP=SHR) -  
TO      (DSN="FN || &FILTYF" -  
LINK=(IVVB,WIVVB,W,191) -  
DISP=(RPL))
```

Double quotation marks are not valid for symbolic substitution in Microsoft Windows. Use the SYMBOL statement and concatenation instead.

If you are using symbolic substitution for SNODEID and want to pass both a user ID and password, you must have separate symbolic names for each entity, that is, a symbolic name for the user ID and another for the password.

For example, if you code the following:

```
&MYSTUFF=myid,mypass  
SNODED=(&MYSTUFF)
```

IBM Connect:Direct will interpret &MYSTUFF as one block rather than as two entities—a user ID and password.

To pass two pieces of information in this example, code it like the following:

```
&MYNAME=myid
&MYPASS=mypass
SNODEID=(&MYNAME,&MYPASS)
```

Intrinsic Symbolic Variables Used in Connect:Direct for z/OS and Microsoft Windows

IBM Connect:Direct provides the following intrinsic symbolic variables that you can use to substitute user-defined values when a Process is executed. This flexibility lets you use the same Process for multiple applications when these values change.

Value	Description
%DD2DSN	Specifies an allocated DD statement, which references a DSN to be passed to a Process being submitted (for Connect:Direct for z/OS)
%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. 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.
%JOBID	Specifies the job number.
%JOBNM	Specifies the job name.
%JUSER	Specifies a variable that resolves to the USERID of the submitted job.
%NUM1	Specifies the submission time of the Process in minutes, seconds, and fraction of seconds in the format mmssth.
%NUM2	Specifies the submitted time of a Process as the low order 4 bits of the milliseconds of the time expressed as 1 hex digit (a value from 0 through 15 expressed as 0 through F).
%PNODE	PNODE name where the submit occurs
%PRAND	Pseudo-random number (6 hex digits)
%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. The value returned can be used to create a file name on the node receiving the file.
%SUBDATE1	Use this parameter to substitute the submitted date in the yyyyymmdd 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 mmdyyy date format.
%SUBDATE4	Use this parameter to substitute the submitted date in the ddmyyyy 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. The return value can be used to create a file name on the node receiving the file. 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.

Value	Description
%USER	Specifies a variable that resolves to the user submitting the Process

In the following example for Connect:Direct for z/OS, the DSN specified in the FROMDD statement is DALLAS.DATA.FILE.

```
&DSN = %DD2DSN(FROMDD)
```

The DSN resolves to DALLAS.DATA.FILE when Connect:Direct for z/OS executes the Process containing the intrinsic symbolic variable

Termination

A statement is terminated by the end of data without a continuation mark.

SYSOPTS Syntax

SYSOPTS (system operations) are a specialized type of parameter used by every IBM Connect:Direct platform. SYSOPTS specify platform-specific commands to perform during a Process. For example, when transferring a file from a mainframe system to a Microsoft Windows system, you use SYSOPTS to specify that the file be translated from EBCDIC to ASCII and that any trailing blanks be removed.

All IBM Connect:Direct platforms use SYSOPTS on the Copy statement. Some platforms also use SYSOPTS on the Run Job and Run Task statements to pass parameters to the external program.

Because of operating system differences, SYSOPTS parameters and syntax vary by platform. This can be confusing when you create Processes with a text editor. (However, the IBM Connect:Direct Requester and the Connect:Direct Browser User Interface Process Builder both automatically handle Process syntax.)

You must use the SYSOPTS syntax for the platform that is performing the work. For example, if you are copying a file from z/OS to HP NonStop, then performing a Run Task on the copied file, you use the Run Task's SYSOPTS syntax for HP NonStop.

If you are sending files to B2B Integrator and want to use SYSOPTS parameters to customize the format of those files, see [Specifying File Formats using SYSOPTS](#).

Note: Depending on how you use a variable string, you may need to include bracketing characters. This situation is often required when a SYSOPTS string is sent as a symbolic parameter and must be enclosed in quotation marks.

For example, to transfer a file to a UNIX system using a symbolic variable, you would type the SYSOPTS clause as follows:

```
&SYSOPTS="\ :datatype=text:xlate=yes:"\
```

In this example, what the Process states:

```
SYSOPTS=&SYSOPTS
```

resolves to:

```
SYSOPTS=" :datatype=text:xlate=yes:"
```

The following list explains the SYSOPTS syntax differences between platforms. Remember, IBM Connect:Direct Requester and the Connect:Direct Browser User Interface Process Builder handle Process syntax automatically, so you do not need to worry about these conventions if you use those tools.

Processes Submitted from HP NonStop

Copy Statement: Copy statement SYSOPTS are expressed as HP NonStop SET commands. Enclose each SYSOPTS string in double quotation marks except when copying from Microsoft Windows to HP NonStop. For example:

```
SYSOPTS=("SET parameter")
```

When copying from Microsoft Windows to HP NonStop, enclose each SET parameter in single quotation marks and enclose the entire SYSOPTS string in double quotation marks. For example:

```
SYSOPTS="'SET parameter' 'SET parameter' 'SET parameter' "
```

There are two ways to express multiple SET command parameters:

- SET precedes each parameter. For example:

```
SYSOPTS=("SET parameter" "SET parameter")
```

- SET precedes the first parameter, and commas separate subsequent parameters. For example:

```
SYSOPTS=("SET parameter, parameter")
```

Do not use continuation marks.

Run Task Statement: Enclose a Run Task statement's SYSOPTS string in either single or double quotation marks. Enclose any literal parameter values to be passed in single quotation marks. Enclose any symbolic values (&value) in double quotation marks

Processes Submitted from z/OS

Enclose the complete SYSOPTS string in double quotation marks. Separate individual SYSOPTS parameters with spaces and use the delimiter appropriate to the platform. For example, if you are copying to Microsoft Windows, you would enclose keyword values in parentheses:

```
SYSOPTS = "DATATYPE(BINARY) XLATE(YES) STRIP.BLANKS(NO) "
```

Use backslashes (\) and concatenation characters (||) to continue the SYSOPTS string over multiple lines when the Process. For example:

```
SYSOPTS=\ "TYPE(MBR) \  
          \TEXT('CREATED BY PROC#001')\  
          \RCDLEN(133) "\
```

Processes Submitted from i5/OS

Copy Statement: Enclose all SYSOPTS parameter values in parentheses. Enclose the entire SYSOPTS string in double quotation marks. Separate subparameters with blanks. For example:


```
SYSOPTS="TYPE(FILE) PRECMPR(*YES) XTRAN(EBCXKSC) XTRANLDATA(MIXED) "
```

Run Job Statement: Enclose the string to be passed in double quotation marks. For example:

```
SYSOPTS = "string"
```

Run Task Statement: Enclose the CL command in parentheses. Enclose the entire SYSOPTS string in double quotation marks. For example:

```
SYSOPTS = "cmd(CL command) "
```

Processes Submitted from OpenVMS

Copy Statement: Enclose the SYSOPTS string in double quotation marks. Enclose each subparameter string in single quotation marks. Separate the subparameters by blanks. For example:

```
SYSOPTS="MOUNT='MUA0 TAPELABEL' NODISMOUNT"
```

Run Job Statement: Enclose the SYSOPTS string in double quotation marks. Enclose each subparameter string in single quotation marks. Separate the subparameters by blanks. For example:

```
SYSOPTS ="KEEP LOG='log-name' NOPRINT"
```

Run Task Statement: Enclose the entire SYSOPTS string in double quotation marks. Enclose each subparameter string in single quotation marks. Separate the subparameters by blanks. For example:

```
SYSOPTS="[OUTPUT='file specification']"
```

If you are calling a DCL command procedure that contains embedded blanks and quotation marks, replace the embedded blanks with underscores and remove the quotation marks. For example, specify the DCL command *MAIL/SUBJECT* "two words" filename as:

```
SYSOPTS="CMD='MAIL/SUBJECT=two_words filename' "
```

Processes Submitted from UNIX

Copy Statement: Enclose the SYSOPTS string in double quotes. For example:

```
sysopts=":datatype=text:xlate=no:pipe=yes:"
```

Run Job and Run Task Statements: Enclose the SYSOPTS string in double quotes. Separate the UNIX commands with semicolons. For example:

```
sysopts = "unix command;unix command;unix command"
```

Processes Submitted from VM

Enclose the SYSOPTS string in single or double quotes. For example:

```
SYSOPTS='!SPOOL CLASS B DIST VM1500'
```

or

```
SYSOPTS="!SPOOL CLASS B DIST VM1500"
```

Processes Submitted from VSE

Enclose each SYSOPTS parameter string in double quotation marks. For example:

```
SYSOPTS = "DBCS=(tablename,so,si,PAD)" "parameter1,parameter2"
```

Processes Submitted from Microsoft Windows

Enclose the entire string in double quotation marks. Separate the parameters by spaces. For example:

```
"xlate(yes) xlate.tbl(tbl)"
```

Chapter 2. Sterling Connect:Direct Process Statements and Parameters

HP Non-Stop Statements and Parameters

Connect:Direct for HP NonStop Process Statement

The PROCESS statement defines the attributes of a Process and is always the first statement in a Process.

The following is the Connect:Direct for HP NonStop Process statement format. Refer to the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

Label	Statement	Parameters
process name	PROcEss	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])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

Connect:Direct for HP NonStop Copy Statement for Spooler Jobs

Connect:Direct for HP NonStop supports the use of ISO/ANSI printer control characters between files on an IBM z/OS node and the HP NonStop 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 z/OS node and the HP NonStop spooler system, the IBM Connect:Direct system converts HP NonStop carriage control to ANSI or

ANSI to HP NonStop 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 z/OS node.

The following is the Connect:Direct for HP NonStop Copy statement for Spooler Jobs. See the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

Label	Statement	Parameters	
stepname	COPY	From	(
			DSN=filename FILE=filename
			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"]
)

Connect:Direct for HP NonStop Copy Statement for Guardian Disk and Tape Files

The COPY statement allows you to copy files from one node to another.

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

To copy from one IBM Connect:Direct platform to another, refer to the appropriate COPY FROM and TO pages for those platforms. For example, if the source file is located on a Connect:Direct for z/OS node, refer to the Connect:Direct for z/OS COPY FROM information. If the destination node is Connect:Direct for HP NonStop, refer to the TO information for this platform.

Note: Use SYSOPTS parameters, not DCB, when creating HP NonStop files, as not all z/OS DCB statements have an exact correlation to HP NonStop file attributes.

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.

The following is the Connect:Direct for HP NonStop Copy statement for Guardian Disk and Tape Files. See the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

Label	Statement	Parameters	
stepname	COPY	FROM	(
			DSN=filename FILE=filename
			PNODE SNODE
			DCB=([RECFM=record-format] [,BLKSIZE=no.-bytes] [,LRECL=no.-bytes
			DISP=([OLD SHR] , [KEEP DELETE])

Label	Statement	Parameters	
			IOEXIT=[exit-name (exit-name [,parameter,...])] [\$process-name]
			SYSOPTS=["SET XLATE ON YES OFF NO table-name"] ["SET OPENFILEXMT Y"]
)
		TO	(
			DSN=filename FILE=filename
			PNODE SNODE
			DCB=([BLKSIZE=no.-bytes] [,DSORG=[U 0] [R 1] [E 2] [K 3] [,KEYLEN=no.-bytes] [,LRECL=no.-bytes] [,RECFM=record-format]
			TYPE=typekey
			DISP=([NEW OLD MOD RPL SHR],, [KEEP DELETE])
			IOEXIT=[exit-name (exit-name [,parameter,...])] [\$process-name]

Label	Statement	Parameters
	<p>Note: For transfers from Microsoft Windows to HP NonStop, use single quotation marks around each SET statement. Omit the parentheses and enclose the entire SYSOPTS statement in double quotation marks.</p>	<pre> 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")" ["SET ALTFILE key-file-number ,filename"] ["SET [NO] ALTCREATE"] ["SET FAST.LOAD.PRI priority"] ["SET FAST.LOAD.CPU cpu-number"] ["SET FAST.LOAD.SORTED Y"] ["SET XLATE ON YES OFF NO table-name"])) </pre>
		CKPT=nK nM
		<pre> COMPRESS [[PRIMEchar=X'xx' X'20' C'c'] EXTended] </pre> <p>Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.</p>

Connect:Direct for HP NonStop Copy Statement for OSS Disk and Tape Files

The COPY statement allows you to copy files from one node to another.

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

To copy from one IBM Connect:Direct platform to another, refer to the appropriate COPY FROM and TO pages for those platforms. For example, if the source file is located on a Connect:Direct for z/OS node, refer to the Connect:Direct for z/OS COPY FROM information. If the destination node is Connect:Direct for HP NonStop, refer to the TO information for this platform.

Note: Use SYSOPTS parameters, not DCB, when creating HP NonStop files, as not all z/OS DCB statements have an exact correlation to HP NonStop file attributes.

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.

The following is the Connect:Direct for HP NonStop Copy statement for Guardian Disk and Tape Files. See the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

Label	Statement	Parameters	
stepname	COPY	FROM	(
			DSN=filename FILE=filename
			PNODE SNODE
			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"] ["SET OPENFILEXMT Y"]
)
		TO	(
			DSN=filename FILE=filename
			PNODE SNODE
			DCB=([BLKSIZE=no.-bytes] [,DSORG=[U 0] [R 1] [E 2] [K 3] [,KEYLEN=no.-bytes] [,LRECL=no.-bytes] [,RECFM=record-format]
			TYPE=typekey
			DISP=([NEW OLD MOD RPL SHR] ,, [KEEP DELETE])
			IOEXIT=[exit-name (exit-name [,parameter,...])] [\$process-name]

Label	Statement	Parameters
	<p>Note: For transfers from Microsoft Windows to HP NonStop, use single quotation marks around each SET statement. Omit the parentheses and enclose the entire SYSOPTS statement in double quotation marks.</p>	<pre> 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")"] ["SET ALTFILE key-file-number ,filename" ["SET [NO] ALTCREATE" ["SET FAST.LOAD.PRI priority" ["SET FAST.LOAD.CPU cpu-number" ["SET FAST.LOAD.SORTED Y" ["SET XLATE ON YES OFF NO table-name") </pre>
		<pre> CKPT=nK nM </pre>
		<pre> COMPRESS [[PRIMEchar=X'xx' X'20' C'c'] EXTended] </pre> <p>Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.</p>

Connect:Direct for HP NonStop Run Job Statement

While Connect:Direct for HP NonStop cannot execute the RUN JOB statement, it supports a RUN JOB statement executed on the Connect:Direct for z/OS and Connect:Direct for VSE nodes (RUN JOB SNODE). See the RUN JOB Statement syntax for the appropriate platform.

Connect:Direct for HP NonStop Run Task Statement

The RUN TASK statement executes user-written or system programs during a Process. With a RUN TASK, the Process waits until the program completes before executing the next step in the Process.

Connect:Direct for HP NonStop issues a return code of zero if the program starts successfully. If the program does not start successfully, IBM Connect:Direct generates a failure return code. Once the external program begins, IBM Connect:Direct has no control of it and no knowledge of the outcome.

You can specify a list of parameters for the program in the RUN TASK statement. The program can run at either the PNODE or SNODE.

The RUN TASK statistics log records the program name, HP NonStop process id (PID), and the date and time the program began

The following is the Connect:Direct for HP NonStop Run Task statement format. See the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN TASK	(PGM=program-name)
		SYSOPTS=("[/run-option parameters/] [program-parameter] [program-parameter...]")
		TIMEOUT=n
		PNODE SNODE

Connect:Direct for HP NonStop Submit Statement

The SUBMIT statement submits another Process to either the PNODE or to the SNODE during Process execution. The submitted Process must reside on the node where the SUBMIT statement will execute. This node is referred to as the SUBNODE.

The following is the Connect:Direct for HP NonStop Submit statement format. See the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	SUBMIT	DSN=filename FILE=filename
		NEWNAME=new-name
		SUBNODE=PNODE SNODE
		SNODE=secondary-node-name
		SNODEID=(id [,pswd] [,newpswd])
		SACCT='snode-accounting-data'
		PNODEID=(id [,pswd])
		PACCT='pnode-accounting-data'
		CLASS=n
		HOLD=Yes No Call
		PRTY=n

Label	Statement	Parameters
		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

Connect:Direct for HP NonStop Conditional Statements

Conditional statements alter Process execution sequence based on the completion of the previous step in the Process.

The following is the Connect:Direct for HP NonStop Conditional statements format. See the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

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

Connect:Direct for HP NonStop Symbol Statement

The Symbol statement creates symbolic substitution values.

The following is the Connect:Direct for HP NonStop Symbol statement format. See the [Sterling Connect:Direct for HP NonStop Process Parameters](#) for more information.

Label	Statement	Parameters
optional	SYMBOL	&symbolic_name=variable-string

Connect:Direct for HP NonStop Process 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 IBM Connect:Direct Process or Submit command, CLASS defaults to the default class specified in the PARSESS parameter of the adjacent node network map record.

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. IBM 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 for HP NonStop supports checkpointing for entry sequenced files, key sequenced files, unstructured files (but not unstructured code 101), and relative files. Connect:Direct for HP NonStop does not support checkpointing for edit files, spool files, or file copies using the Fastload option.

Specifying CKPT=0K in the Copy statement overrides any checkpointing values specified in the initialization parameters; checkpointing will not occur.

COMPRESS [[PRIMEchar = X'xx' | X'20' | C'c'] | EXTended]

specifies to compress the data, which reduces the amount of data transmitted. The file is automatically decompressed at the destination node. 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 you specify compression, IBM Connect:Direct 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 1 byte.
- Repetitive occurrences (ranging from 3-63) of any other character will be compressed to 2 bytes.

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

Note: If the SNODE is B2B Integrator, only COMPRESS EXT is supported.

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.

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.

COPY

identifies the Copy statement. This statement identifier is specified with either the FROM or TO parameter, whichever is specified first.

CLASS = n

determines the node-to-node session where 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 Process or Submit command, CLASS defaults to the default class specified in the PARSESS parameter of the adjacent node network map record.

DSN = filename | FILE = filename

specifies the name of the file containing the Process.

EIF

is required for specifying the end of the IF THEN or IF THEN ELSE block of statements. There are no parameters for this statement.

ELSE

designates a block of IBM Connect:Direct statements that execute when the IF THEN condition is not satisfied. There are no parameters for this statement.

EXIT

bypasses all remaining steps within a Process. There are no parameters for this statement.

FROM

specifies the source file characteristics.

(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 for HP NonStop 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, specify the following in the FROM clause of the Connect:Direct for HP NonStop 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 HP NonStop 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

(FROM) DCB = ([,LRECL = no.-bytes])

specifies the length in bytes of the record when sending a binary OSS file (ranging from 1-32764). This value overrides the default record length (4096 bytes) for binary files.

(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. KEEP is the default.
- **DELETE** specifies that the system will delete the file after the Process step has been successfully completed.

(FROM) DSN | FILE

specifies the source file name. File names are verified based on HP NonStop standard file name conventions. DSN or FILE is invalid if you specify the IOEXIT parameter.

(FROM) SYSOPTS = [“SET XLATE ON | YES | OFF | NO | table-name”] [“SET OPENFILEXMT Y”]

specifies whether the file being transferred is converted from either ASCII to EBCDIC or EBCDIC to ASCII during Guardian disk and tape file copies.

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 for HP NonStop will check the XLFILE for a table named DEFAULT. If the DEFAULT table is not in XLFILE, then Connect:Direct for HP NonStop uses the standard English language ASCII/EBCDIC table as defined by HP NonStop.

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 *IBM Sterling Connect:Direct for HP NonStop Installation Guide* and the *IBM Sterling Connect:Direct for HP NonStop Administration Guide* for further details on user-defined translation tables.

SET OPENFILEXMT allows the transfer of open source files. This SYSOPTS parameter overrides the SENDOPENFILE setting in the NDMINIT file. If the source file is being written to by another process and SETOPENFILEXMT Y is specified in the SYSOPTS, when Connect:Direct for HP NonStop reaches the current end of the data, it waits for and transfers any additional data until the creating process completes and closes the file.

(FROM) SYSOPTS = [“SET XLATE ON | YES | OFF | NO | table-name”] [“SET DATATYPE ASCII | BINARY”]

specifies system operation parameters on the Copy statement. These parameters can be used to specify whether to convert files from either ASCII to EBCDIC or EBCDIC to ASCII during OSS disk and tape file copies. The SET OPENFILEXMT Y parameter can be used to allow data to be transferred while it is being written to the source file.

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 for HP NonStop will check the XLFILE for a table named DEFAULT. If the DEFAULT table is not in XLFILE, then Connect:Direct for HP NonStop uses the standard English language ASCII/EBCDIC table as defined by HP NonStop.

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 *IBM Sterling Connect:Direct for HP NonStop Installation Guide* and the *IBM Sterling Connect:Direct for HP NonStop Administration Guide* for further details on user-defined translation tables.

SET DATATYPE specifies whether the source file being transferred is in ASCII or BINARY format.

Note: To successfully transfer binary files from an OSS files system to either UNIX or Microsoft Windows, you must specify BINARY as the datatype because UNIX and Microsoft Windows expect text (ASCII) files.

(FROM) SYSOPTS =(["SET SPOOLER \$spooler-name"] ["SET SPOOLNUM job-number"])

specifies system operation parameters on a spooler file copy. It is an alternative way of specifying file creation attributes.

Each system operation is indicated as a HP NonStop 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 HP NonStop file attributes or special processing instructions for Connect:Direct for HP NonStop. 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 IBM 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 HP NonStop 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 IBM 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 IBM Connect:Direct system accesses the job with the highest job number that is owned by any user in the SUPER group.

If the HP NonStop 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 IBM Connect:Direct system accesses the job with the highest number. If one or more of the jobs are owned by SUPER.SUPER, the IBM Connect:Direct system accesses the job with the highest job number that is owned by SUPER.SUPER.

GOTO

moves to a specific step within a Process. See the following Field Descriptions section for details on using the step label.

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. A Process submitted with HOLD=Yes is placed on the Hold queue even if a start time is specified.

No specifies that the Process is to execute as soon as possible. This 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.

IF THEN

specifies that the IBM Connect:Direct system executes a block of IBM 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.

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. The parameters are only valid if the SNODE is a Connect:Direct for HP NonStop 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.

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.

User-defined labels must begin in column one. Labels are 1-8 character alphanumeric strings. The first character must be alphabetic.

NEWNAME = new-name

specifies a new name for the Process. The default value is the label on the Process statement.

nn

specifies the numeric value to be used for completion code checking. If specified as X'nn', it is a hexadecimal value; any other specification 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.

PACCT = 'pnode-accounting-data'

specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. Enclose the string in single quotation marks if it contains special characters.

PACCT is used only for documentation purposes.

PGM = program-name

specifies the name of the object file to be executed.

PNODE

On the Copy statement, specifies the transfer direction. When PNODE is specified on the FROM parameter, the file to be copied resides on the primary node. When PNODE is specified in the TO parameter, the file is sent to the primary node. PNODE is the default for the FROM parameter.

On the Run Task statement, specifies that the program will be executed on the PNODE. PNODE is the default.

PNODE = primary-node-name

specifies the 1-16 character name of the primary node (PNODE) used in the Process. The name can be expressed in alphanumeric or nationals (@ # \$), with embedded periods. The Process is always submitted to 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 IBM Connect:Direct node.

id specifies the HP NonStop 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).

PROcEss

identifies the statement. This statement can be abbreviated to PROC.

process name

specifies the 1-8 character name of the Process. The Process name is required. The first character must be alphabetic and must start in column one. The PROCESS keyword must be on the same line as the Process name.

This label identifies the Process in messages or statistics.

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 IBM Connect:Direct installation. See the Connect:Direct for HP NonStop installation and administration guides 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 specified. 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 executed every time the IBM Connect:Direct system is initialized. Processes submitted with RETAIN=INITIAL do not execute when initially submitted. Do not specify STARTT with RETAIN=INITIAL.

RUN TASK

identifies the Run Task statement.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. Enclose the string in single quotation marks if it contains special characters.

This parameter is ignored when Connect:Direct for i5/OS is the SNODE.

SNODE

On the Copy statement, this parameter specifies the secondary node and defines the direction of transfer. When SNODE is specified with the FROM parameter, the file to be copied resides on the secondary node.

When SNODE is specified with the TO parameter, the file is sent to the secondary node. SNODE is the default with the TO parameter.

On the Run Task statement, this parameter specifies that the program will be executed on the SNODE, which is the destination node.

SNODE = secondary-node-name

specifies the 1-16 character alphanumeric name of the secondary node (SNODE) used in this Process. The name can be expressed in alphanumerics or nationals (@ # \$), with embedded periods. This parameter is required.

This is the logical node name defined as the adjacent node in the network map.

This parameter is not required if SNODE is specified on the Submit command.

When specified in the Submit statement, this parameter overrides the value specified in the Process statement.

The PNODE and SNODE can specify the same symbolic node name.

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

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

For Connect:Direct for i5/OS, if an SNODEID and password of the Process submitter is not specified in the Process statement, the user ID and password of the Process submitter is used for the security ID and password check by Connect:Direct for i5/OS.

For Connect:Direct for UNIX, the security user ID and passwords are case sensitive.

id specifies the HP NonStop group number and user number. These numbers can range 0-255 and are separated by a period (.). Other operating environments limit the ID to 1-8 alphanumeric characters.

For Connect:Direct for i5/OS, this subparameter specifies the i5/OS user profile used for authorization checks during Process execution and is limited to 8 characters.

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.

z/OS, VM, and VSE nodes only recognize passwords specified in uppercase alphanumeric characters. See the Connect:Direct for HP NonStop installation and administration guides for a solution for mixed-case passwords.

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 for HP NonStop, SAFEGUARD must be running on HP NonStop.

This subparameter is ignored for Connect:Direct for i5/OS.

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

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

date specifies that the Process executes on the indicated date. Specify the date in one of the following formats:

yymmdd	mm/dd/yy	yy/mm/dd	mm.dd.yy
yy.mm.dd	mmddy	yyddd (Julian date)	yy/ddd (Julian date)

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

If RETAIN=YES, do not specify a date in the STARTT parameter.

day specifies the day of the week that the Process is 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 you specify the day of the week with RETAIN=YES, the Process executes the same day every week. If you only specify a day, 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 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, specify 01:00:00 hours as 1:00AM, and specify 13:00 hours as 1:00PM.

If you specify hh:mm:ssXM with RETAIN=YES, the Process executes at the same time every day. You do not need to specify minutes and seconds.

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

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

Do not specify STARTT with RETAIN=I.

stepname

specifies the user-defined name of the Copy, Run Task, or Submit step.

Stepnames must begin in column one. Stepnames are 1-8 character alphanumeric strings. The first character must be alphabetic.

SUBMIT

identifies the Submit statement.

SUBNODE = PNODE | SNODE

specifies the node where the Process defined in the Submit statement will execute. PNODE means that the Process is submitted on the node that has Process control. 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.

SYMBOL

identifies the Symbol statement.

&symbolic_name = variable-string

specifies a string that is substituted into the Process.

When IBM Connect:Direct software encounters an ampersand (&) plus 1-17 alphanumeric characters, IBM 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 IBM 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. Enclose a symbolic parameter string containing special characters in single quotation marks.

&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. Enclose a symbolic parameter string containing special characters in single quotation marks.

When specified on the Submit statement, this parameter override the value in the Process statement.

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.

SYSOPTS = ("[/run-option parameters/] [program-parameter] [program-parameter...])

specifies the parameters passed to the new HP NonStop Process when it is created. Enclose the SYSOPTS string in either single or double quotation marks. Enclose any literal parameter values to be passed in single quotation marks. Enclose any symbolic values (&value) in double quotation marks.

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 HP NonStop process will execute. Values range from 0-15 inclusive.

IN file specifies the name of the file containing the HP NonStop program parameters.

INSPECT [OFF | ON | SAVEABEND] specifies the debugging environment for the HP NonStop 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 ends abnormally. 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 HP NonStop process. Values range from 1-64 inclusive.

NAME specifies the name of the new HP NonStop process, where name is a 1-5 character alphanumeric string with the first character alphabetic.

If the new HP NonStop 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 HP NonStop process. Values range from 1-199 inclusive.

If this parameter is not specified, a priority is assigned by the program that creates the new HP NonStop process.

SWAP filename specifies the name of the file that will hold the virtual data for the new HP NonStop process. This parameter is for debugging purposes only.

TERM \$termname specifies the home terminal for the new HP NonStop 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.

THEN

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

TIMEOUT=n

specifies the number of minutes to attempt Process execution before the session is cancelled. The range is 1-32768 minutes (22 days and 18 hours).

TO

specifies the destination file characteristics.

(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 HP NonStop 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 HP NonStop node defaults to an unstructured file, code 0.

For UNIX to HP NonStop copies: When copying files from UNIX to HP NonStop, use the DCB parameter to allocate destination files. Specify 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 HP NonStop 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.

(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 HP NonStop 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".

When using checkpointing, DISP=MOD is required.

- **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. IBM 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. IBM 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. This is the default.
- **DELETE** specifies the system will delete the file when the Process step is terminated abnormally.

(TO) DSN | FILE

specifies the destination file name. DSN or FILE is invalid if you specify the IOEXIT parameter.

(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"]
)

```

specifies system operation parameters on the Copy statement. It is an alternative way of specifying file creation attributes. You can use HP NonStop File Utility Program (FUP) syntax to create HP NonStop-specific file options that are available through Sterling IBM Connect:Direct syntax.

SYSOPTS are expressed as HP NonStop SET commands. There are two ways to express multiple SET command parameter:

- SET precedes each parameter. For example:

```
SYSOPTS=("SET parameter" "SET parameter" "SET parameter")
```

- SET precedes the first parameter, and commas separate subsequent parameters. For example:

```
SYSOPTS=("SET parameter, parameter, parameter")
```

Enclose each sysopts string in double quotation marks except when copying from Microsoft Windows to HP NonStop. For copies from Microsoft Windows to HP NonStop, enclose each SET parameter in single quotation marks and enclose the entire SYSOPTS string in double quotation marks. For example:

```
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 details on listed FUP SET commands, refer to the appropriate HP NonStop manual. The following commands are Connect:Direct for HP NonStop SET commands and are described only in the documentation set for Connect:Direct for HP NonStop: 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 HP NonStop file attributes or special processing instructions for Connect:Direct for HP NonStop. 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 IBM 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 HP NonStop node is the destination. With FASTLOAD, the Connect:Direct for HP NonStop 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 HP NonStop files, when receiving, and the file is being created NEW. It is not used in a non-HP NonStop data set definition, when sending a file from HP NonStop, or when overwriting an existing file on a HP NonStop 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 for HP NonStop will check the XLFILE for a table named DEFAULT. If the DEFAULT table is not in XLFILE, then Connect:Direct for HP NonStop will use the standard English language ASCII/EBCDIC table as defined by HP NonStop.

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 *IBM Sterling Connect:Direct for HP NonStop Administration Guide* for further details on user-defined translation tables.

(TO) SYSOPTS = “SET XLATE ON | YES | OFF | NO | table-name” [“SET DATATYPE ASCII | BINARY”]

specifies system operation parameters on OSS disk and tape file copies. It is an alternative way of specifying file creation attributes.

You can use HP NonStop File Utility Program (FUP) syntax to create HP NonStop-specific file options that are available through IBM Connect:Direct syntax.

SYSOPTS are expressed as HP NonStop SET commands. There are two ways to express multiple SET command parameter:

- SET precedes each parameter. For example:

```
SYSOPTS=("SET parameter" "SET parameter" "SET parameter")
```

- SET precedes the first parameter, and commas separate subsequent parameters. For example:

```
SYSOPTS=("SET parameter, parameter, parameter")
```

Enclose each sysopts string in double quotation marks except when copying from Microsoft Windows to HP NonStop. For copies from Microsoft Windows to HP NonStop, enclose each SET parameter in single quotation marks and enclose the entire SYSOPTS string in double quotation marks. For example:

```
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 details on listed FUP SET commands, refer to the appropriate HP NonStop manual. The following commands are Connect:Direct for HP NonStop SET commands and are described only in the documentation set for Connect:Direct for HP NonStop: 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 HP NonStop file attributes or special processing instructions for Connect:Direct for HP NonStop. The SET commands follow in alphabetical order.

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 for HP NonStop will check the XLFILE for a table named DEFAULT. If the DEFAULT table is not in XLFILE, then Connect:Direct for HP NonStop will use the standard English language ASCII/EBCDIC table as defined by HP NonStop.

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 *IBM Sterling Connect:Direct for HP NonStop Administration Guide* for further details on user-defined translation tables.

SET DATATYPE specifies whether the source file being transferred is in ASCII or BINARY format.

To successfully transfer binary files from an OSS files system to either UNIX or Microsoft Windows, specify BINARY as the datatype.

(TO) SYSOPTS = [“SET SPOOLER \$spooler-name”]

specifies system operation parameters on spooler job copies. It is an alternative way of specifying file creation attributes.

SET parameter commands define HP NonStop file attributes or special processing instructions for Connect:Direct for HP NonStop.

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 IBM Connect:Direct system accesses the job with the highest job number corresponding to the given filename in the READY state.

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.

i5/OS Statements and Parameters

Connect:Direct for i5/OS Process Statements

Connect:Direct for i5/OS uses CL commands instead of the IBM Connect:Direct Process language to initiate Processes using LU6.2 or TCP/IP sessions to remote IBM Connect:Direct nodes. The Connect:Direct for i5/OS statements are used in Processes where Connect:Direct for i5/OS is the remote node.

Connect:Direct for z/OS, VSE, VM, and HP NonStop users can submit Processes to connect to Connect:Direct for i5/OS using SNA LU0 sessions. Connect:Direct for z/OS, VM, UNIX, VSE, and Microsoft Windows users can submit IBM Connect:Direct Processes to initiate SNA LU6.2 independent or dependent LU sessions. Connect:Direct for z/OS, VSE, OpenVMS, HP NonStop, UNIX, and Microsoft Windows users can submit Processes to connect to Connect:Direct for i5/OS 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 z/OS, you will use the RUN JOB statement for Connect:Direct for z/OS.

Connect:Direct for i5/OS CL commands are used to initiate connections to remote nodes running IBM Connect:Direct using LU6.2 sessions or TCP/IP. See the *IBM Sterling Connect:Direct for i5/OS User's Guide* for a description of CL commands, their syntax and parameters.

Connect:Direct for i5/OS Copy Files Statement

The Connect:Direct for i5/OS COPY statement is used in a Process initiated on a non-i5/OS node to copy files, members, objects, and spooled files to or from an i5/OS node.

The COPY statement consists of a COPY FROM clause that specifies the source object name and a COPY TO clause that specifies the destination object name. You can specify additional parameters to customize the file transfer operation.

Use the file format to copy a physical database file to and from a Connect:Direct for i5/OS node.

Refer to [Connect:Direct for i5/OS File Support](#) for guidelines when copying files in Connect:Direct for i5/OS.

Note: z/OS member naming conventions can have unexpected results when you copy a file from i5/OS to z/OS. Refer to [Member Name Length in Connect:Direct for i5/OS Copy to z/OS](#) for more information.

The PROCESS statement defines the attributes of a Process and is always the first statement in a Process.

The following is the Connect:Direct for i5/OS Copy-Files statement format. Refer to [Connect:Direct for i5/OS Process Parameters](#) for more information.

Label	Statement	Parameters	
stepname	COPY	FROM	(
			DSN=`library-name/file-name' `/directory/file-name' `/QLANSrv/file-name' `/QDLS/folder-name' `/QOpenSys/file-name'
			SYSOPTS="TYPE(FILE)
			PRECMPR (*YES *NO)
			EXITCMD(valid i5/OS command)
			FAILCMD(valid i5/OS command)
			SNDFFD(*YES *NO)
			TEXTFILE(*YES *NO)
			EORCHAR(xxxx)
			CCSID(nnnnn)
			CODEPAGE(from code set, to Unicode set)
			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)

Label	Statement	Parameters
		SYSOPTS="TYPE(FILE) DECMPR (*YES *NO) EXITCMD(valid i5/OS command) FAILCMD(valid i5/OS command) TEXTFILE(*YES *NO) EORCHAR(xxxx) CCSID(nnnnn) CODEPAGE(from Unicode set, to code set) 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=(<code>[NEW OLD MOD RPL SHR]</code>)
		UNIT=(unit-identifier)
)
		CKPTINV= <code>[nnnnnnn nnnnnnK nnnnnnM]</code>
		COMPRESS [<code>[PRIMEchar =X`xx' X`40' C`cc'] </code> EXTENDED = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n) Note: If the SNODE is B2B Integrator, only COMPRESS EXT is supported.

Connect:Direct for i5/OS Copy Members Statement

The Connect:Direct for i5/OS COPY statement is used in a Process initiated on a non-i5/OS node to copy files, members, objects, and spooled files to or from an i5/OS node.

The COPY statement consists of a COPY FROM clause that specifies the source object name and a COPY TO clause that specifies the destination object name. You can specify additional parameters to customize the file transfer operation.

Use this format to copy a physical database file member to and from a Connect:Direct for i5/OS node.

The following is the Connect:Direct for i5/OS Members statement format. Refer to [Connect:Direct for i5/OS Process Parameters](#) for more information.

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'

Label	Statement	Parameters
		SYSOPTS="TYPE(MBR) PRECMR (*YES *NO) EXITCMD(valid i5/OS command) FAILCMD(valid i5/OS command) SNDDFD(*YES *NO) TEXTFILE(*YES *NO) CODEPAGE(from code set, to Unicode set) 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
		SYSOPTS="TYPE(MBR) DECMR (*YES *NO) EXITCMD(valid i5/OS command) FAILCMD(valid i5/OS command) TEXTFILE(*YES *NO) CODEPAGE(from Unicode set, to code set) 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) IGCDA(*YES *NO)"
		DISP=([NEW OLD MOD RPL SHR])
		UNIT=(unit-identifier)
)
		CKPTINV=[nnnnnnn nnnnnnK nnnnnnM]

Label	Statement	Parameters
		COMPRESS [[PRIMEchar = X`xx' X`40' C`cc'] EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n) Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

Connect:Direct for i5/OS Copy Objects Statement

The Connect:Direct for i5/OS COPY statement is used in a Process initiated on a non-i5/OS node to copy files, members, objects, and spooled files to or from an i5/OS node.

The COPY statement consists of a COPY FROM clause that specifies the source object name and a COPY TO clause that specifies the destination object name. You can specify additional parameters to customize the file transfer operation.

Use this format to copy one or more objects to and from a Connect:Direct for i5/OS node. The object must be in save file format on the i5/OS prior to copying. You can view the save file information by displaying the save files on the i5/OS.

The following is the Connect:Direct for i5/OS Copy Objects statement format. Refer to [Connect:Direct for i5/OS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	COPY	FROM
		(
		DSN= ` library-name/save-file-name'
		SNODE
		SYSOPTS="TYPE(OBJ)
		EXITCMD(valid i5/OS command)
		FAILCMD(valid i5/OS command)
		DISP=([SHR OLD] ,[KEEP DELETE] ,[KEEP DELETE])
)
		TO
		(
		DSN= ` library-name/save-file-name'
		SNODE
		SYSOPTS="TYPE(OBJ)
		EXITCMD(valid i5/OS command)
		FAILCMD(valid i5/OS command)
		MAXRCDS(number *NOMAX)
		ASP(auxiliary-storage-pool)
		TEXT(` text-description')
		AUT(*EXCLUDE *CHANGE *ALL *USE)"
		DISP=([NEW OLD MOD RPL SHR])
)

Label	Statement	Parameters
		COMPRESS [[PRIMEchar = X`xx' X`40' C`cc'] EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n) Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

Connect:Direct for i5/OS Copy Spooled Files Statement

The Connect:Direct for i5/OS COPY statement is used in a Process initiated on a non-i5/OS node to copy files, members, objects, and spooled files to or from an i5/OS node.

This format is used to copy to a spooled file on a Connect:Direct for i5/OS node. There is no FROM clause.

The following is the Connect:Direct for i5/OS Copy-Spooled Files statement format. Refer to [Connect:Direct for i5/OS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	COPY	FROM
		(
		DSN= ` library-name/save-file-name'
		SNODE
		SYSOPTS="TYPE(OBJ)
		EXITCMD(valid i5/OS command)
		FAILCMD(valid i5/OS command)
		DISP=([SHR OLD] ,[KEEP DELETE] ,[KEEP DELETE])
)
		TO
		(
		DSN= ` library-name/save-file-name'
		SNODE
		SYSOPTS="TYPE(OBJ)
		EXITCMD(valid i5/OS command)
		FAILCMD(valid i5/OS command)
		MAXRCDS(number *NOMAX)
		ASP(auxiliary-storage-pool)
		TEXT(` text-description')
		AUT(*EXCLUDE *CHANGE *ALL *USE)"
		DISP=([NEW OLD MOD RPL SHR])
)

Label	Statement	Parameters
		COMPRESS [[PRIMEchar = X`xx' X`40' C`cc'] EXTended = ECCLEVEL(n), ECWINSIZE(n), ECMEMLEVEL(n) Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

Connect:Direct for i5/OS 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 for i5/OS node, you can specify whether:

- To send an entire file.
- To send one specific file member.
- To select certain members for transmission (with the SELECT parameter).
- To exclude certain members from transmission (with the EXCLUDE parameter).
- To rename a member (with the NEWNAME parameter).
- To replace existing members of the same name at the receiving node (with the REPLACE and NOREPLACE parameters, or the R and NR subparameters of the SELECT parameter).

Use the following guidelines when copying a file:

- You cannot use EXCLUDE and SELECT if the FROM DSN or TO DSN contains a member name.
- The hierarchy for the SELECT and EXCLUDE parameters from highest override priority to lowest, is:
 - 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 you specify EXCLUDE and do not specify SELECT, all members not excluded are copied. If you do not specify EXCLUDE and specify SELECT, 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.

Member Name Length in Connect:Direct for i5/OS Copy to z/OS

i5/OS allows 10-character member names. z/OS only allows 8-character member names. When you copy a file from a Connect:Direct for i5/OS node to a PDS on a Connect:Direct for z/OS node, the ninth and tenth characters of the member name are dropped on z/OS.

This can have unwanted results with members whose names have 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 on z/OS. The first member name is written to z/OS. The second membrnam will overwrite the previous membrnam, unless NO REPLACE (NR) is specified.

Connect:Direct for i5/OS Run Job Statement

The Connect:Direct for i5/OS RUN JOB statement is used in a non-i5/OS Process to execute i5/OS CL commands on a remote Connect:Direct for i5/OS node. The i5/OS CL command is submitted to the i5/OS to run as a separate job through the SBMJOB command. The Process does not wait until the i5/OS command finishes before executing the next step in the Process.

The RUN JOB execution results in a return code but IBM Connect:Direct does not verify the execution of the CL commands.

The following is the Connect:Direct for i5/OS Run Job statement format. Refer to [Connect:Direct for i5/OS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN JOB	(DSN=AS400)
		SYSOPTS="cmd(CL command)"
		SNODE

Connect:Direct for i5/OS Run Task Statement

The Connect:Direct for i5/OS RUN TASK statement is used in a non-i5/OS Process to execute i5/OS CL commands on a remote Connect:Direct for i5/OS 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 for i5/OS node (SNDBRKMSG, SNDMSG, or an equivalent command)

The following is the Connect:Direct for i5/OS Run Task statement format. Refer to [Connect:Direct for i5/OS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN TASK	(PGM=AS400)
		SNODE
		SYSOPTS = "string"

Connect:Direct for i5/OS Process 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 compressed file is sent and decompressed by the receiving node during the send operation. That is, the (FROM) SYSOPTS has PRECMR(*YES) specified and (TO) SYSOPTS has DECMR(*YES) specified.
- A file is sent to a single z/OS 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, IBM 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.

Note: If the SNODE is B2B Integrator, only COMPRESS EXT is supported.

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.

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.

COPY

identifies the COPY statement.

DSN = AS400

is required when submitting a Process with RUN JOB from z/OS that executes commands on i5/OS. This parameter is used to satisfy syntax requirements on the z/OS node. In this case, both SYSOPTS and DSN are required. Specify the value for this parameter in uppercase characters. This parameter is required.

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 statement. EXCLUDE allows the user to make exceptions to members specified generically or by range in the SELECT option.

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

specifies that the subsequent parameters and subparameters define the source object characteristics. This parameter is required.

(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.

(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 i5/OS standard file name conventions. The entire DSN must be in single quotation marks. This parameter is required.

'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.

(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 i5/OS standard name conventions. The entire DSN must be in single quotation marks. This parameter is required.

'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.

(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.

(FROM) DSN = 'library-name/save-file-name'

specifies the source save file name. File names are verified based on the i5/OS 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.

(FROM) SYSOPTS = "TYPE(FILE)

PRECMR (*YES | *NO)

EXITCMD(valid i5/OS command)

FAILCMD(valid i5/OS command)

SNDFFD(*YES | *NO)

TEXTFILE(*YES | *NO)

EORCHAR(xxxx) CCSID(nnnnn)

CODEPAGE(from code set, to Unicode set)

XTRAN (table-name)

[XTRANLSO (so-code) | XTRANLSI (si-code) | XTRANLDATA (MIXED | DBCS)]"

specifies system operation parameters on the Connect:Direct for i5/OS COPY statement. The maximum number of characters for SYSOPTS is 256. This parameter is required.

Enclose all SYSOPTS parameter values in parentheses. Enclose the entire SYSOPTS string in double quotation marks. Separate subparameters with 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 or an IFS file. **File** is the default.

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 i5/OS command) specifies a command to be executed only if the copy process is successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

FAILCMD(valid i5/OS command) specifies a command to be executed only if the copy process is not successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

SNDDFD(*YES | *NO) specifies whether the sending node will transfer file field descriptions. You cannot use this parameter with the integrated file system. ***NO** is the default.

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
```

TEXTFILE(*YES | *NO) specifies whether the file being sent is a text file. Specify **TEXTFILE(YES)** if you need to add or remove end of record characters, such as CRLF. This keyword is for the i5/OS only. **NO** is the default.

EORCHAR(xxxx) is the hex value of the end of record character. Specify this parameter if the EOR is not CRLF, for example, 0D or 0D25

CCSID(nnnnn) specifies the value for the character code set if the IFS file has a specific CCS that is not the system/job default, for example, CCSID(1252). Must be used for text files.

CODEPAGE(from code set, to Unicode set) invokes code set conversion utilities, for example, from ASCII to EBCDIC would be **CODEPAGE(1252,37)**.

from code set is the name of the set of the original data set and is required.

to Unicode set is the name of the code set on the local node that will be used as the intermediate conversion format. The code set name is required and can be UTF-8 or UCS-2, or their equivalent on other operating systems. For example, 1208 is the UTF-8 equivalent on the i5/OS system.

The code set specifications are only validated for basic syntax. An invalid code set specification will produce an error message on the node attempting conversion.

A code set value of LOCALE specifies the default code set for the node performing conversion.

Note: The EORCHAR, CCSID, and CODEPAGE parameters are valid when using Connect:Direct for i5/OS Version 3.5 or later.

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 OE. 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 PRECMR(*YES) or DECMR(*NO)

When sending a file from the i5/OS, 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.

(FROM) SYSOPTS = "TYPE(MBR)

PRECMR (*YES | *NO)

EXITCMD(valid i5/OS command)

FAILCMD(valid i5/OS command)

SNDFD(*YES | *NO)

TEXTFILE(*YES | *NO)

CODEPAGE(from code set, to Unicode set)

XTRAN (table-name)

[XTRANLSO (so-code) | XTRANLSI (si-code) | XTRANLDATA (MIXED | DBCS)]

specifies system operation parameters on the Connect:Direct for i5/OS COPY statement. The maximum number of characters for SYSOPTS is 256. This parameter is required.

Enclose all SYSOPTS parameter values in parentheses. Enclose the entire SYSOPTS string in double quotation marks. Separate subparameters with blanks. For example:

```
SYSOPTS="TYPE(FILE) PRECMR(*YES) XTRAN(EBCXKSC) XTRANLDATA(MIXED)"
```

TYPE(MBR) specifies that the data being copied is a member of a physical database file.

PRECMR(*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 PRECMR(*YES) parameter with the CDSND command. The default is PRECMR(*NO).

EXITCMD(valid i5/OS command) specifies a command to be executed only if the copy process is successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

FAILCMD(valid i5/OS command) specifies a command to be executed only if the copy process is not successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

SNDFD(*YES | *NO) specifies whether the sending node will transfer file field descriptions. You cannot use this parameter with the integrated file system. **NO** is the default.

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
```

TEXTFILE(*YES | *NO) specifies that the file being sent is a text file. This keyword is for the i5/OS only. **NO** is the default.

CODEPAGE(from code set, to Unicode set) invokes code set conversion utilities, for example, from ASCII to EBCDIC would be CODEPAGE(1252,37).

from code set is the name of the set of the original data set and is required.

to Unicode set is the name of the code set on the local node that will be used as the intermediate conversion format. The code set name is required and can be UTF-8 or UCS-2, or their equivalent on other operating systems. For example, 1208 is the UTF-8 equivalent on the i5/OS system.

The code set specifications are only validated for basic syntax. An invalid code set specification will produce an error message on the node attempting conversion.

A code set value of LOCALE specifies the default code set for the node performing conversion.

Note: The CODEPAGE parameter is valid when using Connect:Direct for i5/OS Version 3.5 or later.

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 **PRECMR(*YES)** or **DECMR(*NO)**

When sending a member from the i5/OS, 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.

(FROM) SYSOPTS = "TYPE(OBJ)"

EXITCMD(valid i5/OS command)

FAILCMD(valid i5/OS command)

specifies system operation parameters on the Connect:Direct for i5/OS COPY statement. The maximum number of characters permitted for SYSOPTS is 256. This parameter is required.

Enclose all SYSOPTS parameter values in parentheses. Enclose the entire SYSOPTS string in double quotation marks. Separate subparameters with blanks. For example:

```
SYSOPTS="TYPE(OBJ) EXITCMD(command) FAILCMD(command)"
```


TYPE(OBJ) specifies that the object being copied is to be transferred in save file format. This parameter is required.

EXITCMD(valid i5/OS command) specifies a command to be executed only if the copy process is successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

FAILCMD(valid i5/OS command) specifies a command to be executed only if the copy process is not successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

Label

IBM 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.

PGM = AS400

is used to satisfy syntax checking. This parameter is required.

RUN JOB

identifies the RUN JOB statement.

RUN TASK

identifies the RUN TASK statement.

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.

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.

SNODE

On a Copy statement, specifies the secondary node. The Connect:Direct for i5/OS node is always the SNODE. The SNODE is the default and is included in the Process as documentation only.

On a Run Job or Run Task statement, specifies that the commands are executed on the SNODE.

SYSOPTS = "cmd(CL command)"

specifies system-specific submission information. For a remote i5/OS system, this parameter specifies a string 'cmd(any valid batch CL command)'. This CL command will be submitted on the remote i5/OS system through a SBMJOB command. This parameter is required.

SYSOPTS = "string"

allows you to specify batch-capable i5/OS CL commands and parameters in the statement. The maximum number of permitted characters is 256. This parameter is required.

A CMD(CL command) specifies that a CL command is to be executed by the Process. An arbitrary number of CMD() parameters can be specified. Any batch-capable CL command the user is authorized to issue may be specified.

TO

specifies that the subsequent parameters and subparameters define the destination file characteristics. This parameter is required.

(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.

TO

specifies that the subsequent parameters and subparameters define the destination file characteristics. This parameter is required.

(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 i5/OS standard file name conventions. The entire DSN must be in single quotation marks. This parameter is required.

'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.

(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 i5/OS standard file name conventions. The entire DSN must be in single quotation marks. This parameter is required.

'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.

(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.

(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. Enclose the name in single quotation marks if it contains special characters.

**(TO) SYSOPTS = "TYPE(FILE)
DECMPR (*YES | *NO)
EXITCMD(valid i5/OS command)
FAILCMD(valid i5/OS command)
TEXTFILE(*YES | *NO)
EORCHAR(xxxx)
CCSID(nnnnn)
CODEPAGE(from Unicode set, to code set)
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 for i5/OS COPY statement. The maximum number of characters permitted for SYSOPTS is 256. This parameter is required.

Enclose the entire SYSOPTS string in double quotation marks. Enclose each subparameter string in parentheses. Separate the subparameters by blanks. For example:

```
SYSOPTS="TYPE(FILE) DECMPR(*YES)"
```

TYPE(FILE) specifies that the data is to be copied to the Connect:Direct for i5/OS node as a physical database file or an IFS file. This parameter is required.

DECMPR(*YES | *NO) specifies whether the receiving IBM Connect:Direct node is to decompress the received file. This parameter is valid only when the receiving system is a Connect:Direct for i5/OS node. ***NO** instructs the receiving system to place the receiving data in a database file without decompressing it. ***NO** is the default.

EXITCMD(valid i5/OS command) specifies a command to be executed only if the copy process is successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

FAILCMD(valid i5/OS command) specifies a command to be executed only if the copy process is not successful.

TEXTFILE(*YES | *NO) specifies that the file being received is a text file. This keyword is for the i5/OS only. ***NO** is the default.

EORCHAR(xxxx) is the hex value of the end of record character. Specify this parameter if the EOR is not CRLF, for example, 0D or 0D25

CCSID(nnnnn) specifies the value for the character code set if the IFS file has a specific CCS that is not the system/job default, for example, CCSID(1252). Must be used for text files.

CODEPAGE(from Unicode set, to code set) invokes code set conversion utilities, for example, from ASCII to EBCDIC would be CODEPAGE(1252,37).

- **from Unicode set** is the name of the Unicode set of the encoded data sent to the receiving node. The code set name is required and can be UTF-8 or UCS-2, or their equivalent on other operating systems. For example, 1208 is the UTF-8 equivalent on the i5/OS system.
- **to code set** is the name of the final code set that will be used on the remote node. You can use LOCALE to indicate the default code page set relevant to the receiving node.

The code set specifications are only validated for basic syntax. An invalid code set specification will produce an error message on the node attempting conversion.

A code set value of LOCALE specifies the default code set for the node performing conversion.

Note: The EORCHAR, CCSID, and CODEPAGE parameters are valid when using Connect:Direct for i5/OS Version 3.5 or later.

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 IBM 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. ***SRC** is the default.
- ***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 i5/OS command line.

MAXMBS(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(*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. *CHANGE is the default.
- ***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 i5/OS security, refer to the IBM i5/OS 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 i5/OS system and the file being sent contains DBCS data. This parameter is only valid if the destination i5/OS system is configured for DBCS support. Including this parameter in a Process that sends a file to an i5/OS system without DBCS support will cause an error and terminate the Process.

(TO) SYSOPTS = "TYPE(MBR)
DECMR(*YES | *NO)
EXITCMD(valid i5/OS command)
FAILCMD(valid i5/OS command)
TEXTFILE(*YES | *NO)
RCDLEN(record-length)
FILETYPE(*SRC | *DATA)
TEXT('text-description')
CODEPAGE(from Unicode set, to code set)

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 for i5/OS COPY statement. The maximum number of characters permitted for SYSOPTS is 256. This parameter is required.

Enclose the entire SYSOPTS string in double quotation marks. Enclose each subparameter string in parentheses. Separate the subparameters by blanks. For example:

```
SYSOPTS="TYPE(MBR) DECMPR(*YES) "
```

TYPE(MBR) specifies that the data is to be copied to the Connect:Direct for i5/OS node as a member of a physical database file. This parameter is required.

DECMPR(*YES | *NO) specifies whether the receiving IBM Connect:Direct node is to decompress the received file. This parameter is valid only when the receiving system is a Connect:Direct for i5/OS node.

*NO instructs the receiving system to place the receiving data in a database file without decompressing it. *NO is the default.

EXITCMD(valid i5/OS command) specifies a command to be executed only if the copy process is successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

FAILCMD(valid i5/OS command) specifies a command to be executed only if the copy process is not successful. This parameter is only valid when the sender is running Connect:Direct for i5/OS Version 3.3 or later.

TEXTFILE(*YES | *NO) specifies that the file being received is a text file. This keyword is for the i5/OS only. *NO is the default.

CODEPAGE(from Unicode set, to code set) invokes code set conversion utilities, for example, from ASCII to EBCDIC would be CODEPAGE(1252,37).

from Unicode set is the name of the Unicode set of the encoded data sent to the receiving node. The code set name is required and can be UTF-8 or UCS-2, or their equivalent on other operating systems. For example, 1208 is the UTF-8 equivalent on the i5/OS system.

to code set is the name of the final code set that will be used on the remote node. You can use LOCALE to indicate the default code page set relevant to the receiving node.

The code set specifications are only validated for basic syntax. An invalid code set specification will produce an error message on the node attempting conversion.

A code set value of LOCALE specifies the default code set for the node performing conversion.

Note: The CODEPAGE parameter is valid when using Connect:Direct for i5/OS Version 3.5 or later.

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 IBM 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. *SRC is the default.
- ***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 i5/OS 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. *CHANGE is the default.
- ***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 i5/OS security, refer to the IBM i5/OS 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 i5/OS system and the file being sent contains DBCS data. This parameter is only valid if the destination i5/OS system is configured for DBCS support. Including this parameter in a Process that sends a file to an i5/OS system without DBCS support will cause an error and terminate the Process.

**(TO) SYSOPTS = "TYPE(OBJ)
EXITCMD(valid i5/OS command)
FAILCMD(valid i5/OS command)
MAXRCDS(number | *NOMAX),
ASP(auxiliary-storage-pool),
TEXT('text description'),**

AUT(*EXCLUDE | *CHANGE | *ALL | *USE)"

specifies system operation parameters on the IBM Connect:Direct i5/OS COPY statement. The maximum number of characters permitted for SYSOPTS is 256. This parameter is required.

Enclose the entire SYSOPTS string in double quotation marks. Enclose each subparameter string in parentheses. Separate subparameters by blanks. For example:

```
SYSOPTS="TYPE(OBJ) MAXRCD(*NOMAX) AUT(EXCLUDE)"
```

TYPE(OBJ) specifies that the object is to be copied to the IBM Connect:Direct i5/OS node and is assumed to be in save file format. This parameter is required.

EXITCMD(valid i5/OS command) specifies a command to be executed only if the copy process is successful. This parameter is only valid when the sender is running IBM Connect:Direct i5/OS Version 3.3 or later.

FAILCMD(valid i5/OS command) specifies a command to be executed only if the copy process is not successful. This parameter is only valid when the sender is running IBM Connect:Direct i5/OS Version 3.3 or later.

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 i5/OS security, refer to the *IBM i5/OS Data Management Guide and Security Concepts and Planning* manuals.

(TO) SYSOPTS = "TYPE(SPLF)

EXITCMD(valid i5/OS command)

FAILCMD(valid i5/OS 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 IBM Connect:Direct i5/OS COPY statement. The name of the IBM Connect:Direct printer device file is NDMPRINT. This file is created by the installation process. The defaults used by Sterling 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.

Enclose the entire SYSOPTS string in double quotation marks. Enclose each subparameter string in parentheses. Separate subparameters 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 i5/OS CL command CHGPRTF (Change Printer File) to use the options.

TYPE(SPLF) specifies that the data is copied to an i5/OS spooled output file. This parameter is required.

EXITCMD(valid i5/OS command) specifies a command to be executed only if the copy process is successful. This parameter is only valid when the sender is running IBM Connect:Direct i5/OS Version 3.3 or later.

FAILCMD(valid i5/OS command) specifies a command to be executed only if the copy process is not successful. This parameter is only valid when the sender is running IBM Connect:Direct i5/OS Version 3.3 or later.

DEV(*JOB|*SYSVAL|device-name) specifies the name of the printer device description.

- ***JOB** indicates the printer used by the IBM Connect:Direct job is to be used as the printer device.

- ***SYSVAL** indicates that the value in the i5/OS system value QPRTDEV is to be used as the printer device.
- **device-name** identifies the printer device used for nonspoiled output with the printer device to produce the printed output. For spooled output, if OUTQ(*DEVQ), 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 z/OS 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(*DEV D | *CONT | *CUT | *AUTOCUT) specifies the form feed attachments used by the printer (4214, 5219, and 5553 printers only).

- ***DEV D** 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 | *DEV D | *NLQ) specifies the quality of print produced.

- ***STD** indicates standard quality.
- ***DRAFT** indicates draft quality.
- ***DEV D** 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 | *DEV D | 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.
- ***DEV D** 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(*DEV D | *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.

- ***DEV D** indicates that the CHRID value the device is designed to handle is used.
- ***SYSVAL** indicates that the CHRID specified for the system on which IBM 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 IBM 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 IBM 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).

(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.

Microsoft Windows Statements and Parameters

Connect:Direct for Microsoft Windows Process Statement

The Process statement defines the attributes of a IBM Connect:Direct Process and must be the first statement in a Process.

The following is the Connect:Direct for Microsoft Windows process statement format. Refer to [Connect:Direct for Microsoft Windows Process Parameters](#) for more information.

Label	Statement	Parameters
process name	PROcEss	class=n
		execprty=nn
		hold=yes no call
		crc=on off
		localacct="pnode node accounting data"
		pnodeid=(id [,pswd])
		notify=username
		prty=nn
		snode=[nodename] [hostname IPaddress;portnumber servicename]
		remoteacct="snode accounting data"
		snodeid=(id [,pswd[,newpswd]])
		retain=yes no initial

Label	Statement	Parameters
		SECURE=(OFF) or SECURE=(<protocol>[, [<cipher_suite> (cipher_suite_list)]] [,ENC[RYPT.DATA]=Y N]) or SECURE=(ENC[RYPT.DATA]=Y N) Note: See “Connect:Direct for Microsoft Windows Process Parameters” on page 90 page for details on protocol and cipher_suite specifications.
		startt=([date day][,time [am pm]])
		&symbolic name1=variable string 1 &symbolic name2=variable string 2 . . . &symbolic namen=variable string n

Connect:Direct for Microsoft Windows Copy Statement

The copy statement copies text or binary files with a remote IBM Connect:Direct nodes.

The copy statement contains a from clause that specifies the source file name and a to clause that specifies the destination file name. You can specify additional parameters to customize the file transfer operation.

To copy from one IBM Connect:Direct platform to another, refer to the appropriate Copy Statement chapters for the platforms.

The copy statement produces a return code that can be used in conditional statements.

The following is the Connect:Direct for Microsoft Windows copy statement format. Refer to [Connect:Direct for Microsoft Windows Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	copy	from (
		file=filename dsn=filename
		file=businessprocess mailbox
		pnode snode

Label	Statement	Parameters
		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)"
)
		to (
		file=filename dsn=filename
		file=businessprocess mailbox
		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=[(] new mod rpl shr [)]
)
		ckpt=no nnnnnnnnk nnnnnnnnm
		compress [[primechar = x'xx' x'20' c'c'] extended] Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.
		SECURE = (ENC[RYPT.DATA]=Y N)

Connect:Direct for Microsoft Windows Run Job Statement

The run job statement calls external programs that run on the same system as Connect:Direct for Microsoft Windows.

The called program runs as a separate Microsoft Windows process. Process statements that follow the run job statement do not wait for the submitted job to complete.

IBM Connect:Direct does not verify the validity of the submitted job statements or commands. A IBM Connect:Direct return code indicates the success or failure of the run job statement, not the success or failure of the submitted job.

The following is the Connect:Direct for Microsoft Windows run job statement format. Refer to [Connect:Direct for Microsoft Windows Process Parameters](#) for more information.

Label	Statement	Parameters
step name	run job	pnode snode
		dsn=Windows
		sysopts="pgm(filespec) cmd(command parms) args(arguments) desktop(yes no)"

Connect:Direct for Microsoft Windows Run Task Statement

The Connect:Direct for Microsoft Windows run task statement calls external programs that run on the same system as Connect:Direct for Microsoft Windows.

The new program runs as a separate Microsoft Windows process. The IBM Connect:Direct Process waits for the completion of the command or program specified before continuing.

Note: Do not specify programs in the run task statement that cannot complete without user intervention. Unless you explicitly allow IBM Connect:Direct to interact with the desktop through the Control Panel's Services applet, you cannot provide input to the program and it will not complete execution.

The run task execution results in a return code which is the exit code for the program executed using run task.

Note: When using an external program executed by the IBM Connect:Direct run task statement, do not use a return code of 16 in the external program or the IBM Connect:Direct Process will fail.

The following is the Connect:Direct for Microsoft Windows run task statement format. Refer to [Connect:Direct for Microsoft Windows Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	run task	pnode snode
		pgm=Windows
		sysopts="pgm(filespec) cmd(command parms) args(arguments) desktop(yes no)"
		restart(yes) restart (no)

Connect:Direct for Microsoft Windows Submit Statement

The submit statement submits a IBM Connect:Direct Process from within an executing IBM Connect:Direct Process.

The Process to be submitted must reside on the node where the submit statement executes. This node is referred to as the subnode.

Any parameter values in the submit statement override the parameter values contained in a process statement.

The submit statement produces a return code. The return code indicates the success of the submit statement, not the success of the submitted Process.

The SUBMIT statement is not the same as the SUBMIT command.

The following is the Connect:Direct for Microsoft Windows submit statement format. Refer to [Connect:Direct for Microsoft Windows Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	submit	file=filename
		<i>business process name</i>
		class=n
		execprty=nn
		hold=yes no call
		localacct="pnode node accounting data"
		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

Connect:Direct for Microsoft Windows Conditional Statements

Conditional statements alter the sequence of IBM Connect:Direct Process execution based on the completion of the previous step in the Process. For example, if a file copy fails, the Process may call an external program to generate a console message and stop the Process. If the file copy succeeds, the Process continues with the next step.

The following is the Connect:Direct for Microsoft Windows Conditional statement format. Refer to [Connect:Direct for Microsoft Windows Process Parameters](#) for more information.

Label	Statement	Parameters
optional	if	(label condition nn) then
		(process steps executed if condition is true)
	else	
		(process steps executed if condition is false)
	EIF	
optional	goto	label
		exit

Connect:Direct for Microsoft Windows Pend Statement

The optional pend statement indicates the end of an Connect:Direct for Microsoft Windows Process. There are no parameters for the pend statement.

Connect:Direct for Microsoft Windows Process Parameters

class = n

determines the node-to-node session on which a Process executes. A Process can execute in the class specified or any higher session class up to the maximum allowed for a node (**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**.

ckpt=no | nnnnnnnk | nnnnnnm

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.

nnnnnnnk specifies the number of kilobytes.

nnnnnnnm 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.

Note: Not all combinations of source and destination codepages may be supported by the underlying platform. If the COPY step fails with an MBCS001E error, try the third method below using UTF-8 or UTF-16 as the transitional codepage.

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.

Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

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 for Microsoft 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.

condition

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.

disp = [(new | mod | rpl | shr)]

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**.

shr indicates the file must already exist and will be overwritten.

dsn=Windows

specifies the job as either a Microsoft Windows program or command. This field is required.

eof

is required for specifying the end of the **if then** or **if then else** block of statements. No parameters exist.

else

designates a block of IBM Connect:Direct statements that execute when the **if/then** condition is not satisfied. No parameters exist.

execprty = nn

specifies the operating system priority to be assigned to the Process. The **execprty** keyword determines how Microsoft 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.

Scheduling Processes to run with a high priority may have an adverse effect on the execution of other applications in the system.

exit

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

file=filename | dsn=filename

specifies the 1-256 character file name on the **copy** statement. This parameter is required 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 Microsoft Windows version 3.51 or later system.

Connect:Direct for Microsoft 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.

Specify the UNC form of the filename if the file is not on a drive directly connected to the Microsoft Windows server where IBM Connect:Direct resides. The UNC name format is as follows:

```
\servername\share point name\filename
```

Replace the **servername** with the name of the Microsoft Windows server where data resides. The **share point name** represents the name under which the remote Microsoft Windows server shares the directory you want to access. The **filename** specifies the name of the file and includes any subdirectories, if appropriate.

If the file is on the Microsoft Windows server where IBM Connect:Direct is installed, you can specify the drive letter.

Enclose the file name in double quotation marks if you use a statement name, parameter name, or subparameter name for the file name.

file = filename

specifies the 1-256 character file name of the file containing the Process in the **submit** statement. 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.

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 if you use a statement name, parameter name, or subparameter name for the file name.

This field is required.

from

specifies the parameters and subparameters that define the source file characteristics. This field is required.

goto

moves to a specific step within a Process. See the following Field Descriptions section for details.

hold = yes | no | call

specifies that a Process is held from executing until released.

yes indicates that the Process is held until it is explicitly released by a **change process** command. When both **hold=yes** and **startt** values are specified, the **hold** specification takes precedence. So, 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. This is the default value.

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.

When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

if then

specifies that IBM Connect:Direct executes a block of IBM 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.

Label

IBM Connect:Direct statements are identified by user-defined 1-8 character alphanumeric labels. The label must begin in position one.

Statement names and keywords are reserved and cannot be used as labels.

Labels are required on the Process statement. The Process statement must be on the same line as the label.

In a conditional statement, the **label** parameter specifies the step to which the condition applies. This parameter is required in a conditional statement.

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.

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.

nn

specifies the numeric value used for return code checking of the step named in the **label1**. If specified as **x'nn'**, it is a hexadecimal value. Any other specification indicates it is decimal. Standard return codes are as follows:

- 0 indicates successful completion.
- 4 indicates warning.
- 8 indicates error.
- 16 indicates catastrophic error.

notify = username

specifies the user name to receive Process completion messages.

pend

The **pend** statement marks the end of the Connect:Direct for Microsoft Windows Process. No parameters are associated with the **pend** statement.

pgm=Windows

specifies the task as either a Microsoft Windows program or command. This field is required.

pnode | snode

On a **copy** statement, this parameter specifies whether the file is sent from the pnode or the snode. as shown on the following table:

Value	From Parameter	To Parameter
pnode	File to be copied is sent from the pnode. This is the default for the from parameter	File to be copied is sent from the snode
snode	File to be copied is sent from the snode	File to be copied is sent to the snode. This is the default for the to parameter/

For a run task or run job statement, this value specifies the IBM Connect:Direct node where the statement executes. The default is **pnode**.

pnodeid = (id [, pswd])

specifies security user IDs and passwords for the pnode. Each parameter value can be 1-20 alphanumeric characters long.

id specifies the user ID 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 specified in the order shown.

When copying to a file server other than the one containing IBM Connect:Direct, **pnodeid** is required.

Process name

The Process name is a user-defined 1-8 character alphanumeric string. It must begin in position one. You cannot use a statement name, parameter name, or subparameter name as the Process name. This field is required on the Process statement. The **process** keyword must be on the same line as the Process name.

prty = nn

specifies the selection priority of the Process in the Transmission Control Queue (TCQ). 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.

Values range from 0-15. The highest priority is 15. 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.

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 on the copy statement overrides the accounting data specified on the process statement.

restart(yes) | restart(no)

specifies whether a program runs again after a session failure. This value overrides the value set in the initialization parameter.

retain = yes | no | initial

specifies whether the Process is retained in the TCQ after execution.

yes specifies that the Process is retained in HR status after execution. A Process submitted with **retain=yes** remains in the TCQ after execution until:

- it is explicitly released through the **change process** command)
- Until the time specified in the **startt** parameter is reached
- Until the Process is 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 cloned Process runs every week on that day and at that time. If **startt** specifies day only, the cloned Process runs every week on that day at 12:00 a.m. If **startt** specifies time only, the cloned Process runs every day at that time.

no specifies that the Process is deleted immediately after execution. This is the default value.

initial specifies that the Process is retained in the Hold queue and automatically executes each time the Connect:Direct for Microsoft 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.

Note the following:

- The Submit command fails if you specify both the **startt** parameter and the **retain=initial** parameter.
- When a Process is submitted with **retain=yes** and **hold=no** or **hold=call**, the **hold** parameter is ignored.

snode = [nodename] | [hostname | IPaddress;portnumber|servicename]

is a 1-16 alphanumeric character string that specifies the IBM Connect:Direct snode name. This parameter is required on either the submit command or the process statement.

nodename is the node name of the IBM Connect:Direct snode. The snode name corresponds to a IBM Connect:Direct **remote.node** object in the network map.

hostname is the name of the host machine where the IBM Connect:Direct snode is running. This is applicable only for TCP/IP.

IPaddress is the IP address of the IBM 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 IBM Connect:Direct software. The **portnumber** is a decimal number from 1024-65535. This is applicable only for TCP/IP.

The snode parameter on the copy statement overrides the value specified in the process statement.

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

specifies security user IDs 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 IBM Connect:Direct, **snodeid** is required.

id specifies the user ID used as a security ID on the snode.

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 user ID 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 for Microsoft Windows node.

If **snodeid** is specified, **id** must be specified also. If **newpswd** is specified, **pswd** must be specified also. Specify these values in the order of **id**, **pswd**, and **newpswd**.

For IBM Connect:Direct HP NonStop nodes, **snodeid** specifies the HP NonStop group number and user number. The valid range for these numbers is 0-255

SECURE=(OFF)

or

SECURE=(<protocol>[, [<cipher_suite> | (cipher_suite_list)]] [,ENC[RYPT.DATA]=Y|N])

or

SECURE=(ENC[RYPT.DATA]=Y|N)

turns on security for a specific session by allowing you to select a protocol (SSL, TLS, TLS1.1, or TLS1.2) and one or more ciphers when non-secure sessions are the default or turns off security when secure sessions are the default. In addition, you can specify the type of encryption you want performed. Secure+ must be configured to allow overrides or this parameter will generate an error on process submission.

Note: See the table below for details on protocol and cipher_suite specifications.

Parameter	Definition
,ENCRYPT.DATA=Y N or ENC=Y N	Depending on the option chosen, encrypts both the control block information contained in Function Management Headers (FMHs) and the files being transferred (ENC=Y) or encrypts only the FMHs (ENC=N). <ul style="list-style-type: none"> ENCRYPT.DATA must be the last (or only) value specified on the SECURE= parameter. Both sides of the connection must support ENCRYPT.DATA= for SSL/TLS.
OFF	Attempts to start the session as a non-secure session. The session will be successfully established if Secure+ on the snode has been configured to allow overrides or is configured for non-secure sessions.
SSL TLS TLS1.1 TLS1.2	Attempts to start the session as a secure session using the specified protocol. The session will be successfully established if Secure+ on the snode has been configured to allow overrides or is configured for the specified protocol.
,Cipher Suite	Specifies the one cipher suite for IBM Connect:Direct to use when executing the Process overriding the cipher suite defined in the Secure+ parameters file, for example, SSL_RSA_WITH_3DES_EDE_CBC_SHA.
,(Cipher suite list)	Specifies a list of cipher suites for IBM Connect:Direct to use when executing the Process overriding the cipher suite defined in the Secure+ parameters file. The cipher list must be enclosed in parentheses and each cipher suite separated by a comma, for example, (TLS_RSA_AES_128_SHA,TLS_RSA_AES_256_SHA, TLS_RSA_WITH_DES_CBC_SHA)

startt = ([date | day] [,time [am | pm]])

specifies that the Process executes at a specific date and time. The Process is placed in the Timer queue in WS status. The **date**, **day**, and **time** values are positional subparameters. If you do not specify **date** or **day**, precede time with a comma.



CAUTION: Do not specify the **startt** parameter and the **retain=initial** parameter. This will cause the submit command to fail.

date specifies that the Process will execute on a specific date. The current date is the default. 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

day specifies the day of the week that the Process will execute. 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** 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.

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 is specified.

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

is a symbolic parameter assigned a value. The value is substituted within the Process when the symbolic parameter is encountered. You can override this value when you submit the Process.

The following built-in variables can be used for the symbolic variable values; they must be enclosed in double quotation marks:

Value	Description
%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. 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.
%NUM1	Specifies the submission time of the Process in minutes, seconds, and fraction of seconds in the format mmssth.
%NUM2	Specifies the submitted time of a Process as the low order 4 bits of the milliseconds of the time expressed as 1 hex digit (a value from 0 through 15 expressed as 0 through F).
%PNODE	PNODE name where the submit occurs
%PRAND	Pseudo-random number (6 hex digits)
%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. The value returned can be used to create a file name on the node receiving the file.
%SUBDATE1	Use this parameter to substitute the submitted date in the yyyyymmdd 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 mmdyyy 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. The return value can be used to create a file name on the node receiving the file. 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	Specifies a variable that resolves to the user submitting the Process

Do not use statement names, parameter names, and subparameter names as symbol names.

Enclose symbolic parameter value in double quotation marks if it is a parameter, subparameter, or statement name or if it contains special characters. It can be set to a single ampersand symbolic parameter that was resolved during a previous Process submission. Do not use identical symbolic names.

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)"
```

The **sysopts** subparameters 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 Microsoft Windows nodes. The following list shows the default source file attributes assigned for Microsoft Windows text files, which are used if necessary by the remote IBM 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 Microsoft Windows nodes. The following shows the default source file attributes assigned for Microsoft Windows binary files, which are used if necessary by the remote IBM 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 511 characters and cannot contain a closing parenthesis ')' character.

Note: The file open exit DLL is defined in the IBM 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 Microsoft 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 UNIX, Microsoft Windows, z/OS, VM, VSE, and i5/OS nodes.
- **no** does not remove trailing blank characters. The default is **no** for all nodes except UNIX, Microsoft Windows, z/OS, VM, VSE, and i5/OS.

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 for Microsoft 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. This is the default value.

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 Microsoft Windows and all other nodes with the exception of z/OS, VM, VSE, and i5/OS is **no**.
- **yes** specifies the translation should be performed. The default for z/OS, VM, VSE, and i5/OS nodes is **yes**.

xlate.tbl(pathname/filename) specifies that a different translation table from the default table used by IBM Connect:Direct is used. **pathname/filename** should identify a translation table created using the translation table editor of the IBM Connect:Direct Requester for Microsoft 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.

codepage=(source codepage, destination codepage)

The sysopts subparameters valid only in the **to** clause are:

acl(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)

sysopts="pgm(filespec) cmd(command | parms) args(arguments) | desktop (yes | no)"

specifies the parameters for the operation being performed by the **run job** statement. This field is required.

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**.

to

specifies the parameters and subparameters that define the destination file characteristics. This field is required.

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.

OpenVMS Statements and Parameters

Connect:Direct for OpenVMS Process Statement

The Process statement defines the attributes of a Process and is always the first statement in a Process.

The following is the Connect:Direct for OpenVMS Process statement format. Refer to [Sterling Connect:Direct for OpenVMS Process Parameters](#) for more information.

Label	Statement	Parameters
process name	PROCCess	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])
		&symbolic_name_1 = variable-string-1 &symbolic_name_2 = variable-string-2 . . . &symbolic_name_n = variable-string-n

Connect:Direct for OpenVMS Copy Statement

The COPY statement copies files from one node to another.

The COPY statement contains a FROM parameter that specifies the source file name and a TO parameter that specifies the destination file name. You can specify other parameters to customize the file transfer operation.

To copy from one IBM Connect:Direct platform to another, refer to the appropriate COPY FROM and TO pages for those platforms. For example, if the source file is located on a Connect:Direct for OpenVMS node, refer to the OpenVMS COPY FROM information. If the destination for the file is a Connect:Direct for z/OS node, refer to the z/OS TO information.

For information about copying from OpenVMS to a z/OS node, refer to [“Copying from an OpenVMS Node to a z/OS Node”](#) on page 103.

The following is the Connect:Direct for OpenVMS COPY statement format. Refer to [Sterling Connect:Direct for OpenVMS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	COPY	FROM (
		DSN=filename FILE=filename
		PNODE SNODE
		DISP=(OLD SHR)
		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]

Copying from an OpenVMS Node to a z/OS Node

Note the following if you are copying a library from a Connect:Direct for OpenVMS node to a PDS at a Connect:Direct for z/OS node:

Connect:Direct for OpenVMS transmits modules in alphabetical order.

OpenVMS allows module names to be 39 characters. z/OS restricts module names to only eight characters. When a library is copied from OpenVMS to z/OS, z/OS truncates the module name to eight characters. As a result, you should limit OpenVMS module names to unique eight character names.

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. Neither file exists on the z/OS node.

If REPLACE is in effect, the OpenVMS module modulenamea is sent to the Connect:Direct for z/OS node and renamed modulena. Next, the OpenVMS module modulenameb is sent and replaces the module previously sent (as modulenamea) to the Connect:Direct for z/OS node.

If NOREPLACE is in effect, the OpenVMS module modulenamea is sent to the Connect:Direct for z/OS node and renamed modulena. Next, the OpenVMS module modulenameb is sent to z/OS. z/OS truncates the module name to modulena. Because NO REPLACE is in effect, and both modules are identical to eight characters, modulenameb does not replace the module previously sent to the Connect:Direct for z/OS node. modulenameb is not reflected in the library as a new member.

Connect:Direct for OpenVMS Run Job Statement

The RUN JOB statement submits a job at the z/OS, VM, VSE, or OpenVMS nodes. The jobs must reside on the node executing the RUN JOB statement.

IBM Connect:Direct does not verify the job statements. Check IBM Connect:Direct statistics records to determine the completion status of a RUN JOB statement.

The following is the Connect:Direct for OpenVMS Run Job statement format. Refer to [Sterling Connect:Direct for OpenVMS Process Parameters](#) for more information.

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

Connect:Direct for OpenVMS Run Task Statement

The 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 for OpenVMS RUN TASK statement identifies the operations that the operating system should perform.

When a IBM Connect:Direct Process issues a RUN TASK statement, the IBM Connect:Direct Process waits until the OpenVMS process finishes before executing the next IBM Connect:Direct Process step.

You can submit a IBM Connect:Direct Process with a RUN TASK statement from either node.

The following is the Connect:Direct for OpenVMS Run Task statement format. Refer to [Sterling Connect:Direct for OpenVMS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN TASK	(PGM=VMS)
		PNODE SNODE
		SYSOPTS="[OUTPUT= `file specification']
		[CMD= `DCL command']

Connect:Direct for OpenVMS Submit Statement

The SUBMIT statement submits another IBM Connect:Direct Process from within an executing Process. The Process can be submitted to either the PNODE or the SNODE.

The submitted Process must reside on the node where the SUBMIT statement executes. This node is referred to as the SUBNODE.

The following is the Connect:Direct for OpenVMS Submit statement format. Refer to [Sterling Connect:Direct for OpenVMS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	SUBMIT	DSN=filename FILE=filename
		CLASS=n
		NEWNAME=new-name
		SUBNODE=PNODE SNODE
		SNODE=secondary-node-name
		SNODEID=(id [,pswd] [,newpswd])
		SACCT='snode-accounting-data'
		PNODEID=(id [,pswd] [,newpswd])

Label	Statement	Parameters
		PACCT='pnode-accounting-data'
		HOLD=Yes No Call
		PRTY=n
		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

Connect:Direct for OpenVMS Conditional Statements

Conditional statements alter the sequence of IBM Connect:Direct Process execution based on the completion of the previous step in the Process. For example, if a file copy fails, the Process may call an external program to generate a console message and stop the Process. If the file copy succeeds, the Process continues with the next step.

The following is the Connect:Direct for OpenVMS Conditional statement format. Refer to [Sterling Connect:Direct for OpenVMS Process Parameters](#) for more information.

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

Connect:Direct for OpenVMS Symbol Statement

The SYMBOL statement creates symbolic substitution values.

The following is the Connect:Direct for OpenVMS Symbol statement format. Refer to [Sterling Connect:Direct for OpenVMS Process Parameters](#) for more information.

Label	Statement	Parameters
optional	SYMBOL	&symbolic_name=variable-string

Connect:Direct for OpenVMS Process 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 for PARSESS in the network map.

CKPT = [nK | nM]

specifies the byte interval for checkpoint support. Checkpointing enables restart of interrupted transmissions at the last transmission checkpoint, avoiding the need to restart a transmission from the beginning.

Only sequential disk files can be checkpointed.

IBM Connect:Direct 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. Connect:Direct for OpenVMS can only checkpoint a Connect:Direct for z/OSfile if IBM Connect:Direct z/OS initialization parameters are set as CKPT.MODE=(Record Record...). See the Connect:Direct for z/OS documentation for more information.

Specify a checkpoint value of zero in the Process to disable automatic checkpointing. To override automatic checkpointing, use K to denote thousands; M for millions.

COMPRESS [PRIMEchar = X'xx' | X'20' | C'c']

specifies to compress the data, which reduces the amount of data transmitted during a copy and decreases transfer time. The file is automatically decompressed at the destination.

If you specify compression, IBM 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.

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'). However, for variable byte files, use a character other than blank as a compression character.

condition

specifies the type of comparison to be performed. Valid symbols, alternate symbols, and conditions are:

- **= or EQ** specifies that the completion code must be equal to the value nn to satisfy the condition
- **<> or ≠ or NE** specifies that the completion code must not equal the value nn to satisfy the condition
- **>= or ≥ or GE** specifies that the completion code must be greater than or equal to the value nn to satisfy the condition
- **> or GT** specifies that the completion code must be greater than the value nn to satisfy the condition
- **<= or ≤ or LE** specifies that the completion code must be less than or equal to the value nn to satisfy the condition
- **< or LT** specifies that the completion code must be less than the value nn to satisfy the condition

The completion code from the RUN JOB statement is from the job submission only and is not the completion code of the submitted job.

COPY

identifies the COPY statement. you can specify this statement with either the FROM or TO parameter.

DATAEXIT=((exit-name-1 [, C 'p1', C 'p2', ... , C 'pn']) . . . (exit-name-n [, C 'p1', C 'p2', ... , C 'pn']))

calls a user-written program to examine or modify the data for the associated copy. You can specify multiple exit names.

exit-name specifies the name of the user-written program called to examine or modify the data.

parameter specifies a parameter passed to the specified exit. Each parameter must be enclosed in single quotation marks and prefaced by C.

The DATAEXIT parameter is supported on VAX platform only.

The following is an example of the DATAEXIT format:

```
DATAEXIT=((USER.EXIT1,C'PARM1',C'PARM2'))
```

**DCB =([DSORG=PS | PO | KSDS |RRDS]
[,KEYLEN = number-of-bytes]
[,LRECL = number-of-bytes]
[,RECFM = V | F | VB | FB])**

specifies attributes for allocating destination files. These parameters override the source file DCB information.

For SAM-to-SAM copies where the destination file is new and the DCB parameter is not specified with the TO parameter, DSORG is taken from either:

- the source file
- the TYPE defaults file if the TYPE keyword has been specified

The following are the subparameters:

DSORG specifies the file organization. File organizations supported are:

- Physical Sequential (PS), the default
- library files (PO)
- keyed files (KSDS)
- relative files (RRDS)

Connect:Direct for OpenVMS does not support copying a VSAM KSDS file from z/OS to a keyed file under OpenVMS.

KEYLEN specifies the length of the key in bytes 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), the default
- FB (fixed block files)

STREAM overrides RECFM if it is specified.

DEFCONN.MODE = (first | scan | name)

specifies the default method of selecting a communications path used to establish a session for Process execution.

first specifies that Connect:Direct for OpenVMS will only use the first COMM.PATH specification in making a connection.

scan specifies that Connect:Direct for OpenVMS will use each COMM.PATH specification in turn, until a connection is successfully made.

name represents an actual COMM.PATH name that Connect:Direct for OpenVMS should use when making connections.

DSN = 'file-specification'

specifies the file specification of the DCL command procedure. This parameter is required.

Enclose the DSN in single quotation marks.

If you do not specify the OpenVMS device or directory, the login default device or directory is used for the OpenVMS user name. The user name is derived from the IBM Connect:Direct SUBMITTER (node ID, user ID) or specified in the PNODEID or SNODEID.

DSN = filename | FILE = filename

specifies the name of the file that contains the Process. This parameter is required.

EIF

specifies the end of the IF THEN or IF THEN ELSE block of statements. It is required.

ELSE

designates a block of IBM Connect:Direct statements that execute when the IF THEN condition is not satisfied.

EXIT

bypasses all remaining steps within a Process.

FROM

specifies the source file characteristics. This parameter is required.

(FROM) DISP = [OLD | SHR]

specifies the status of data on the source node.

OLD requests exclusive access to a file.

SHR requests shared access to a file. This is the default.

(FROM) DSN | FILE

specifies the source file specification. This parameter is required.

If the source file is a OpenVMS file with a maximum record size of 0 and a longest record size of 0, then DCB LRECL and BLKSIZE information must be specified for the destination file. Use the DCL command ANALYZE/RMS 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, enclose the file specification in single quotation marks if it uses spaces or other special characters.

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 for OpenVMS SUBMIT command string and PNODEID and SNODEID are not specified, the user ID defaults from the SUBMITTER subparameters are used for the file specification. If either PNODEID or SNODEID is provided, the user ID defaults in the PNODEID or SNODEID subparameters are used to complete the file specification.

Use of PNODEID and SNODEID is dependent on the direction of the transfer.

GOTO

moves to a specific step, defined by the label, within a Process.

HOLD = Yes | No | Call

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

Yes submits the Process to the Hold queue where it remains until the operator explicitly releases the Process.

No executes the Process as soon as possible. This is the default.

Call places the Process 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 with HOLD=NO establishes a session and causes execution of any Processes for this node that are designated HOLD=CALL.

IF THEN

specifies that IBM Connect:Direct execute a block of statements based on the completion code of a Process step. You use an EIF statement with an IF THEN statement. A return code with the high order bit on is evaluated as a negative return code.

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. The GOTO statement requires a label. The label name cannot be the label of a preceding step nor the label of the GOTO statement of which it is a part.

Labels must begin in column one. The label consists of a 1-8 character alphanumeric string, with the first letter alphabetic only.

NEWNAME = new-name

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

nn

specifies the numeric value used for completion code checking. If specified as X'nn', it is a hexadecimal value. Any other specification indicates 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 equaling 4 indicates a warning.

PACCT = 'pnode-accounting-data'

specifies the accounting data for the primary node (PNODE). The maximum length of the accounting data is 256 characters. Enclose the string in single quotation marks if special characters are part of the accounting data.

PGM = VMS

indicates that the RUN TASK executes on a Connect:Direct for OpenVMS node. This field is required.

PNODE

For a Copy statement, specifies that the file to be copied resides on the primary node.

PNODE is the default in the COPY FROM statement. If the file specification is incomplete and the PNODEID is specified, the default user ID from the PNODEID is used to complete the file specification. Otherwise the defaults of the submitter user ID are used.

For a Run Task statement, specifies that the task is executed on the primary node, which is the default.

For a Run Job statement, specifies that the job is submitted on the primary node, which is the default.

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 alphanumeric or national characters (@ # \$), 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 IBM Connect:Direct.

id specifies the 1-64 character security ID passed to the security system at the PNODE.

pswd specifies the current 1-64 character security password 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 1-64 character security password. It can be used by the security system to change the current security password to the new security password.

PROcEss

identifies the PROCESS statement.

process name

specifies the 1-8 character name of the Process. The first character must be alphabetic. The Process name must start in column one. The PROCESS keyword must be on the same line as the Process name.

This label identifies the Process in messages or statistics.

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 for OpenVMS. The range is from 0-15. See the *IBM Sterling Connect:Direct for 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 keeps the Process in the Hold queue after initial execution. You must then do one of the following:

- Manually release the Process through the CHANGE PROCESS command to execute it
- Delete the Process through the DELETE PROCESS command
- Specify the STARTT parameter to release the Process again at a specified interval. Using RETAIN=YES with STARTT will run a Process 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. This is the default value.

Initial specifies that the Process executes every time IBM Connect:Direct is initialized. The Process will not execute when initially submitted. Do not specify STARTT with RETAIN=INITIAL.

RUN JOB

identifies the RUN JOB statement.

RUN TASK

identifies the RUN TASK statement.

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 parameter is ignored when the SNODE is a Connect:Direct for i5/OS node.

SELECT = (name | *)

specifies criteria for selecting PDS members for a copy. The SELECT parameter can be specified only with the FROM parameter.

name specifies an individual member name.

***** represents a global generic, which indicates that all members of the file are to be copied.

SNODE

For a Copy statement, specifies that the file to be copied resides on the secondary node. SNODE is the default in the COPY TO statement. If the file specification is incomplete and the SNODEID is specified, the default user ID from the SNODEID is used to complete the file specification. Otherwise the defaults of the submitter user ID are used.

For a Run Task statement, specifies that the task is executed on the secondary node.

For a Run Job statement, specifies that the job is submitted on the secondary node.

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 parameter is required, unless it is specified on the SUBMIT command.

This is the logical node name that has been defined as the adjacent node in the network map.

When specified on the SUBMIT statement, this parameter overrides the value specified in the PROCESS statement. The PNODE and SNODE can specify the same symbolic node name.

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

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

id specifies the 1-8 character security ID passed to the security system on the SNODE.

For Connect:Direct for HP NonStop, this subparameter specifies the HP NonStop group number and user number. These numbers can range from 0-255.

For Connect:Direct for i5/OS, this subparameter specifies the i5/OS user profile used for authorization checks during Process execution. It is limited to 8 characters.

pswd specifies the current 1-8 character 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's security requires a password.

newpswd specifies the new 1-8 character security password. It can be used by the security system to change the current security password to the new security password.

For Connect:Direct for HP NonStop, SAFEGUARD must be running on HP NonStop.

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.

Do not specify STARTT with RETAIN=INITIAL.

If you specify both HOLD=YES and a STARTT value in a Process, the HOLD specification takes precedence, and the Process is placed in the Hold queue.

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)
mmddy	yy/ddd (Julian date)

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

You cannot specify a date in the STARTT parameter if RETAIN=YES.

day specifies the day of the week to release the Process for execution. Valid names include MONday, TUEsday, WEdnesday, THursday, FRiday, SAaturday, and SUNday. You can abbreviate the day value 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.

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

You can specify the time of day using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00:00 to 24:00:00. If you do not use AM and PM, the 24-hour clock is assumed.

If you use the 12-hour clock, 01:00:00 hours can be expressed as 1:00AM, and 13:00 hours can be expressed as 1:00PM.

If you specify hh:mm:ssXM with RETAIN=YES, the Process executes at the same time every day. You do not need to specify minutes and seconds.

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

stepname

specifies the user-defined name of the Copy, Run Job, Run Task, or Submit step.

Stepnames must begin in column one. Stepnames are 1-8 character alphanumeric strings. The first character must be alphabetic.

SUBMIT

identifies the SUBMIT statement.

SUBNODE = PNODE | **SNODE**

specifies the node where the Process defined in this SUBMIT statement executes.

PNODE means that the Process is submitted on the primary node. This is the default.

SNODE means that the Process is submitted on the secondary node.

In both cases, the Process must reside on the node on which it is being submitted.

SYMBOL

identifies the SYMBOL statement.

&symbolic_name=variable-string

specifies a 1-17 character string that is substituted into the Process. This parameter is required for the SYMBOL statement. .

When IBM Connect:Direct encounters a symbolic name, it substitutes the string represented by that name.

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 IBM Connect:Direct statement string can be concatenated, allowing you to move data in a variety of ways.

Specify a null value by an equal sign immediately followed by a comma (=,). Enclose symbolic parameters containing special characters in single quotation marks.

&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. The SUBMIT command will override this value.

Specify a null value by =, (an equal sign immediately followed by a comma). Enclose symbolic parameters strings containing special characters in single quotation marks.

When specified on the SUBMIT statement, the symbolic parameter must begin with a single ampersand. This allows 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.

```
SYSOPTS = "[KEEP | NOKEEP]  
[LOG='file-specification'] | NOLOG]  
[Pn='string']  
[PRINT | NOPRINT]  
[QUEUE='queue-name[:]']  
[WAIT]"
```

specifies system operation parameters. Enclose the string in double quotation marks.

KEEP | NOKEEP specifies log file disposition.

- **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 a file specification is provided, output is saved in a log file. If the file specification is not provided, the log file name uses the name of the command procedure with an extension of LOG.

NOLOG specifies that a log file is not created.

The default is to keep the log and name it after the first (or only) file in the job.

Pn='string' passes parameters to the command procedure. The range for n is 1-8.

PRINT | NOPRINT specifies whether the log file for the job is queued for printing upon job completion.

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 RUN JOB waits for the command procedure to finish before continuing processing. The default is that the statement does not wait for the OpenVMS command procedure to complete.

**SYSOPTS= "[MOUNT='string']
[DISMOUNT | NODISMOUNT]
[TYPE='string']
[LIBRARY='string']
[REPLACE | NOREPLACE]
[BINARY | NOBINARY]
[PROTECTION='string']
[DIROWN | NODIROWN]
[DISSETPROT | NODISSETPROT]
[XLATE='string']"**

performs Connect:Direct for OpenVMS system operations during file transfer.

Enclose the entire SYSOPTS string in double quotation marks. Enclose each subparameter string in single quotation marks. Separate the subparameters by blanks. For example:

```
SYSOPTS="MOUNT='MUA0 TAPELABEL' NODISMOUNT"
```

MOUNT='string' is any valid Digital Command Language (DCL) MOUNT command qualifiers 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.

Do not specify ASSIST on the MOUNT command 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 IBM 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 for 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 transf

er. BINARY (no text conversion) specifies that Connect:Direct for 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 for 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. Specify this subparameter in the format of the DCL SET PROTECTION command. The default is the protection previously defined on the OpenVMS process running Connect:Direct for 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 IBM 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 for OpenVMS NDM\$\$DISSETPROT initialization parameter definition is 1, Connect:Direct for 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 for 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 for 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 IBM Connect:Direct system uses the table named NDM_DEFAULT. The following is an example of the format for XLATE:

```
XLATE='XLT_TABLE1'
```

SYSOPTS= "[OUTPUT='file specification'] [CMD='DCL command']"

specifies DCL commands and parameters on the RUN TASK statement. Enclose the entire SYSOPTS string in double quotation marks. Enclose each subparameter string in single quotation marks. Separate the subparameters with blanks.

OUTPUT='file specification' specifies the SYS\$OUTPUT of the Process. If the file is not included as part of this parameter, the log file name defaults to the command procedure name, with an extension of LOG.

Only use the OUTPUT subparameter when you do not use the Remote Procedure Execution (RPX) Facility. If you use OUTPUT in a RUN TASK statement and RPX is enabled, OUTPUT is ignored.

If you specify OUTPUT='device', the Process output is written to the root directory of the account that issued the command. The output is assigned a type of .LOG. There is no file name.

If you specify OUTPUT='file' (without a directory), the file is written to the root directory on the device assigned to the account that created the Process.

CMD='DCL command' executes a DCL command by the Process. You can specify an arbitrary number of CMD parameters. You can specify any DCL commands that you are authorized to issue, with the exception of commands that require single or double quotation marks.

If a DCL command procedure uses a string parameter containing embedded blanks and normally requiring single or double quotation marks, replace the embedded blanks with underscores, eliminating single or double quotation marks. For example, the DCL command **MAIL/SUBJECT "two words" filename** can be specified in the RUN TASK statement as follows:

```
SYSOPTS="CMD='MAIL/SUBJECT=two_words filename'"
```

THEN

specifies the subsequent processing to perform if the specified condition is met.

TO

specifies the destination file characteristics. This parameter is required.

(TO) DISP = [RPL | NEW | OLD | SHR | MOD]

specifies the status of the data on the destination node.

RPL replaces the contents of the file if the version is specified and that version exists. Otherwise, the IBM Connect:Direct system creates the file or creates 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. This is the default.

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 existing file.

If DISP=MOD is specified on the TO clause of the Connect:Direct for OpenVMS COPY statement and the destination file is a OpenVMS sequential file, Connect:Direct for OpenVMS supports checkpoint-restart. Connect:Direct for z/OS does not support checkpoint-restart when DISP=MOD is specified on the TO clause of the Connect:Direct for z/OSCOPY statement.

(TO) DSN | FILE

specifies the destination file specification. Either the DSN or FILE parameter is valid. This parameter is required.

If you are copying a file to an IBM node, enclose the file specification in single quotation marks if it uses spaces or other special characters.

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 for OpenVMS SUBMIT command string and PNODEID and SNODEID are not specified, the user ID defaults from the SUBMITTER subparameters are used for the file specification. If either PNODEID or SNODEID is provided, the user ID defaults in the PNODEID or SNODEID subparameters are used to complete the file specification.

Use of PNODEID and SNODEID is dependent on the direction of the transfer.

TYPE = typekey

specifies the name of the type file that contains the 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 z/OS, VM, or VSE to OpenVMS copies where the typekey exceeds eight characters, enter the typekey into the SYSOPTS parameter on the TO clause of the Connect:Direct for OpenVMS COPY statement.

On copies from Connect:Direct for OpenVMS to z/OS, the typekey must not be greater than eight characters.

UNIX Statements and Parameters

Connect:Direct for UNIX Process Statement

A process statement defines the attributes of a IBM Connect:Direct Process and is always the first statement in a Process.

The following is the Connect:Direct for UNIX Process statement format. Refer to [“IBM Connect:Direct for UNIX Process Parameters”](#) on page 122 for more information.

Label	Statement	Parameters
process name	PROcEss	class=n
		crc= on off
		hold=yes no 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
		SECURE=(OFF) or SECURE=(<code><protocol></code> [, [<code><cipher_suite></code> (<code>cipher_suite_list</code>)]] [,ENC[RYPT.DATA]=Y N]) or SECURE=(ENC[RYPT.DATA]=Y N) Note: See the process parameters page for details on protocol and cipher_suite specifications.
		sacct="snode accounting data"
		snode=[nodename] [(hostname nnn.nnn.nnn.nnn);(portnumber portname)]
		snodeid=(id [,pswd[,newpswd]])
		startt=(<code>[date day]</code> [,hh:mm:ss[am pm]])
		&symbolic name1=variable string 1 &symbolic name2=variable string 2 . . . &symbolic namen=variable string n

Connect:Direct for UNIX Copy Statement

The copy statement copies a file from one IBM Connect:Direct node to another.

The copy statement contains a from clause that specifies the source file name and a to clause that specifies the destination file name. You can specify additional parameters to customize the file transfer.

To copy from one IBM Connect:Direct platform to another, refer to the appropriate Copy Statement chapters for the platforms.

The copy statement produces a return code that can be used in conditional statements.

The following is the Connect:Direct for UNIX copy statement format. Refer to [“IBM Connect:Direct for UNIX Process Parameters”](#) on page 122 for more information.

Label	Statement	Parameters	
stepname	COPY	from	(
			file=filename dsn=filename
			file=businessprocess mailbox
			pnode snode

Label	Statement	Parameters
		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):" " :yes no:" " :recdl=x<hex value of record delimiter>:"
)
		ckpt=no nk nm
		compress [[primechar = x'xx' x'20' c'c'] [extended [(CMPlevel=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)]] Note: If the SNODE is B2B Integrator, only COMPRESS EXT is supported.
		SECURE = (ENC[RYPT.DATA]=Y N)
		to (
		file=filename dsn=filename
		file=businessprocess mailbox
		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):" " :recdl=x<hex value of record delimiter>:"
		disp=[(] new mod rpl [)]
)ZFBA=2 (AIX only) Note: ZFBA= <i>n</i> where <i>n</i> is the number of devices to be used for the transfer between 2-10 devices.

Connect:Direct for UNIX Run Job Statement

The run job statement invokes a UNIX command shell to execute one or more UNIX commands from within a Process. The Process does not wait until the UNIX commands finish running before executing the next step in the Process.

The run job execution produces a return code. The return code does not indicate if the UNIX command successfully executed. The return code only indicates if IBM Connect:Direct successfully submitted the run job statement.

The following is the Connect:Direct for UNIX Run Job statement format. Refer to [“IBM Connect:Direct for UNIX Process Parameters”](#) on page 122 for more information.

Label	Statement	Parameters
step name	RUN JOB	dsn=dsn[(member)]
		pnode snode
		sysopts="unix command [;unix command [;unix command...]]"

Connect:Direct for UNIX Run Task Statement

The run task statement invokes a UNIX command shell to execute one or more UNIX commands within a Process. The Process waits until the UNIX commands finish running before executing the next step in the Process.

Note: 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 command completes. The run task execution results in a return code which is the exit code for the program executed using run task.

The following is the Connect:Direct for UNIX Run Task statement format. Refer to [“IBM Connect:Direct for UNIX Process Parameters”](#) on page 122 for more information.

Label	Statement	Parameters
stepname	run task	(pgm=program-name)
		pnode snode
		sysopts="unix command [;unix command [;unix command...]]"

Connect:Direct for UNIX Submit Statement

The submit statement submits another IBM Connect:Direct Process from within an executing Process. The Process can be submitted to either the pnode or the snode.

The submitted Process must reside on the node where the submit statement executes. This node is referred to as the subnode.

Any values specified in the submit statement override the values contained in the process statement.

The submit statement execution produces a return code.

The submit statement is not the same as the submit command.

The following is the Connect:Direct for UNIX Submit statement format. Refer to [“IBM Connect:Direct for UNIX Process Parameters”](#) on page 122 for more information.

Label	Statement	Parameters
stepname	submit	file=filename
		<i>business process name</i>

Label	Statement	Parameters
		class=n
		hold=yes no call
		newname=new process name
		notify=username@hostname or user@localhost
		pacct="pnode accounting data"
		plexclass=remote_plexclass
		pnodeid=(id [,pswd])
		prty=nn
		retain=yes no initial
		sacct="snode accounting data"
		snode=[nodename] [(hostname nnn.nnn.nnn.nnn);(portnumber portname)]
		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

Connect:Direct for UNIX Conditional Statements

Conditional statements alter Process execution sequence based on the completion of the previous step in the Process.

The following is the Connect:Direct for UNIX Conditional statements format.

Label	Statement	Parameters
optional	if	(label condition nn) then
		(process steps)
	else	
		(alternative process steps)
	eif	
optional	goto	label
		exit

Connect:Direct for UNIX Pend Statement

The optional pend statement indicates the end of an Connect:Direct for UNIX Process. There are no parameters for the pend statement.

IBM Connect:Direct for UNIX Process Parameters

ckpt = no | n | nK | nM | nG

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. Specify 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
<i>nnnnnnnnnn</i>	number of bytes
<i>nnnnnnnK</i>	number of kilobytes
<i>nnnnM</i>	specifies a number of megabytes
<i>nG</i>	specifies a number of gigabytes

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.

compress [[primechar = x'xx' | x'20' | c'c'] | [extended [(CMPlevel=1|2|3|4|5|6|7|8|9 WINDsize=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. Specify this parameter between the from and to parameters.

Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

primechar specifies the primary compression character. The default value for primechar is x'20'.

IBM Connect:Direct 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 CMPlevel, WINDsize, and MEMlevel to refine the extended compression.

CMPlevel 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 initialization parameters.

WINDsize 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

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.

copy

identifies the copy statement.

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 Connect:Direct Secure Plus or SNA. If CRC checking is enabled for a Process on a Connect:Direct Secure Plus 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.

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.

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. This parameter is required.

eof

is required for specifying the end of the if then or if then else block of statements. No parameters exist.

else

designates a block of IBM Connect:Direct statements that execute when the if then condition is not satisfied. No parameters exist.

exit

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

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. This parameter is required.

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.

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 for 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 for UNIX node to perform a wildcard copy send to any other IBM Connect:Direct platform. With a IBM Connect:Direct UNIX node, you can receive a wildcard copy only from a Connect:Direct for Microsoft Windows node or from another Connect:Direct for UNIX node.

from

specifies the source file characteristics. This parameter is required.

goto

moves to a specific step within a Process. See the following Field Descriptions section for details.

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. 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.

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.

if then

specifies that a block of IBM Connect:Direct statements execute based on the completion code of a Process step. An eif statement must be used in conjunction with an IF THEN statement.

Label

IBM 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

Although the IBM Connect:Direct UNIX label may be up to 256 characters long, labels for many of the IBM Connect:Direct platforms cannot exceed eight characters.

A label is required for the Process statement to define a Process name. The Process name can be used as the pname parameter value on the change process, delete process, flush process, select process, and select statistics commands for the name of the target Process. The Process statement must be on the same line as the label.

A label is not required on any other IBM Connect:Direct UNIX statement.

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.

nn

This parameter specifies the numeric value to be used for return code checking. If specified as x'nn', it is a hexadecimal value. Any other specification indicates it is decimal. Standard return codes are as follows:

- 0 indicates successful completion.
- 4 indicates warning.
- 8 indicates error.
- 16 indicates catastrophic error.

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.

pend

The pend statement is optional and marks the end of an IBM Connect:Direct UNIX Process. There are no parameters associated with the pend statement.

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.

plexclass=remote_plexclass

specifies the 1–8 character name of a valid plexclass on the remote server. This parameter gives you control over which IBM Connect:Direct/Plex Server is selected by the IBM Connect:Direct/Plex Manager to execute a Process. For example, you can specify plexclass=tape in a Process to direct a Process to a IBM Connect:Direct/Plex Server that has a tape drive.

pnode

When specified on the **copy from** parameter, the file to be copied resides on the primary node. When specified on the **copy to** parameter, the file is sent to the primary node. Pnode is the default for the from parameter.

On a run job or run task statement, specifies that the job executes on the PNODE.

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 user ID 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 specified 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 1-15. 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 specified when retain=initial is specified. This causes the SUBMIT command to fail.

run job

identifies the run job statement.

run task

identifies the run task statement.

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.

SECURE=(OFF)

or

SECURE=(`<protocol>` [, [`<cipher_suite>` | (`cipher_suite_list`)]] [,ENC[RYPT.DATA]=Y|N])

or

SECURE=(ENC[RYPT.DATA]=Y|N)

turns on security for a specific session by allowing you to select a protocol (SSL, TLS, TLS1.1, or TLS1.2) and one or more ciphers when non-secure sessions are the default or turns off security when secure sessions are the default. In addition, you can specify the type of encryption you want performed. Secure+ must be configured to allow overrides or this parameter will generate an error on process submission.

Note: See the table below for details on protocol and cipher_suite specifications.

Parameter	Definition
,ENCRYPT.DATA=Y N or ENC=Y N	Depending on the option chosen, encrypts both the control block information contained in Function Management Headers (FMHs) and the files being transferred (ENC=Y) or encrypts only the FMHs (ENC=N). <ul style="list-style-type: none"> ENCRYPT.DATA must be the last (or only) value specified on the SECURE= parameter. Both sides of the connection must support ENCRYPT.DATA= for SSL/TLS.
OFF	Attempts to start the session as a non-secure session. The session will be successfully established if Secure+ on the snode has been configured to allow overrides or is configured for non-secure sessions.
SSL TLS TLS1.1 TLS1.2	Attempts to start the session as a secure session using the specified protocol. The session will be successfully established if Secure+ on the snode has been configured to allow overrides or is configured for the specified protocol.

Parameter	Definition
Cipher Suite	Specifies the one cipher suite for IBM Connect:Direct to use when executing the Process overriding the cipher suite defined in the Secure+ parameters file, for example, SSL_RSA_WITH_3DES_EDE_CBC_SHA.
(Cipher suite list)	Specifies a list of cipher suites for IBM Connect:Direct to use when executing the Process overriding the cipher suite defined in the Secure+ parameters file. The cipher list must be enclosed in parentheses and each cipher suite separated by a comma, for example, (TLS_RSA_AES_128_SHA,TLS_RSA_AES_256_SHA, TLS_RSA_WITH_DES_CBC_SHA)

snode

When specified with the **copy from** parameter, the file to be copied resides on the secondary node. When specified with the **copy to** parameter, the file is sent to the secondary node. Snode is the default for the **to** parameter.

On a run job or run task statement, specifies that the job executes on the PNODE.

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 specified in the process statement. It is required either on the submit command or process statement.

nodename is the node name of the adjacent IBM 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 IBM Connect:Direct is running. This is applicable only for TCP/IP.

nnn.nnn.nnn.nnn is the IP address of the remote IBM 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 IBM Connect:Direct software. The portnumber is a decimal number from 1024-65535. The default is 1364. This is applicable only for TCP/IP.

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

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 user ID 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 user ID and old password are correct. The newpswd parameter is not valid if the snode is a IBM 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 specified 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.

The startt parameter should not be specified when retain=initial is specified. 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.

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

submit

identifies the submit statement.

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. Enclose the symbolic parameter value 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.

If a symbolic value is a double quoted string and it is desired to preserve the double quotes when the symbol is resolved in the process, enclose the double quoted string in single quotes. For example:
&filename = '"file name with spaces"'

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):" " :=yes | no:"

":recdl=x<hex value of record delimiter>:"

specifies system-specific parameters on the copy statement. The subparameters are specified in the same format as fields within a IBM 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 IBM 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 IBM Connect:Direct node:

- dsorg=ps
- recfm=u
- lrecl=0
- blksize=23040

vb indicates the file is in variable block format. Variable block format is structured as follows:

BDW | RDW | Data . . . | BDW | RDW | 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=record 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 IBM 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. This is the default.
- no specifies trailing blank characters are not removed.

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, `perm=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)=yes | no specifies whether IBM Connect:Direct will automatically decompress a file that was ressed with the `cdsacomp` utility. See the *IBM Sterling Connect:Direct for UNIX User's Guide* for more information on the `cdsacomp` utility.

- **Yes** indicates that the from data set is ressed and tells IBM Connect:Direct to decompress the file as part of the Process.
- **No** tells IBM Connect:Direct that the from file is not compressed, or to send the compressed file in compressed format. ressed files copied with this value can be decompressed offline on another Connect:Direct for UNIX or Connect:Direct for UNIX system using the `cdsacomp` utility.

If `sysopts` are not coded or if `sysopts=" :=no: "`, a ressed file is sent in compressed format and the receiver must run `cdsacomp` with `"-mode decompress"`. If the from file is not compressed, regular or extended compression may be used in the Copy step. Do not use regular or extended compression in the Copy step if the from file is compressed.

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.

recdl=x<hex value of record delimiter>

provides a way to specify a non-standard record delimiter for source and destination text files. For example, if the source file is EBCDIC and uses the EBCDIC NL (new line character) as the record delimiter, the source file `sysopts` would include `:RECDL=x15 :`

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. This parameter is required.

The run job statement does not wait until the UNIX command is completed to complete execution. Once started, the UNIX command or commands in a run job statement will be permitted to continue execution. Connect:Direct for UNIX does not have control over the UNIX commands.

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.

then

specifies subsequent processing based on the other specified parameters.

to

specifies the destination file characteristics. This parameter is required

VM/ESA Statements and Parameters

Connect:Direct for VM/ESA Process Statement

The PROCESS statement defines the attributes of a Process and is always the first statement in a Process.

The following is the Connect:Direct for VM/ESA Process statement format. Refer to [Sterling Connect:Direct for VM/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
process name	PROcEss	SNODE = secondary-node-name
		SACCT = 'snode-accounting-data'
		PNODE = primary-node-name
		PNODEID = (id [,pswd] [,newpswd])

Label	Statement	Parameters
		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
		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

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.

Connect:Direct for VM/ESA Copy Statement

The COPY statement copies CMS (Conversational Monitor System) files, a set of CMS files, a Group of files, and VSAM files between nodes.

The COPY statement contains a FROM parameter that specifies the source file name and a TO parameter that specifies the destination file name. You can specify additional parameters to customize the file transfer.

To copy from one IBM 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 for HP NonStop, refer to the Connect:Direct for HP NonStop COPY FROM information. If the destination file is on Connect:Direct for VM/ESA, refer to the VM/ESA COPY TO information.

The following is the Connect:Direct for VM/ESA COPY statement format. Refer to [Sterling Connect:Direct for VM/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	COPY	From (
		(FROM) DSN = 'filename filetype' GROUP = '* * ... *'
		LINK=(vmid,pwd,accmode,ccuu) LINK = (userid,password,mode,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] [COMP XF NOCOMP NF]])
		(FROM) DISP = ([OLD SHR])
		LABEL =([file sequence number], [SL NL], [RETPD=nnnn EXPDT=yyddd])
		UNIT=(3480 TAPE) UNIT = (3480 3480X 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 (
		(TO) DSN = 'filename filetype' DSN = 'ISPOOL vmid fn ft' GROUP = '%1% ... %N%'
		LINK=(vmid,pwd,accmode,ccuu) LINK = (userid,password,mode,ccuu)
		VSAMCAT = (dsn,vmid,pwd,accmode,ccuu)
		PNODE SNODE
		SFSDIR=('dirid', [CRE , NOCRE])
		TYPE = typekey
		PROTECT = Yes No
		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]])
		(TO) DISP = [NEW OLD RPL SHR MOD]
		LABEL =([file sequence number], [SL NL], [RETPD=nnnn EXPDT=yyddd])
		UNIT=(3480 TAPE) UNIT = (3480 3480X TAPE)
		VOL = (,,[volume sequence number], [volume count], SER=volume serial number (list))
)
		SYSOPTS='!SPOOL[CLASS x] [DIST distcode * OFF] [FORM form OFF]'

Label	Statement	Parameters
		CKPT = [nK nM]
		COMPRESS [PRIMEchar=X'40' X'xx' C'c'] [EXTended] Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.
		OLDDATE

Connect:Direct for VM/ESA Run Job Statement

The RUN JOB statement enables a file to be punched to a RDR of a specified virtual machine. This sends EXEC-type job streams and data to a service machine running a product such as VMBATCH or CMSBATCH in a format that those products can interpret and execute.

Specify the file name to be punched in the DSN keyword. Specify link information for access to the disk that contains the file to be sent, using the syntax rules for the LINK parameter on the VM/ESA COPY statement. Specify the target virtual machine ID name in the BATCHID keyword.

The service machine and the disk file to be punched must be on the same system on the PNODE or SNODE. Otherwise, you must perform a copy 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.

The following is the Connect:Direct for VM/ESA Run Job statement format. Refer to [Sterling Connect:Direct for VM/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN JOB	(
		DSN= ' filename filetype'
		LINK=(userid,password,mode,ccuu)
		BATCHID=VM-ID-name
		PNODE SNODE
)

Connect:Direct for VM/ESA Run Task Statement

The RUN TASK statement attaches user programs, or subtasks, during Process execution. When a IBM Connect:Direct Process issues a RUN TASK statement, the IBM Connect:Direct Process waits until the subtask finishes before executing the next IBM Connect:Direct Process step.

The subtask can be attached at either the PNODE or SNODE.

You can pass a list of user parameters to the subtask from the RUN

TASK statement. 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 containing the program used with the RUN TASK statement must be defined with a Group Control System (GCS) Global Loadlib command.

The following is the Connect:Direct for VM/ESA Run Task statement format. Refer to [Sterling Connect:Direct for VM/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN TASK	(

Label	Statement	Parameters
		PGM=program-name
		PARM=(parameter [,parameter,...])
)
		PNODE SNODE

Connect:Direct for VM/ESA Submit Statement

The SUBMIT statement submits another IBM Connect:Direct Process from within an executing Process. The Process can be submitted to either the PNODE or the SNODE.

The submitted Process must reside on the node where the SUBMIT statement executes. This node is referred to as the SUBNODE.

The SUBMIT statement is not the same as the SUBMIT command. The SUBMIT statement parses special characters differently from the SUBMIT command. Refer to for a discussion of special character parsing. See the *IBM Sterling Connect:Direct for VM/ESA User's Guide* for SUBMIT command syntax and parameters.

The following is the Connect:Direct for VM/ESA Submit statement format. Refer to [Sterling Connect:Direct for VM/ESA Process Parameters](#) for more information.

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
		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

Connect:Direct for VM/ESA Conditional Statements

Conditional statements alter the sequence of IBM Connect:Direct Process execution based on the completion of the previous step in the Process. For example, if a file copy fails, the Process may call an external program to generate a console message and stop the Process. If the file copy succeeds, the Process continues with the next step.

The following is the Connect:Direct for VM/ESA Conditional statement format. Refer to [Sterling Connect:Direct for VM/ESA Process Parameters](#) for more information.

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

Connect:Direct for VM/ESA Symbol Statement

The SYMBOL statement creates symbolic substitution values.

The following is the Connect:Direct for VM/ESA Symbol statement format. Refer to [Sterling Connect:Direct for VM/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
optional	SYMBOL	&symbolic_name=variable-string

Connect:Direct for VM/ESA Process Parameters

BATCHID = VM-ID-name

specifies the target virtual machine ID name. This parameter is required.

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.

CKPT = [nK | nM]

specifies the byte interval for checkpoint support. This enables restart of interrupted transmissions at the last valid transmission point and reduces restart time.

K denotes thousands, and M denotes millions. A CKPT value of zero (0) stops automatic processing.

IBM Connect:Direct converts the value to a block boundary, and a data transmission checkpoint is taken at that position. Sequential files can be checkpointed.

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

specifies to compress the file data, reducing the amount of data transmitted during a file copy. The file is automatically decompressed at the destination.

Note: If the SNODE is B2B Integrator, only COMPRESS EXT is supported.

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

If compression is specified, IBM Connect:Direct 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 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 useful to specify this parameter when line transmission speeds are limited, CPU is available, and data is repetitive.

The following are valid options for EXTended:

- **CMPlevel** 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 window size 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
                  )
```

condition

specifies the type of comparison to be performed. Valid symbols, alternate symbols, and conditions are:

- **= 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.

The completion code from the RUN JOB statement is from the job submission only and is not the completion code of the submitted job.

COPY

identifies the COPY statement. This statement identifier is specified with either the FROM or TO parameter.

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 source and destination file allocation attributes.

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 specified 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 range is 18 bytes to 32,760 bytes. For BLKSIZE greater than 32760 bytes, you must use the DMGIOX64 exit described in the Using IBM Connect:Direct Exits chapter of the *IBM Sterling Connect:Direct for VM/ESA Administration Guide*.

DEN specifies the magnetic tape mode setting. The following table shows values for the DEN parameter for 7- and 9-track tape. Specifying the DEN and TRTCH values together selects a tape device for allocation by Connect:Direct for 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
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.

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. The BLKSIZE value does not need to be an even multiple of LRECL.

RECFM specifies the format of the records in the file. Valid values are F (Fixed), FB, V (Variable), or VB.

TRTCH specifies the magnetic tape mode setting. Specifying the TRTCH and DEN values together selects a tape device for allocation by Connect:Direct for VM/ESA. Valid options are:

- **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.
- **NOCOMP | NF** specifies the tape to create without compression.

DSN = 'filename filetype'

specifies the destination filename and file type. Enclose the filename in single quotation marks. This parameter is required.

DSN = 'fn ft [fm]'

specifies the filename (fn), file type (ft), and optional file mode (fm). This parameter is required.

The optional file mode is the mode of a minidisk accessed by the IBM 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, you must specify the file mode of the new Process.

If the Process has been edited, the DTF machine must re-access the disk where the Process resides.

EIF

is required for specifying the end of the IF THEN or IF THEN ELSE block of statements. There are no parameters.

ELSE

designates a block of IBM Connect:Direct statements that execute when the IF THEN condition is not satisfied. There are no parameters.

EXCLUDE = [(| generic | member | (startrange/stoprange)| list)]

excludes specific CMS files from a copy.

generic specifies a generic filename. For example, if CDV* is specified, all filenames beginning with CDV and having the specified file type are excluded from a copy.

You can override an excluded generic filename by specifying an individual filename in the SELECT parameter.

member excludes an individual filename. Its exclusion cannot be overridden.

startrange specifies the first name in an alphanumeric range of files to exclude.

stoprange specifies the last name in an alphanumeric range of files to exclude.

Separate the startrange and stoprange parameters with a slash (/). Do not specify generic filenames (*) as startrange or stoprange values. The first and last files specified in the range as well as all files between are excluded.

You can override an excluded range by specifying an individual filename in the SELECT parameter.

EXIT

bypasses all remaining steps within a Process. There are no parameters.

FROM

specifies the source file characteristics. This parameter is required.

(FROM) DISP = ((OLD | SHR))

specifies the source file status and what to do with the file during transmission. Values are:

- **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.

(FROM) DSN = 'filename filetype' | GROUP = '* * ... *'

specifies the source filename and file type, or VSAM filename. The filename and file type are verified based on the VM standard filename conventions.

Enclose the filename in single quotation marks if the copy is from a Connect:Direct for VM/ESA node or if the Connect:Direct for z/OS PDS member name contains special characters.

This parameter is required.

GOTO

moves to a specific step within a Process.

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 releases the Process. When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence.

No specifies that the Process executes as soon as possible. This 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 can be established by either another Process running on the PNODE or by 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.

IF THEN

specifies that the IBM Connect:Direct system executes a block of IBM 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.

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.

Labels must begin in column 1. The label is a 1-8 character alphanumeric string, with the first letter alphabetic only.

LABEL =([file sequence number], [SL|NL], [RETPD=nnnn|EXPDT=yyddd])

specifies tape label information.

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.

LINK = (userid,password,mode,ccuu)

specifies the disk where the CMS file is located. This parameter allows you to access the CMS file. This parameter is required.

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

userid specifies the 1-8 character owner ID for the CMS minidisk where the file is located.

password specifies the 1-256 character appropriate password for the CMS minidisk where the file is located. 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)
- **MW** (primary multiple access; alternate read/write only access).

When used with the TO parameter, the access modes are:

- **W** (primary read/write access)
- **M** (primary multiple access)
- **WR** (primary read/write access; alternate read only access)
- **MR** (primary multiple access; alternate read only access)
- **MW** (primary multiple access; alternate read/write only access).



CAUTION: MW access to CMS format disks can be destructive. You must guarantee that no other VM user, or IBM 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.

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 specified 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.

NEWNAME = new-name

specifies a new name to be given to a Process. The default value is the PROCESS statement label.

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 IBM Connect:Direct user ID is the same as the VM id. If the IBM Connect:Direct user ID is different from the VM user ID, the user is not notified.

userid specifies the VM user ID to receive Process completion messages.

NOREPLACE

specifies to not replace a set of existing files with the same name at the destination node.

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.

nn

specifies the numeric value used for completion code checking. If specified as X'nn', it is a hexadecimal value. Any other specification indicates 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 equaling 4 indicates a warning.

OLDDATE

specifies that the creation/last modified date and time of the file being transmitted is used to set the creation date and time of the file received.

If OLDDATE, the current date and time are used for the creation date and time of the received file.

Use the OLDDATE parameter for sequential file transfers between two Connect:Direct for VM/ESA systems, and transfers between a set of CMS files on Connect:Direct for VM/ESA to partitioned data sets (PDS) on Connect:Direct for z/OS systems.

PACCT = 'pnode-accounting-data'

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

PARM = (parameter [, parameter,...])

specifies the parameters passed to the subtask when that subtask is attached. These parameters are the actual parameters rather than a list of addresses. Specify null parameters by adjacent commas (,,).

The parameter list format 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 is specified. The value can be specified with a plus (+) or minus (-) sign. For example, H'-32'. If no sign is provided, plus is assumed.

F'value' specifies a full-word value. No length is specified. The value can be specified with a plus (+) or minus (-) sign. If no sign is provided, plus is assumed. For example, F'4096'.

PLn'value' specifies a packed value. The length is optional. The length specifies the size of the field in bytes and cannot be longer than 16.

If the length is not specified, the length of the value is used. If the length specified is longer than the value, the value is padded to the left with zeros. The value can be specified with a plus (+) or minus (-) sign. For example, PL10'+512'. If no sign is provided, 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.

Enclose strings comprised of symbolic substitution in double quotes. Use the data-type format if an ampersand (&) is passed as part of the parameter. For example, CL8'&PARM1' uses no substitution; CL8"&PARM1" indicates that the value is substituted.

PGM = program-name

specifies the name of the program that is attached as a subtask. This parameter is required.

The program runs on the specified node and has access to the DD statements allocated on that node only.

PNODE

On a COPY statement, specifies the primary node in a file copy. When specified with the FROM parameter, the file to be copied resides on the primary node. When specified with the TO parameter, the file is sent to the primary node. PNODE is the default with the COPY FROM parameter.

On a Run Job statement, specifies that the job is submitted on the primary node, which is the default value.

On a Run Task statement, specifies that the program will be executed on the primary node, which is the default.

PNODE = primary-node-name

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

The Process is always submitted is the PNODE. This parameter defaults to the name of the node submitting the Process and does not have to 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 used to sign on to IBM Connect:Direct.

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

pswd specifies the 1-8 alphanumeric character password for the specified ID. This parameter can be validated by the security system at the PNODE. This parameter is optional, unless the user's security requires a password.

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

PROcEss

identifies the PROCESS statement. This statement identifier can be abbreviated to PROC.

process name

specifies the 1-8 character name of the Process. The first character must be alphabetic and must start in column one. The PROCESS keyword must be on the same line as the Process name.

This label identifies the Process in any messages or statistics.

PROTECT = Yes | No

specifies whether an IBM RACF profile will be created for a new file.

Yes specifies that a RACF profile will be created for a newly transferred file.

No specifies that a RACF profile will not be created. No is the default.

The PROTECT parameter is valid only with the TO parameter of the COPY statement.

PRTY = n

specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ holds all Processes submitted to IBM Connect:Direct. The higher the number, the higher the priority.

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 IBM Connect:Direct initialization parameters.

REQUEUE = Yes | No

specifies whether a COPY step should be queued again if an x37 termination occurs during processing. This parameter is valid only if used when checkpointing.

Yes places the Process in the Hold queue with a status of HELD IN ERROR (HE). You can then take corrective action and restart the Process at the failing step. Checkpointing resumes at the last successful checkpoint. The Process must be explicitly released from the Hold queue when the status is HELD IN ERROR (HE).

No enables the Process to run to completion, executing subsequent steps when a COPY step fails with an abnormal termination. The default is NO.

REPLACE

specifies that the set of files replaces files of the same name on the destination node.

RETAIN = Yes | No | Initial

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

Yes keeps the Process in the Hold queue after initial execution. You must then do one of the following:

- Manually release the Process through the CHANGE PROCESS command to execute it
- Delete the Process through the DELETE PROCESS command
- Specify the STARTT parameter to release the Process again at a specified interval. Using RETAIN=YES with STARTT will run a Process 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. This is the default value.

Initial specifies that the Process executes every time IBM Connect:Direct is initialized. The Process will not execute when initially submitted. Do not specify STARTT with RETAIN=INITIAL.

RUN JOB

identifies the RUN JOB statement.

RUN TASK

identifies the RUN TASK statement.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. Enclose the string in single quotation marks if it contains special characters. 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 for i5/OSnode.

SELECT = [(|member | generic | (*) | (member, [newname], [NR|R]) | (generic,,[NR|R]) (startrange/ stoprange,, [NR|R]) | list)]

selects files in a set of CMS files for copying.

generic specifies a generic filename. For example, if CDV* is specified, all filenames beginning with CDV and with the specified file type are selected to be copied.

(*) represents a global generic. A global generic indicates that all files in the set of files are included in the copy.

You can override a generic selection with the EXCLUDE parameter.

If you use a generic and specify **NR** or **R** as the third positional parameter, the second positional parameter must be null. For example, **SELECT = (CDV*, , R)**.

member specifies an individual filename. This is the same as specifying a filename in the DSN.

You can override a filename selection with the EXCLUDE parameter.

newname specifies a new name for a file. If you use a generic name or range is used as the first positional parameter, the newname parameter must be null.

NR specifies that a file will not replace an existing file with the same name at the destination. **NR** overrides the REPLACE parameter.

When used with the newname parameter, NR only applies to the newname and not to the original filename. When used with a generic name or with a range, **NR** applies to all selected files. **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 destination. **R** overrides the NOREPLACE option. **R** is the default.

When used with the newname parameter, **R** only 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 to select.

stoprange specifies the last name in an alphanumeric range of files to select.

Separate the startrange and stoprange parameters with a slash (/). Do not specify generic filenames (*) as startrange or stoprange values. The first and last files specified in the range as well as all files between are selected.

You can override a selected range by specifying an individual filename in the EXCLUDE parameter.

If you specify a file range, with the NR or R option, you must leave the second positional parameter as null. For example, **SELECT = (file1/file99, , R)**.

SFSDIR=(*dirid*, [CRE , NOCRE])

specifies the location of the Shared File System (SFS) managed file.

dirid specifies the directory name where the SFS file resides. The maximum length is 153 characters. The minimum specification for dirid is 'poolid:userid.'

CRE creates a new or extend an existing subdirectory, if the user has creation authority. This is the default.

NOCRE neither creates a new subdirectory nor extends an existing subdirectory.

You cannot specify the SFSDIR and the LINK parameters for the same file.

SNODE

On a Copy statement, specifies the secondary node. When SNODE is specified with the FROM parameter, the file to be copied resides on the secondary node. When SNODE is specified with the TO parameter, the file is sent to the secondary node.

On a Run Job statement, specifies that the job is submitted on the secondary node.

On a Run Task statement, specifies that the subtask will be attached on the secondary node.

SNODE = secondary-node-name

is a 1-16 character alphanumeric name that specifies the secondary node (SNODE) used in the Process. The name can be alphanumerics or nationals (@ # \$) with embedded periods. This parameter is required for the PROCESS statement, unless it is specified in the SUBMIT statement.

This is the logical node name that has been defined in the ADJACENT.NODE entry for that node in the network map. The PNODE and SNODE can specify the same symbolic node name.

When used in the SUBMIT statement, this parameter overrides the value specified in the PROCESS statement. The default value for SNODE is the value specified in the PROCESS statement.

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

specifies security user IDs and passwords at the SNODE.

For Connect:Direct for i5/OS, if the Process submitter's SNODEID and password is not specified in the PROCESS statement, the Process submitter's user ID and password is used for the Connect:Direct for i5/OS security ID and password check.

id specifies the 1-8 character security ID passed to the security system on the SNODE.

For Connect:Direct for HP NonStop, this subparameter specifies the HP NonStop group number and user number. These numbers can range from 0-255. Use a period as a separator between the group number and the user number.

For Connect:Direct for i5/OS, this subparameter specifies the i5/OS user profile used for authorization checks during Process execution. It is limited to 8 characters even though i5/OS user profiles may be 10 characters long.

pswd specifies the current 1-8 character security password and can be used by the SNODE security system to validate the security password. This is optional unless the user's security requires a password.

For Connect:Direct for HP NonStop, the VM node only recognizes passwords specified in uppercase alphanumeric characters. A Process cannot be successfully initiated from Connect:Direct for VM/ESA with Connect:Direct for HP NonStop unless the Connect:Direct for HP NonStop SNODEID password only contains uppercase alphanumeric characters (no control characters).

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

For Connect:Direct for HP NonStop, SAFEGUARD must be running on HP NonStop.

For Connect:Direct for i5/OS, 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.

Do not specify STARTT with RETAIN=INITIAL.

If you specify both HOLD=YES and a STARTT value in a Process, the HOLD specification takes precedence, and the Process is placed in the Hold queue.

date specifies that the Process starts on a specific date. Depending on the value of the DATEFORM initialization parameter, you can specify the date in one of the following formats:

DATEFORM Value	Formats			
DATEFORM=MDY (default)	mm/dd/yy	mm/dd/yyyy	mm.dd.yy	mm.dd.yyyy
DATEFORM=DMY	dd/mm/yy	dd/mm/yyyy	dd.mm.yy	dd.mm.yyyy
DATEFORM=YMD	yy/mm/dd	yyyy/mm/dd	yy.mm.dd	yyyy.mm.dd
DATEFORM=YDM	yy/dd/mm	yyyy/dd/mm	yy.dd.mm	yyyy.dd.mm

You can use periods or slashes (/) to separate the components of a date value. You can omit the period or slash separators for transfers between mainframe nodes.

If you only specify a date, the time defaults to 00:00.

If you specify RETAIN=YES, you cannot specify a date in the STARTT parameter.

Valid Julian date formats are yyddd, yyyyddd, yy/ddd, yyyy/ddd, yy.ddd, or yyyy.ddd.

day specifies the day of the week to release the Process for execution. Valid names include MONday, TUesday, WEdnesday, THursday, FRiday, SAaturday, and SUNday. You can abbreviate the day value 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.

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.

You can specify the time of day using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00:00 to 24:00:00. If you do not use AM and PM, the 24-hour clock is assumed.

If you use the 12-hour clock, 01:00:00 hours can be expressed as 1:00AM, and 13:00 hours can be expressed as 1:00PM.

If you specify hh:mm:ssXM with RETAIN=YES, the Process executes at the same time every day. You do not need to specify minutes and seconds.

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

stepname

specifies the user-defined name of the Copy, Run Job, Run Task, or Submit step.

Stepnames must begin in column one. Stepnames are 1-8 character alphanumeric strings. The first character must be alphabetic.

SUBMIT

identifies the SUBMIT statement.

SUBNODE = PNODE | SNODE

specifies the node where the Process defined in a SUBMIT statement will execute. PNODE indicates that the Process is submitted on the node that has Process control. SNODE indicates 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.

SYMBOL

identifies the SYMBOL statement.

&symbolic_name=variable-string

specifies a string that is substituted into the Process.

When IBM Connect:Direct encounters an ampersand (&) followed by 1-17 alphanumeric characters, IBM 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 IBM Connect:Direct statement string can be concatenated, enabling you to move data in a variety of ways.

Specify a null value by an equal sign (=) immediately followed by a comma. Enclose a symbolic parameter containing special characters in single quotation marks.

&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 value in the SUBMIT command.

Enclose a symbolic parameter containing special characters in single quotation marks. Specify a null value by the equal sign (=) immediately followed by a comma.

The symbolic parameter for the SUBMIT statement must begin with a single ampersand. This allows correct symbolic parameters resolution.

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.

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'
```

THEN

specifies the subsequent processing to perform if the specified condition is met.

TO

specifies the destination file characteristics. This parameter is required.

(TO) DISP = [NEW|OLD|RPL|SHR|MOD]

specifies the destination file disposition. Values are:

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

Only OLD and RPL apply to VSAM files.

(TO) DSN = 'filename filetype' | DSN = '!SPOOL vmid fn ft' | GROUP = '%1% ... %N%'

specifies the destination filename and file type, or VSAM filename.

If copying the file to a Connect:Direct for VM/ESA node, enclose the filename in single quotation marks.

This parameter is required.

Connect:Direct for VM/ESA can be used to spool files to a specific virtual reader. This is useful for downloading files, because IBM 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 file type of the spooled file:

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

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

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

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

TYPE = typekey

specifies the member name of the type defaults file. This file contains the file attribute defaults used to open the destination file.

UNIT = (3480 | 3480X | TAPE)

specifies the device type. Acceptable values are `3480`, `3480X` and `TAPE`.

3480X specifies that the unit type is an IDRC-compatible drive.

VOL = (, [volume sequence number], [volume count], SER=volume serial number | (list))

specifies the volume serial numbers 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 volumes where 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 you use 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)
- **MW** (primary multiple access; alternate read/write only access).



CAUTION: MW access to CMS format disks can be destructive. You must guarantee that no other VM user, or IBM 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.

VSE/ESA Statements and Parameters

Connect:Direct for VSE Process Statement

The PROCESS statement defines the attributes of a Process and is always the first statement in a Process.

The following is the Connect:Direct for VSEProcess statement format. Refer to [Sterling Connect for VSE/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
process name	PROcEss	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])
		&symbolic_name_1=variable-string-1 &symbolic_name_2=variable-string-2 . . . &symbolic_name_n=variable-string-n

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.

Connect:Direct for VSE Copy Statement

The COPY statement copies files from one node to another. The Connect:Direct for 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.

The COPY statement contains a FROM parameter that specifies the source file name and a TO parameter that specifies the destination file name. You can specify additional parameters to further the file transfer operation.

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

To copy from one IBM Connect:Direct platform to another, refer to the appropriate COPY FROM and TO descriptions for those platforms. For example, if the source file is on a Connect:Direct for HP NonStop node, refer to the COPY FROM description for Connect:Direct for HP NonStop. If the file destination is a Connect:Direct for VSE node, refer to the Connect:Direct for VSE COPY TO description.

The following is the Connect:Direct for VSE COPY statement format. Refer to [Sterling Connect for VSE/ESA Process Parameters](#) for more information.

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)] [*])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))
		UNIT=([group name device-type unit address DLBLOONLY TNOASGN])
		UNIT=([group name device-type unit address DLBLOONLY TNOASGN])
		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)"

Label	Statement	Parameters
)
		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 RPL SHR], [KEEP CATLG])
		LABEL=([file-sequence-number] [,SL AL BLP LTM NL] [,PASSWORD NOPWREAD], [IN OUT] [,RETPD = nnnn EXPDT = yyddd yyyy/ddd])
		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] [*])

Label	Statement	Parameters
		<pre> 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]) </pre>

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))]
		UNIT=([group name device-type unit address DLBONLY 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
)

Label	Statement	Parameters
		CKPT=nK nM
		COMPRESS [PRIMEchar=X'40' X'xx' C'c' EXTended] Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

Connect:Direct for VSE Run Job Statement

The RUN JOB statement submits a job through the VSE virtual reader, which is a facility that transfers jobs to VSE/POWER. The job must reside in a file on the node that executes the RUN JOB statement.

IBM Connect:Direct does not verify job statements. To determine the completion status of a RUN JOB statement, check the IBM Connect:Direct statistics records.

The following is the Connect:Direct for VSE Run Job statement format. Refer to [Sterling Connect for VSE/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN JOB	(DSN = member-type(member) member)
		PNODE SNODE

Connect:Direct for VSE Run Task Statement

The RUN TASK statement attaches user programs or subtasks during Process execution. When a IBM Connect:Direct Process issues a RUN TASK statement, the IBM Connect:Direct Process waits until the subtask finishes before executing the next IBM Connect:Direct Process step.

You can pass a list of user parameters to the subtask from the RUN TASK statement. 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 subtask can be attached at either the PNODE or SNODE. The subtask must reside in a load library that can be accessed by the IBM Connect:Direct DTF.

The following is the Connect:Direct for VSE Run Task statement format. Refer to [Sterling Connect for VSE/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	RUN TASK	(
		PGM=program-name
		PARM=(parameter [,parameter,...])
)
		PNODE SNODE

Connect:Direct for VSE Submit Statement

The SUBMIT statement submits another IBM Connect:Direct Process from within an executing Process. The Process can be submitted to either the PNODE or the SNODE.

The submitted Process must reside on the node where the SUBMIT statement executes. This node is referred to as the SUBNODE.

The SUBMIT statement is not the same as the SUBMIT command. The SUBMIT statement parses special characters differently from the SUBMIT command. Refer to [Sterling Connect for VSE/ESA Process](#)

Parameters for a discussion of special character parsing. See the *IBM Sterling Connect for VSE/ESA User's Guide* for SUBMIT command syntax and parameters.

The following is the Connect:Direct for VM/ESA Submit statement format. Refer to [Sterling Connect for VSE/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	SUBMIT	DSN=mt(membername)
		NEWNAME=new-name
		SUBNODE=PNODE SNODE
		SNODE=secondary-node-name
		SACCT=(snode-accounting-data)
		PNODEID=(id [,pswd] [,newpswd])
		SNODEID=(id [,pswd] [,newpswd])
		PACCT=(pnode-accounting-data)
		CASE=Yes No
		CLASS=n
		HOLD=Yes No Call
		PRTY=n
		QUEUE=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

Connect:Direct for VSE/ESA Conditional Statements

Conditional statements alter the sequence of IBM Connect:Direct Process execution based on the completion of the previous step in the Process. For example, if a file copy fails, the Process may call an external program to generate a console message and stop the Process. If the file copy succeeds, the Process continues with the next step.

The following is the Connect:Direct for VSE Conditional statement format. Refer to [Sterling Connect for VSE/ESA Process Parameters](#) for more information.

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

Label	Statement	Parameters
	EXIT	

Connect:Direct for VSE Symbol Statement

The SYMBOL statement creates symbolic substitution values.

The following is the Connect:Direct for VSE Symbol statement format. Refer to [Sterling Connect for VSE/ESA Process Parameters](#) for more information.

Label	Statement	Parameters
optional	SYMBOL	&symbolic_name=variable-string

Connect:Direct for VSE/ESA Process 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 uses more memory.

CASE = Yes | No

specifies whether parameters associated with accounting data, user ID, password, and data set name are case sensitive. The default is No.

CKPT = nK | nM

specifies the byte interval for checkpoint support. This enables restart of interrupted transmissions at the last valid transmission point and reduces restart time.

K denotes thousands, and M denotes millions. A CKPT value of zero (0) stops automatic processing.

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

Checkpointing is controlled by the SNODE. IBM Connect:Direct does not support checkpoint/restart of VSAM-managed SAM files.

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.

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

specifies to compress data, reducing the amount of data transmitted during a file copy. The file is automatically decompressed at its destination.



CAUTION: Compression is CPU-intensive, and its effectiveness is data-dependent.

If compression is specified, IBM 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.

Note: If the SNODE is B2B Integrator, only COMPRESS EXT is supported.

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

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. 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 to allocate to maintain the internal compression state. This memory is above the line. 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)
```

condition

specifies the type of comparison to be performed. Valid symbols, alternate symbols, and conditions are:

- **= or EQ** specifies that the completion code must be equal to the value nn to satisfy the condition
- **<> or ≠ or NE** specifies that the completion code must not equal the value nn to satisfy the condition
- **>= or ≥ or GE** specifies that the completion code must be greater than or equal to the value nn to satisfy the condition
- **> or GT** specifies that the completion code must be greater than the value nn to satisfy the condition
- **<= or ≤ or LE** specifies that the completion code must be less than or equal to the value nn to satisfy the condition
- **< or LT** specifies that the completion code must be less than the value nn to satisfy the condition

The completion code from the RUN JOB statement is from the job submission only and is not the completion code of the submitted job.

COPY

identifies the COPY statement. This statement identifier is specified with either the FROM or TO parameter.

```
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 used when allocating source and destination files.

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

Note: You do not have to specify a DCB parameter when the input data set is VSE/POWER. IBM 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. IBM 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 following table shows values for the DEN parameter for 7- and 9-track tape. When specified together, the DEN and TRTCH values select a tape device for allocation by Connect:Direct for 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.

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 is used, the BLKSIZE value must be at least the LRECL value plus another 4 bytes. If RECFM=VB is used, the BLKSIZE value does not need to be an even multiple of LRECL.

An entry-sequenced file coming from HP NonStop to an IBM PS/VB must be specified with an LRECL 4 bytes larger than the HP NonStop 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 HP NonStop.

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.

An OpenVMS file with a record format of undefined (U) cannot be copied to VSE.

TRTCH specifies the magnetic tape mode setting. When specified together, the TRTCH and DEN values select a tape device for allocation by Connect:Direct for 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.

DSN = member-type(member) | member

specifies the name of a member in the LIBDEF source chain that contains the job to be submitted. This parameter is required.

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.

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 IBM Connect:Direct JCL. This parameter is required.

Specify **mt** as a single character A-Z, 0-9, \$, #, or @.

membername is the name of the member in the Process library.

DSN = 'DLBLNAME = filename'

allocates the COPY FROM dataset by using the VSE DTF (Define The File) name rather than the data set name.

DLBLNAME is a required keyword.

filename is the 1–7 character VSE DTF name. The first and last character must be alphabetic, @, # or \$ of a previously allocated file.

Enclose the parameter string in single quotes ('**DLBLNAME = filename**').

IBM Connect:Direct allocates the file-id (data set name) associated with the DLBLNAME (filename) information obtained in the VSE partition standard label area.

This parameter provides the greatest benefit for Processes submitted through DMBATCH where partition/processor independent data set names are generated using CA-DYNAM/D, CA-EPIC or BIM-EPIC.

The filename associated with the 'DLBLNAME =filename' parameter must reside in the same partition standard label area as the other IBM Connect:Direct DLBL statements. Prior to Process submission, the file-id (dataset name) must have been opened and allocated in a previous job or step. The file-id in the partition standard label area must contain a fully-qualified data set name (all symbolic or substitution characters in the file-id must be resolved).

EIF

specifies the end of the IF THEN or IF THEN ELSE block of statements. It is required.

ELSE

designates a block of IBM Connect:Direct statements that execute when the IF THEN condition is not satisfied.

EXIT

bypasses all remaining steps within a Process.

FROM

specifies the source file characteristics. This parameter is required.

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

specifies the status of the file and what to do with the file after successful transmission.

Subparameters are:

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 a non-VSAM 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 file after the Process step completes.
- **DELETE** specifies that the system deletes the file after the Process step completes successfully.

(FROM) DSN =

specifies the source data set name. This parameter is required. (DSN is optional when used with the IOEXIT parameter.)

Data set names are verified according to standard VSE data set name conventions. If the data set name does not follow VSE naming conventions, enclose the data set name in single quotation marks.

GDG data sets (controlled by Epic or DYNAM) can be copied by either the relative generation number or the absolute generation number.

Connect:Direct for 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, specify the password in the COPY statement in the format **datasetname/ password** (separate the data set name and password with a slash). For example:

```
COPY FROM DSN=datasetname/password...
```

The password is used at data set allocation. If the password is not correct, VSE issues a WTOR requesting the password when Connect:Direct for VSE software opens the data set.

GOTO

moves to a specific step, defined by the label, within a Process.

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 releases the Process. When both HOLD=YES and a STARTT value are specified, the HOLD specification takes precedence.

No specifies that the Process executes as soon as possible. This 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 can be established by either another Process running on the PNODE or by 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.

IF THEN

specifies that IBM Connect:Direct execute a block of statements based on the completion code of a Process step. You use an EIF statement with an IF THEN statement. A return code with the high order bit on is evaluated as a negative return code.

IOEXIT = exit-name | (exit-name| ,parameter,...)

calls a user-written program to perform I/O requests for the associated data.

exit-name specifies the name of the user-written program to call.

parameter specifies a parameter, or list of parameters, to pass to the specified exit. See the RUN TASK parameters for the appropriate platform for parameter formats.

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. The GOTO statement requires a label. The label name cannot be the label of a preceding step nor the label of the GOTO statement of which it is a part.

Labels must begin in column one. The label consists of a 1-8 character alphanumeric string, with the first letter alphabetic only.

**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)]
[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 which members, sublibraries or types are selected or excluded from a copy.

EXCLMEM=(generic | mem | start-range/stop-range | list) excludes specified members from a copy.

EXCLSLIB=(generic | sublib | start-range/stop-range | list) excludes specified sublibraries from a copy.

EXCLTYPE=(generic | type | start-range/stop-range | list) excludes specified types from a copy.

All three exclusions use the following subparameters:

- **generic** excludes a generic member, sublibrary or type name. For example, if you specify EXCLMEM=CDM*, all member names beginning with CDM are excluded. You can override an excluded generic by specifying an individual member, sublibrary or type in the SELMEM, SELSLIB, or SELTYPE parameter.
- **mem | sublib | type** excludes an individual member, sublibrary or type name. You cannot override it.
- **startrange** specifies the first name in an alphanumeric range of members, sublibraries or types to exclude.
- **stoprange** specifies the last name in an alphanumeric range of members, sublibraries or types to exclude.

Separate the startrange and stoprange parameters with a slash (/). Do not specify generic names (*) as startrange or stoprange values. The first and last members, sublibraries or types specified in the range as well as all members, sublibraries or types between are excluded.

You can override an excluded range by specifying an individual member, sublibrary or type name in the SELMEM, SELSLIB, or SELTYPE parameter.

- **list** excludes a list of member, sublibrary or type names.

REPLACE= YES | NO specifies whether to replace the member with the same name in the target library or PDS.

SELMEM=member | generic | * | (member, [new-name], [NR | R]) | (generic,, [NR | R]) | (start-range/stop-range,, [NR | R]) | (list) selects specified members for a copy.

SELSLIB=sublibrary | generic | * | (sublibrary, [new-name], [NR | R]) | (generic,, [NR | R]) | (start-range/stop-range,, [NR | R]) | (list) selects specified sublibraries for a copy.

SELTYPE=type | generic | * | (type, new-name) | (start-range/stop-range) | (list) selects specified types for a copy.

These selections use the following subparameters:

- **generic** selects a generic member, sublibrary or type. For example, if CDM* is specified, all members, sublibraries or types beginning with CDM are selected for copying. You can override a generic selection by specifying an individual member, sublibrary or type in the EXCLMEM, EXCLSLIB, or EXCLTYPE parameter.
- An **asterisk (*)** represents a global generic. A global generic indicates that all members, sublibraries or types of the library are to be included.
- **member | sublibrary | type** specifies an individual member, sublibrary, or type. You can override this selection by specifying the individual member, sublibrary or type in the EXCLMEM, EXCLSLIB, or EXCLTYPE parameter.

- **newname** specifies a new name for the member, sublibrary or type.
- **NR** specifies that the member or sublibrary does not replace an existing member or sublibrary of the same name at the destination VSE Library or PDS. NR overrides the REPLACE and REPLACE=YES parameters.

You cannot specify NR on SELTYPE.

Note: NR applies to the member or sublibrary. REPLACE=NO (or NOREPLACE) applies to an entire VSE Library or PDS.

- **R** specifies that the member or sublibrary replace an existing member or sublibrary at the destination VSE Library or PDS. R overrides the REPLACE=NO or NOREPLACE parameter.

You cannot specify R on SELTYPE.

When you use R with the NEWNAME subparameter, the replacement applies to the new file name and not to the original name. When you use R with a generic name or with a range, R applies to all members selected for that criteria.

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

- **startrange** specifies the first name in an alphanumeric range of members, sublibraries or types to select.
- **stoprange** specifies the last name in an alphanumeric range of members, sublibraries or types to select.

Separate the startrange and stoprange parameters with a slash (/). Do not specify generic names (*) as startrange or stoprange values. The first and last members, sublibraries or types specified in the range as well as all members, sublibraries or types between are selected.

You can override an selected range by specifying an individual member, sublibrary or type name in the EXCLMEM, EXCLSLIB, or SEXCLTYPE 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]v [TYPE=type] [*])

specifies how the copied members are catalogued.

DATA= NO| YES specifies if the DATA=YES flag is set on catalogued members. This flag only applies to type PROC members. Default value is NO.

LOCKID=lockid specifies whether to lock catalogued members with the specified lock ID. If not specified, the members are not locked, unless the source member is also a VSE Library member and it was originally locked.

MSHP NO| YES | OVERRIDE specifies if catalogues members are flagged as under MSHP control. MSHP=OVERRIDE specifies that the members are flagged as under MSHP control and 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. Default value is NO.

REPLACE= YES| NO specifies whether the existing member should be replaced in the target VSE Library. Default value is YES.

RESETLOCK= NO| YES specifies to replace an existing VSE Library even if it contains locked members. Default value is NO.

REUSE= AUTOMATIC | IMMEDIATE specifies the space attribute is set when a sublibrary is defined. Default value is AUTOMATIC.

SLIBDISP= SHR | OLD | NEW | RPL | MOD specifies the disposition of the sublibrary for the member being cataloged. Specify SLIBDISP=RPL to define the sublibrary as empty even if it currently exists. Specify SLIBDISP=MOD to define the sublibrary only if it does not currently exist and you do not want the current members deleted if the sublibrary does exist. Default value is SHR.

SUBLIB=sublibrary specifies the target sublibrary for the members being cataloged. This parameter is required when the source is not a VSE Library.

TYPE=type specifies the target member type for the members being cataloged. This parameter is required when the source is not a VSE Library.

***** specifies to take the Connect:Direct for VSE Library defaults 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 is 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 the disposition 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 for 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 is printed.

For more detailed explanation of the following parameters, see the *IBM VSE/POWER Administration and Operation Manual*.

BANNER=(literal1=value1, literal2=value2,...) specifies to print a IBM Connect:Direct banner page at the beginning of the LST queue entry. Banner lines are specified as *literal=value*. The *literal* is 1-25 characters and the *value* is 1-30 characters. Literals or values that exceed the maximum length are truncated. Both can be either literal or symbolic parameters.

The following example prints a banner page with three lines of user supplied information:

```
BANNER=(PROGRAMMER=&PROGR, JOBNAME=&JOBNM, &SUBMITTER=JDOE1)
```

The banner page generated would look like the following:

```

***** 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.

BURST=YES | NO specifies the 3800 specification for the burster/trimmer/stacker for this LST queue entry.

CC= M | ASA | (NOCC[,TOF=1 | char | x'xx'], [LINECT=55 | nn]) specifies the type of control characters 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. Default value of CC is M. Default value of TOF is 1. Default value of LINECT is 55.

CHARS=(tablename, tablename) specifies the 3800 character arrangement table names.

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 specify the z/OS 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 z/OS JCL manual.

CLASS=class specifies the class of the LST or PUN queue entry. Valid classes are A-Z, 0-9.

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.

CONTROL=program single double triple specifies the z/OS 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 z/OS JCL manual.

COPIES=nnn | COPY=nnn specifies the number of copies to print or punch for the LST or PUN queue entry

DEFAULT=YES | NO specifies whether or not the 3800 default values are used from the installation SETDF.

DEPT='department-identification' specifies the department identification for the LST or PUN queue entry.

DEST=nodename (nodename, userid) specifies the node name and user ID to associated with this LST or PUN queue entry. Specifying the node name places the LST or PUN queue entry in the XMT queue.

DFLT=YES | NO specifies whether or not the 3800 default values will be used from the installation SETDF.

DISP=D | K | H | L specifies the VSE/POWER disposition 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 used when the LST queue entry is physically printed.

FLASH=(overlay-name, count) specifies the 3800 printer flash overlay name and the number of pages flashed.

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.

HOLD=YES | NO specifies the VSE/POWER disposition for the LST or PUN queue entry. This keyword is provided for compatibility with z/OS. **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 printed on each page for the 3800 LST queue entry.

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.

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.

PRTY=n specifies the priority for the LST or PUN queue entry. Valid VSE/POWER priorities are 0-9.

PWD=password specifies the password associated with the LST or PUN queue entry. If not specified, the master password (or binary zeros) will be used as the password.

ROOM='room-identification' specifies the room identification for the LST or PUN queue entry.

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.

SYSID=n specifies the system id for a VSE/POWER Shared Spool environment for this LST or PUN queue entry.

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.

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.

USER='user-data-description' specifies up to 16-bytes of user data for the LST or PUN queue entry.

USERID=userid specifies the user ID associated with this LST or PUN queue entry. Specifying the node name places the LST or PUN queue entry in the XMT queue.

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.

NEWNAME = new-name

specifies a new name for the Process. The default value is the label on the PROCESS statement.

nn

specifies the numeric value used for completion code checking. If specified as X'nn', it is a hexadecimal value. Any other specification indicates 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 equaling 4 indicates a warning.

%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 Process submission time in a six-digit numeric-value format (minutes, seconds, fraction of seconds). Precede the %NUM1 value with 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. Enclose the string in single quotation marks if it contains special characters. This data overrides any accounting data specified on the SIGNON command and can be used by a user-written program or statistics exit.

PARM = (parameter [, parameter,...])

specifies the parameters passed to the subtask when the subtask is attached. These are the actual parameters rather than a list of addresses. Specify null parameters by adjacent commas.

The parameter list format is a 2-byte field, indicating the length of the parameter, followed by the parameter itself. The valid data types for the PARM parameter are:

CLn'value' specifies a character data type with a length of n bytes. The length is optional. If a length is not specified, the actual length of the value is used. If you specify a length less than the actual value length, the data is truncated. If you specify a length longer than the actual value length, the value is padded with blanks on the right.

For example, CL44'FILE.NAME'

XLn'value' specifies a hexadecimal data type with a length of n bytes. The length is optional. If a length is not specified, the actual length of the value is used. If you specify a length less than the actual value length, the data is truncated. If you specify a length longer than the actual value length, the value is padded on the left with binary zeros.

For example, XL8'FF00'

H'value' specifies a half-word value. You cannot specify a length. Specify the value with a plus (+) or minus (-) sign. If no sign is specified, plus is assumed.

For example, H'-32'

F'value' specifies a full-word value. You cannot specify a length. Specify the value with a plus (+) or minus (-) sign. If no sign is given, plus is assumed.

For example, F'4096'

PLn'value' specifies a packed value with a length of n bytes. The length is optional, but cannot be longer than 16 bytes. If the length is not specified, the actual length of the value is used. If the length specified is longer than the actual value length, the value is padded on the left with zeros.

You can specify the value with a plus (+) or minus (-) sign. If no sign is given, plus is assumed.

For example, PL10'+512'

If you do not specify a data type or length, the character data type is assumed and the length of the parameter name is used. For example, if PARM=('FILE.NAME') is specified, the length used is 9.

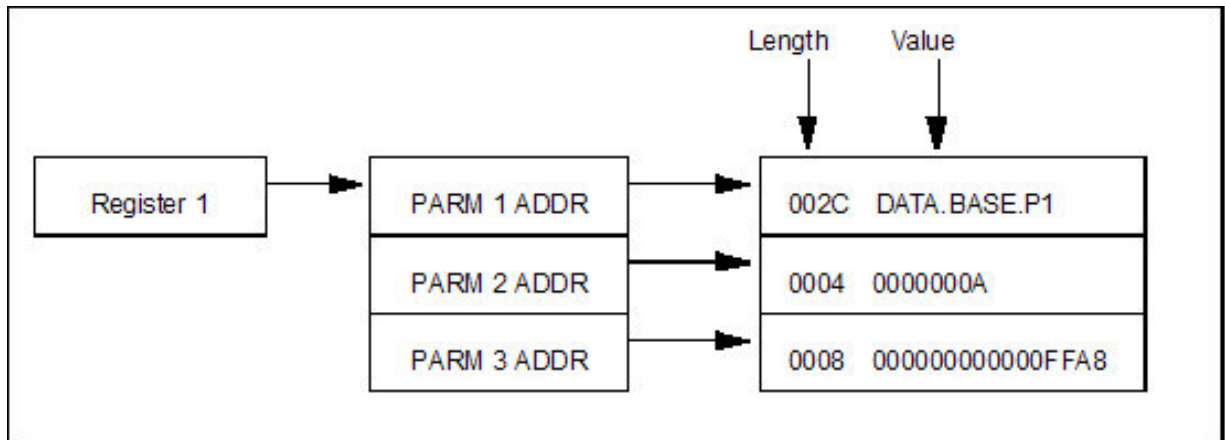
You can also specify the parameter as a symbolic value that is resolved when the Process is submitted. Do not specify a data type or length if you use a symbol. For example, &PARM1

When using strings comprised of symbolic substitution, enclose the strings in double quotes. If an ampersand (&) is passed as part of the parameter, use the data-type format. For example, CL8'&PARM1' uses no substitution; CL8"&PARM1" indicates that the value is substituted.

The following example shows how parameters are passed. This example runs the program named MYTASK. It attaches to the Process on the 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 following figure shows the parameter passing convention for MYTASK. Register 1 points to a parameter list of three parameters. It would contain 0 if no parameters were specified. IBM Connect:Direct sets the high-order bit in PARM 3 ADDR to indicate the end of the PARM list.



PGM = program-name

specifies the name of the program to be attached as a subtask. This parameter is required. The program runs on the node specified and has access to the DLBL cards allocated on that node only.

PNODE

On a Copy statement, specifies the primary node. When you specify PNODE with the COPY FROM parameter, a send takes place. When you specify PNODE with the COPY TO parameter, a receive takes place.

PNODE is the default for the FROM parameter.

On the Run Job statement, specifies that the job is to be submitted on the primary node.

On a Run Task, specifies that the program is executed on the PNODE.

PNODE = primary-node-name

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

The Process is always submitted as the PNODE. This parameter defaults to the name of the node submitting the Process and does not have to 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 used to sign on to IBM Connect:Direct.

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

pswd specifies the 1-8 alphanumeric character password for the specified ID. This parameter can be validated by the security system at the PNODE. This parameter is optional, unless the user's security requires a password.

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

PROcEss

identifies the PROCESS statement. You can abbreviate the statement identifier to PROC.

process name

specifies the 1-8 character name of the Process. The first character must be alphabetic and must start in column one. The PROCESS keyword must be on the same line as the Process name.

This label identifies the Process in any messages or statistics.

PRTY = n

specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ holds all Processes submitted to IBM Connect:Direct. The higher the number, the higher the priority.

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 IBM Connect:Direct initialization parameters.

**(FROM) PUN=(
[CLASS=class]
[DISP=D | K | H | L]
[PWD=password])**

specifies how the PUN queue entry is 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 for 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 is processed.

For more detailed explanation of the following parameters, see the *IBM VSE/POWER Administration and Operation Manual*.

BLDG=building specifies the building identification for this LST or PUN queue entry.

CC=M | ASA | NOCC specifies the type of control characters provided with the data. If NOCC is specified, the data is placed in the VSE/POWER LST or PUN queue with ASA control characters. Default value of cc is M.

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 z/OS 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 z/OS JCL manual.

CLASS=class specifies the class of the LST or PUN queue entry. Valid classes are A-Z, 0-9.

CONTROL=program single double triple specifies the z/OS 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 z/OS JCL manual.

COPIES=nnn | COPY=nnn specifies the number of copies to be printed or punched for the LST or PUN queue entry.

DEPT='department-identification' specifies the department identification for the LST or PUN queue entry.

DEST=nodename (nodename,userid) specifies the nodename and/or the user ID 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.

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. .

HOLD=YES | NO specifies the VSE/POWER disposition for the LST or PUN queue entry. This keyword is provided for compatibility with z/OS. 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.

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.

PRTY=n specifies the priority for the LST or PUN queue entry. Valid VSE/POWER priorities are 0-9.

PWD=password specifies the password associated with the LST or PUN queue entry. If not specified, the master password (or binary zeros) is 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.

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.

SYSID=n specifies the system ID for a VSE/POWER Shared Spool environment for this LST or PUN queue entry.

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.

USERID=userid specifies the user ID associated with this LST or PUN queue entry. .

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.

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 places the requeued Process in the Hold queue with a status of HELD IN ERROR (HE). After you take corrective action, the Process will restart with the failing step. Checkpointing resumes at the last successful checkpoint. You must explicitly release the Process from the Hold queue when the status is HELD IN ERROR (HE).

No allows a Process to run to completion, even if a COPY step fails with an abend. This is the default.

RETAIN = Yes | No | Initial

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

Yes keeps the Process in the Hold queue after initial execution. You must then do one of the following:

- Manually release the Process through the CHANGE PROCESS command to execute it
- Delete the Process through the DELETE PROCESS command
- Specify the STARTT parameter to release the Process again at a specified interval. Using RETAIN=YES with STARTT will run a Process 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. This is the default value.

Initial specifies that the Process executes every time IBM Connect:Direct is initialized. The Process will not execute when initially submitted. Do not specify STARTT with RETAIN=INITIAL.

RUN JOB

identifies the RUN JOB statement.

RUN TASK

identifies the RUN TASK statement.

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. Enclose the string in single quotation marks if it contains special characters. 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 for i5/OS node.

SNODE

On a Copy statement, specifies the secondary node. When you specify SNODE with the COPY FROM parameter, a receive takes place. When you specify SNODE with the COPY TO parameter, a send takes place.

On a Run Job statement, specifies that the job is submitted on the secondary node.

On a Run Task statement, specifies that the subtask is attached on the SNODE. The program must exist as an executable module in the LIBDEF PHASE chain on the specified node.

SNODE = secondary-node

SNODE = TCPNAME = tcpvalue

is a 1-16 character alphanumeric name that specifies the secondary node (SNODE) used in the Process. The name can be alphanumeric or national (@ # \$) with embedded periods. This parameter is required for the PROCESS statement, unless it is specified in the SUBMIT statement.

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

When used in the SUBMIT statement, this parameter overrides the value specified in the PROCESS statement. The default value for SNODE is the value specified in the PROCESS statement.

Use **TCPNAME=tcpvalue** to specify TCP/IP connections that are not defined in the IBM Connect:Direct Network Map. **tcpvalue** specifies 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.

When you use the TCPNAME keyword, the default TCP/IP port number is assumed.

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

specifies security user IDs and passwords at the SNODE.

For Connect:Direct for i5/OS, if the Process submitter's SNODEID and password is not specified in the PROCESS statement, the Process submitter's user ID and password is used for the Connect:Direct for i5/OS security ID and password check.

id specifies the 1-8 character security ID passed to the security system on the SNODE.

For Connect:Direct for HP NonStop, this subparameter specifies the HP NonStop group number and user number. These numbers can range from 0-255. Use a period as a separator between the group number and the user number.

For Connect:Direct for i5/OS, this subparameter specifies the i5/OS user profile used for authorization checks during Process execution. It is limited to 8 characters even though i5/OS user profiles may be 10 characters long.

pswd specifies the current 1-8 character security password and can be used by the SNODE security system to validate the security password. This is optional unless the user's security requires a password.

For Connect:Direct for HP NonStop, the VM node only recognizes passwords specified in uppercase alphanumeric characters. A Process cannot be successfully initiated from Connect:Direct for VM/ESA with Connect:Direct for HP NonStop unless the Connect:Direct for HP NonStop SNODEID password only contains uppercase alphanumeric characters (no control characters).

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

For Connect:Direct for HP NonStop, SAFEGUARD must be running on HP NonStop.

For Connect:Direct for i5/OS, this subparameter is ignored.

**SPACE=[(start-track | start-block | , (allocation))]
[(trigger-key, (allocation))]
[(record-size, (primary-allocation, secondary-allocation))]**

specifies the amount of DASD storage allocated for new files on the destination node. You must specify SPACE 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 different formats for noncontrolled BSAM files, START TRACK-1 controlled BSAM data sets, and 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 use a disk management system such as CA-DYNAM/D or CA-EPIC and you specify an allocation trigger value, specify the UNIT=DNOASGN in your Process.

When these parameters are specified, CA-DYNAM/D or CA-EPIC perform both primary and optional secondary data set extent allocation for IBM 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 for 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.

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.

Do not specify STARTT with RETAIN=INITIAL.

If you specify both HOLD=YES and a STARTT value in a Process, the HOLD specification takes precedence, and the Process is placed in the Hold queue.

date specifies that the Process starts on a specific date. Depending on the value of the DATEFORM initialization parameter, you can specify the date in one of the following formats:

DATEFORM Value	Formats			
DATEFORM=MDY (default)	mm/dd/yy	mm/dd/yyyy	mm.dd.yy	mm.dd.yyyy
DATEFORM=DMY	dd/mm/yy	dd/mm/yyyy	dd.mm.yy	dd.mm.yyyy
DATEFORM=YMD	yy/mm/dd	yyyy/mm/dd	yy.mm.dd	yyyy.mm.dd
DATEFORM=YDM	yy/dd/mm	yyyy/dd/mm	yy.dd.mm	yyyy.dd.mm

You can use periods or slashes (/) to separate the components of a date value. You can omit the period or slash separators for transfers between mainframe nodes.

If you only specify a date, the time defaults to 00:00.

If you specify RETAIN=YES, you cannot specify a date in the STARTT parameter.

Valid Julian date formats are yyddd, yyyyddd, yy/ddd, yyyy/ddd, yy.ddd, or yyyy.ddd.

day specifies the day of the week to release the Process for execution. Valid names include MONday, TUEsday, WEDnesday, THURsday, FRIday, SATurday, and SUNday. You can abbreviate the day value 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.

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.

You can specify the time of day using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00:00 to 24:00:00. If you do not use AM and PM, the 24-hour clock is assumed.

If you use the 12-hour clock, 01:00:00 hours can be expressed as 1:00AM, and 13:00 hours can be expressed as 1:00PM.

If you specify hh:mm:ssXM with RETAIN=YES, the Process executes at the same time every day. You do not need to specify minutes and seconds.

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

stepname

specifies the user-defined name of the Copy, Run Job, Run Task, or Submit step.

Stepnames must begin in column one. Stepnames are 1-8 character alphanumeric strings. The first character must be alphabetic.

SUBMIT

identifies the SUBMIT statement.

SUBNODE = PNODE | SNODE

specifies the node where the Process defined in a SUBMIT statement will execute. PNODE indicates that the Process is submitted on the node with Process control. SNODE indicates 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 submitted. The default is PNODE.

SUBNODE=SNODE is not valid if communicating with a Connect:Direct for i5/OS node.

SYMBOL

identifies the SYMBOL statement.

&symbolic_name = variable string

specifies a string that is substituted into the Process. This parameter is required. The symbolic name can be 1-17 alphanumeric characters.

When IBM Connect:Direct encounters a &symbolic name, IBM Connect:Direct substitutes the specified string.

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 IBM Connect:Direct statement string can be concatenated, allowing you to move data in many ways.

Specify a null value by =, (an equal sign immediately followed by a comma). Enclose symbolic parameters strings containing special characters in single quotation marks.

&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 value in the SUBMIT command.

Enclose a symbolic parameter string containing special characters in single quotation marks. Specify a null value by the equal sign (=) immediately followed by a comma.

The symbolic parameter for the SUBMIT statement must begin with a single ampersand. This allows correct symbolic parameters resolution.

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.

SYSOPTS = "DBCS=(tablename,so,si,PAD | PAD=pc)" "parameter1[,parameter2...]"

specifies system operation parameters.

DBCS=(tablename,so,si,PAD | PAD=pc) invokes the double-byte character set (DBCS) translation facility. File transfer with double-byte character set is not supported in block mode.

Note: Block mode transfers can produce unpredictable results in the destination file. These results can compromise data integrity. You will not receive an error message in these situations.

- **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.

IBM Connect:Direct provides the following tables:

- **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'.

NOSO indicates no shift-out or shift-in character and is specified 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 for VSE node.

TO

specifies the destination file characteristics. This parameter is required.

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

specifies the status of the file on the destination node. Subparameters are:

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

- **NEW** specifies that the Process step will create the destination file. This 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.

(TO) DSN =

specifies the destination data set name. This parameter is required. (DSN is optional when used with the IOEXIT parameter.)

If the data set name does not follow VSE naming conventions, enclose the data set name in single quotation marks.

If the data set being copied from requires a password for read or the data set being copied to requires a password for write, specify the password in the COPY statement in the format **datasetname/password** (separate the data set name and password with a slash). For example:

```
COPY TO DSN=datasetname/password...
```

The password is used at data set allocation. If the password is not correct, VSE issues a WTOR requesting the password when Connect:Direct for VSE software opens the data set.

THEN

specifies the subsequent processing to perform if the specified condition is met.

TYPE = typekey

specifies the entry name of the type defaults file. This file contains the default file attributes 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, you must enter the typekey in the SYSOPTS parameter on the TO clause of the Connect:Direct for OpenVMS COPY statement.

For OpenVMS to VSE copies, the typekey cannot exceed eight characters.

UNIT=([group-name | device-type | unit-address | DLBONLY | 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 specified 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, specify 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.

See the following table to select the supported IBM Connect:Direct group-name for the input or output device on your Processes. For device independence and greatest flexibility, specify UNIT=group-name in your Processes.

UNIT=device-type specifies the type of device for your Processes.

See 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, specify 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, specify UNIT=CUU in your Process and replace the CUU with the device Channel/Unit address.

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
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=DLBONLY 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 IBM 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 IBM 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 IBM 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 IBM Connect:Direct.

**VOL=([PRIVATE],[RETAIN],[volume-count],
[SER=(serial-number [, serial-number.,,,])] | SER=(serial-number, [serial-number])**

specifies the volume serial numbers 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.

Connect:Direct for VSE does not use this parameter but it is supported on other IBM Connect:Direct platforms and is listed here for compatibility with those platforms.

RETAIN is ignored by IBM Connect:Direct, because IBM Connect:Direct dynamically deallocates data sets. However, if you omit RETAIN, you must specify a comma in its place.

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.

Do not specify the VOL=SER parameter when you allocate an output scratch tape data set for a VSE node. IBM 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 where the VSAM file resides.

dsn is the name of the catalog where the file resides. This subparameter is required.

mode specifies the catalog's 2-character VSE file mode. This subparameter is included for compatibility with Connect:Direct for VM.

userid specifies the 8-character VSE user ID that owns the catalog. This subparameter is included for compatibility with Connect:Direct for VM.

pswd specifies the password of the VSE user that owns the catalog. This subparameter is included for compatibility with Connect:Direct for 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 for VSE.

Specify VSAMCAT=(dsn,1,1,,111) as a standard.

z/OS Statements and Parameters

Connect:Direct for z/OS Process Statement

The PROCESS statement defines the attributes of a Process and is always the first statement in a Process.

The maximum storage area allowed for a Process statement is 1MB. 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.

To ensure that the Transmission Control Queue has adequate space to store a large Process, see *Planning the Installation* in *IBM Sterling Connect:Direct for z/OS Configuration Guide*.

Note: Passwords for PNODEID and SNODEID are limited to 8 characters in either of the following scenarios: if you submit a process between the PNODE and SNODE over a SNA or SNUF connection, or if the process is a PNODE=SNODE PROCESS.

The following is the Connect:Direct for z/OS Process statement format. Refer to [Process Parameters](#) for more information.

Label	Statement	Parameters
process name	PROcEss	SNODE = secondary-node-name SNODE = TCPNAME = tcpvalue; port SNODE = UDT33NAM = udtvalue;port
		CRC=(OFF ON)
		CLASS = n
		DEBUG = trace bits
		HOLD = Y YES N NO CALL
		MAXDELAY = [UNLIMITED QUEUED hh:mm:ss 0]
		NOTIFY = %USER userid
		PACCT = `pnode-accounting-data'
		PLEXCLASS = (pnode class, snode class)
		PNODE = primary-node-name %PNODE
		PNODEID = (id [,pswd] [,newpswd])
		PRTY = nn

Label	Statement	Parameters
		REQUEUE = Y YES N NO
		RETAIN = Y YES N NO INITIAL
		SECURE=OFF SSL TLS10 TLS11 TLS12 or SECURE=ENCRYPT.DATA=Y N or SECURE = (OFF SSL TLS10 TLS11 TLS12 ENCRYPT.DATA=Y N) or SECURE = (OFF SSL TLS10 TLS11 TLS12 <cipher_suite> (cipher_suite_list), ENCRYPT.DATA=Y N) Note: If System SSL is in FIPS mode, only certain ciphers are valid. See the <i>IBM Sterling Connect:Direct for z/OS Release Notes</i> for a list of valid FIPS-mode ciphers.
		SACCT = `snode-accounting-data'
		SNODEID = (id [,pswd] [,newpswd])
		STARTT = ([date day] [,hh:mm:ssXM])
		&symbolicName1 = variable-string-1 &symbolicName2 = variable-string-2 . . . &symbolicNamen = variable-string-n

Connect:Direct for z/OS Copy Statement

The Connect:Direct for z/OS COPY statement copies the following file types from one IBM Connect:Direct node to another:

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

The Connect:Direct for z/OS COPY statement supports both disk and tape transfers.

The COPY statement contains a FROM parameter that specifies the source file name and a TO parameter that specifies the destination file name. You can specify additional parameters to customize the file transfer.

To copy from one IBM Connect:Direct platform to another, use the COPY FROM and COPY TO statements for the applicable platforms. For example, if the source file is on a Connect:Direct for z/OS node, use the

z/OS COPY FROM parameters. If the file destination is a Connect:Direct for HP NonStop node, use the Connect:Direct for HP NonStop COPY TO parameters.

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

The COPY statement supports Multibyte Character Set (MBCS) conversion.

See the following links for information on supported files types:

- [“PDS Support” on page 188](#)
- [“SMS Support” on page 189](#)
- [“VSAM Support” on page 189](#)
- [“GDG Support” on page 191](#)
- [“MBCS Support” on page 191](#)

The following is the Connect:Direct for z/OS COPY statement format. Refer to [Connect:Direct for z/OS Process Parameters](#) for more information.

Label	Statement	Parameters	
stepname	COPY	From	(
			DSN=data set name/password FILE=filename
			FILE=businessprocess mailbox
			PNODE SNODE
			DCB=([model-file-name] [,BLKSIZE=no.-bytes] [,NCP] [,DEN=0 1 2 3 4] [,DSORG= PS PO DA VSAM] [,KEYLEN=no.-bytes] [,LIMCT=no.-blocks-or-tracks] [,LRECL=no.-bytes] [,OPTCD=W Q Z] [,RECFM=record-format] [,RKP=first-byte-position-of-record-key] [,TRTCH=C E T ET COMP NOCOMP])
			DISP=([OLD SHR], [KEEP DELETE], [KEEP DELETE])
			RESGDG = S SUB R 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])

Label	Statement	Parameters
		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
		SELECT=(member generic (*) (member, [new-name],[NR R]) (generic,, [NR R]) (start-range/stop-range,,[NR R]) list)
		BUFND=number
		IOEXIT=exit-name (exit-name [,parameter,...])
		DATAEXIT=exit-name (exit-name [,parameter,...])
		SYSOPTS="DBCS=(tablename,so,si,PAD PAD=pc, LOGIC=A B (B,RC))" "CODEPAGE=(from code set,to Unicode code set)" "parameter1[,parameter2,...]" "DATATYPE=TEXT BINARY" "XLATE=N NO Y YES" "STRIP.BLANKS=N NO Y YES" "PERMISS=nnn" "PRECOMP=Y YES N NO " SYSOPTS="ZIIP=ON ZIIP=OFF and ZEDC=ON ZEDC=OFF"
)
		TO (
		DSN=data set name/password FILE=filename
		FILE=businessprocess mailbox
		PNODE SNODE
		TYPE=typekey

Label	Statement	Parameters
		DCB=([model-file-name] [,BLKSIZE=no.-bytes] [,NCP] [,DEN=0 1 2 3 4] [,DSORG=PS PO DA VSAM] [,KEYLEN=no.-bytes] [,LIMCT=no.-blocks-or-tracks] [,LRECL=no.-bytes] [,OPTCD=W Q Z] [,RECFM=record-format] [,RKP=first-byte-position-of-record-key] [,TRTCH=C E T ET COMP NOCOMP])
		DISP=([NEW OLD MOD RPL SHR] , [KEEP CATLG] , [KEEP CATLG DELETE])
		AVGREC=U K M
		DATACLAS=data-class-name
		DSNTYPE=PDS LIBRARY EXTPREF EXTREQ BASIC LARGE
		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 XPDT=[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])

Label	Statement	Parameters
		BUFND=number
		IOEXIT=exit-name (exit-name [,parameter,...])
		DATAEXIT=exit-name (exit-name [,parameter,...])
		SYSOUT=(sysout_parameter1, sysout_parameter2, . . .)
		SYSOPTS="UNIQUE=YES" "DBCS=(tablename,so,si,PAD PAD=pc), LOGIC=A B (B,RC))" "CODEPAGE=(from Unicode code set,to code set)" "DATATYPE=TEXT BINARY" "XLATE=N NO Y YES" "STRIP.BLANKS=N NO Y YES" "PERMISS=nnn" "SYSOUT=(sysout_keyword1, sysout_keyword2, . . .)" SYSOPTS="ZIIP=ON ZIIP=OFF and ZEDC=ON ZEDC=OFF")
		CKPT=nK nM ZFBA=2
		COMPRESS [[PRIMEchar=X'40' X'xx' C'c'] EXTended EXTended=(TYPE=(ZEDC NOZEDC , ZIIP NOZIIP) Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

PDS Support

Connect:Direct for z/OS supports transmission of all PDS files, including load modules and overlay files.

You can specify whether:

- An entire PDS is sent.
- Specific members are selected for transmission (SELECT parameter).
- Specific 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).

The following are the guidelines for copying a PDS:

- EXCLUDE or SELECT cannot be used if the FROM DSN parameter 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 file is specified in the FROM DSN, the TO DSN must specify a single member.
- When the COPY statement involves two PDS files, 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. Also, 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.

VSAM Support

Copies to existing VSAM files are processed in EXTEND mode. The VSAM file is extended rather than replaced. To completely copy over an existing VSAM file (to copy to a VSAM file in LOAD mode) set the DISP parameter to DISP=RPL. You must define the destination file with the REUSE attribute; otherwise OPEN will fail.

Copies to new VSAM files propagate those attributes supported by SMS. Specifically, attributes supported by SVC 99 for dynamic allocation are propagated. See “SMS Support” on page 189 for the specific attributes that are propagated. If the file requires other VSAM attributes, you must pre-allocate the file outside of IBM Connect:Direct using IDCAMS, or in a Process step using the DMRTAMS utility prior to the COPY step. See the *IBM Sterling Connect:Direct for z/OS User Guide* for information about DMRTAMS.

SMS Support

The Connect:Direct for z/OS COPY statement supports transmitting and creating 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, DCB or SPACE parameters are not 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 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 data class is propagated):

- KEYLEN - For KSDS data sets only. The length of the key.
- KEYOFF - For KSDS data sets 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.

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

- For platforms other than z/OS, 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 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 SMS class name 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))

```

When a class contains the class name value of \$\$\$\$\$\$\$, IBM Connect:Direct performs an INFO call to dynamic allocation to obtain the SMS classes associated with the sending data set. If IBM Connect:Direct returns an SMS class, this class is substituted for the \$\$\$\$\$\$\$ value. If IBM Connect:Direct does not

return an SMS class, the keyword and value are not passed to the receiving node. The propagated class name must be defined on the receiving node (if the receiving node is a z/OS) or an error occurs.

GDG Support

Connect:Direct supports sending and receiving Generation Data Set (GDS) files to a Generation Data Group (GDG) with the following stipulations:

- Only one file (GDS) of a GDG can be referenced in a COPY step.
- Because enqueueing is needed to ensure data integrity during transfer to the receiving node, Connect:Direct does not support allocating and reading an entire GDG through a single DD statement (GDG ALL).
- Specify CATLG as the second and third subparameters of the DISP parameter to ensure that the Connect:Direct recovery routines use the correct GDS when RETRY and CHECKPOINT are used. For example, DISP=(NEW,CATLG,CATLG).

For more information on GDG files, see the IBM z/OS reference manuals. The *z/OS MVS JCL User's Guide (SA22-7598)* contains a good explanation of GDGs, allocation, and use and includes references to additional resources.

MBCS Support

Connect:Direct for z/OS Multibyte Character Set (MBCS) support enables you to convert between Unicode and other code sets supported on the z/OS platform. To perform an MBCS conversion, use the CODEPAGE parameter of the COPY statement FROM and/or TO SYSOPTS clauses .

You can perform MBCS conversions in the following ways:

- Perform a conversion on the FROM node only and then send the Unicode file to the TO node.
- Send a file to the TO node and let that node perform the conversion.
- Perform a conversion from one z/OS compatible code set to a Unicode code set supported on the local node (specified in the FROM clause CODEPAGE parameter). Then send the encoded Unicode file to the remote node to be converted to another z/OS compatible code set.

Instead of requiring that each IBM Connect:Direct node provide the capability to convert from any supported character set to any other supported character set, the recommended approach is to convert the original character set to a common intermediate form (UTF-8 or UCS-2) on the local node, transmit the intermediate form to the remote node, and then perform the conversion to the final desired character set on the remote node. This way, each node is responsible only for conversion between the Unicode encoding and the character sets relevant to and supported by the node.

To display the CODEPAGE specification for a COPY step in a Process after step completion, use the Select Statistics command for an SY Statistics record. Each node involved in a COPY generates an SY record containing the SYSOPTS relevant to that node.

Except for syntax, the CODEPAGE parameter is not validated when the Process is submitted. However, when the Process is executed, an MBCS001E error will result on the node attempting the conversion if an invalid code set is specified.

The following are examples of COPY performing MBCS conversion:

- MBCS Conversion During z/OS to UNIX Copy
- MBCS Conversion During Microsoft Windows to z/OS Copy
- MBCS Conversion During z/OS to z/OS Copy

Note: To convert between Unicode (ISO 10646) and other code sets, IBM Connect:Direct makes calls to system routines which are part of the optional z/OS Language Environment component – National Language Support. Verify that your z/OS installation supports the code set conversions specified in the Process language.

Connect:Direct for z/OS Run Job Statement

The RUN JOB statement submits a job through the z/OS internal reader, a facility that transfers jobs to the job entry subsystem (JES). The job must reside in a file on the node that executes the RUN JOB statement.

IBM Connect:Direct does not verify job statements. To determine the completion status of a RUN JOB statement, check your IBM Connect:Direct statistics records.

The following is the Connect:Direct for z/OS RUN Job statement format. Refer to [Connect:Direct for z/OS Process Parameters](#) for more information.

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

Connect:Direct for z/OS Run Task Statement

The RUN TASK statement attaches user programs or subtasks during Process execution. When a IBM Connect:Direct Process issues a RUN TASK statement, the IBM Connect:Direct Process waits until the subtask finishes before executing the next IBM Connect:Direct Process step.

You can pass a list of user parameters to the subtask from the RUN TASK statement. 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.

Parameters are passed in upper case using the PARM parameter. Parameters are passed in mixed case using the SYSOPTS parameter. (A double-quoted SYSOPTS string can contain up to 1816 bytes.)

The subtask can be attached at either the PNODE or SNODE. The subtask must reside in a load library that can be accessed by the target (SNODE or PNODE) IBM Connect:Direct server.

Note: Run Task cannot execute C or C++ programs that contain a "main()" routine or enclave. Using Run Task to execute such a program results in a U4093 ABEND and creates error CEE5151C in SYSOUT. The return code 156 is generated with a reason code of 0D070201.

The following is the Connect:Direct for z/OS RUN TASK statement format. Refer to [Connect:Direct for z/OS Process Parameters](#) for more information.

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

Connect:Direct for z/OS Submit Statement

The SUBMIT statement submits another Process to either the PNODE or SNODE, from within an executing Process. 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.

The SUBMIT statement is not the same as the SUBMIT command. The SUBMIT statement parses special characters differently from the SUBMIT command. See *Process Language Syntax* for a discussion of special character parsing. See the *IBM Sterling Connect:Direct for z/OS User Guide* for SUBMIT command syntax and parameters.

The following is the Connect:Direct for z/OS SUBMIT statement format. Refer to [Connect:Direct for z/OS Process Parameters](#) for more information.

Label	Statement	Parameters
stepname	SUBMIT	DSN=dsn[(member)]
		<i>business process name</i>
		DEBUG=trace bits
		NEWNAME=new-name
		SUBNODE=PNODE SNODE
		SNODE=secondary-node-name SNODE = TCPNAME = tcpvalue, port
		SNODEID=(id [,pswd] [,newpswd])
		SACCT='snode-accounting-data'
		PLEXCLASS=(pnode class, snode class)
		PNODEID=(id [,pswd] [,newpswd])
		PACCT='pnode-accounting-data'
		CASE= Y YES N NO
		CLASS = n
		HOLD = Y YES N NO CALL
		PRTY = nn
		NOTIFY = %USER userid
		REQUEUE = Y YES N NO
		RETAIN = Y YES N NO INITIAL
		STARTT = ([date day] [,hh:mm:ssXM])
		&symbolicName1 = variable-string-1 &symbolicName2 = variable-string-2 . . . &symbolicNamen = variable-string-n

Connect:Direct for z/OS Conditional Statements

Conditional statements alter the sequence of IBM Connect:Direct Process execution based on the completion of the previous step in the Process. For example, if a file copy fails, the Process may call an external program to generate a console message and stop the Process. If the file copy succeeds, the Process continues with the next step.

The following is the Connect:Direct for z/OS Conditional statement format. Refer to [Connect:Direct for z/OS Process Parameters](#) for more information.

Label	Statement	Parameters
optional	IF	(label condition nn) THEN

Label	Statement	Parameters
		(process steps)
	ELSE	
		(alternative process steps)
	EIF	
optional	GOTO	label
	EXIT	

Connect:Direct for z/OS Symbol Statement

The SYMBOL statement creates symbolic substitution values, and can contain up to 254 bytes.

The following is the Connect:Direct for z/OS Symbol statement format. Refer to [Connect:Direct for z/OS Process Parameters](#) for more information.

Label	Statement	Parameters
optional	SYMBOL	&symbolic_name=variable-string

Creating Processes from z/OS Statement Models

Connect:Direct for z/OS has a sample Process library that contains Process statement models. You can use these models as templates for building IBM Connect:Direct Processes.

There are two types of the Process statement models in the Process library:

- Commented files have a file name preceded by the at sign (@).
- Files without comments have a file name preceded by a pound sign (#).

To create a Process from a Process model,

1. Use either the Connect:Direct for z/OS Interactive User Interface (IUI) or the Interactive System Productivity Facility (ISPF) EDIT facility.
2. Access the Process definition screen (DF on the Connect:Direct for z/OS Primary Option Menu), and specify a new PDS member.
3. Press **ENTER**.
4. Type COPY on the command line at the top of the blank member.
5. Press **ENTER**. An ISPF Edit-Copy screen is displayed.
6. 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, and press ENTER.

For example, you build your Process by first including the PROCESS statement. If you specify @PROCESS with the Process library, the commented IBM 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.

7. To add additional statement models, type a over the 0 (zero) in the number column of the last line of the member and repeat steps 4 through 6.

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

8. Continue adding statement models until your Process is complete.
9. Edit the statements:
 - Replace underscores with the appropriate parameter values.

- Provide an appropriate Process name in the PROCESS statement.
 - Delete any lines that are not applicable.
 - Continuation marks are necessary on all but the last line of each statement model.
 - Delete any unwanted Comment lines.
10. Save the Process to the PDS member. You can now submit it from the Submit panels in the IUI. See the *IBM Sterling Connect:Direct for z/OS User Guide* for instructions on executing Processes.

Connect:Direct for z/OS Process 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, IBM 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 z/OS: 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.

CASE = Y | YES | N | NO

specifies whether parameters associated with accounting data, user ID, password, and data set name in the command and in the submitted 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 IBM 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.

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.

Valid values are:

- 1-2147483K
- 1-2147M

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

Connect:Direct for i5/OS 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, and PDSs can be checkpointed. For PDS-to-PDS transmission, any value specified causes Connect:Direct for z/OS to checkpoint each member. Note that sequential-to-PDS and PDS-to-sequential transmissions cannot be checkpointed.

Note: For sequential files, do not specify a CKPT value less than:

```
BLKSIZE * NCP * 10 * # Stripes
```

where **NCP** is the number of buffers for reading data from or writing data to a sequential data set using BSAM and **# stripes** refers to striped extended-format data sets (see the **NCP** parameter for more information).

If the specified checkpoint interval is too small, it can significantly reduce transmission speed in the following ways:

- The amount of data in an RU or packet may be reduced, thus increasing the number needed and with that the number of network I/O operations. In V2 (TCP and LU6.2) transmissions, the packet or RU count increases one for one with the number of checkpoints taken.
- Sequential file I/O slows down because the average number of overlapping I/Os outstanding is reduced when checkpointing occurs too frequently.

COMPRESS [[PRIMEchar = X'40' | X'xx' | C'c'] | EXTended] | EXTended=(TYPE=(ZEDC | NOZEDC , ZIIP | NOZIIP)

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.

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, IBM 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.

Note: If the SNODE is B2B Integrator, only COMRESS EXT is supported.

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

EXTended converts repetitive strings in the data into codes that are transmitted to the remote node. These codes are converted back to the original data string by the remote node during decompression. Specify this parameter when line transmission bandwidth is limited and CPU cycles are available.



CAUTION: Compression consumes significant CPU resources. To avoid performance degradation in your production environment when changing extended compression parameter settings, review *Testing the Effects of Changing Values for Extended Compression Parameters* in the *IBM Sterling Connect:Direct for z/OS User Guide* for information on the CDSACOMP offline utility. This utility can help determine if changing the extended compression parameters' default values at the global level or overriding them at the Process level will significantly improve your system performance.

The following are options for EXTended:

- **CMPlevel** determines the compression level. The valid value range is 1-9. Level 1 is the fastest compression and usually provides sufficient 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 window size specifications increase the degree of compression and use more virtual memory. Size 8 uses 1 KB of memory. Size 15 requires 128 KB of memory. The default is 13.
- **MEMlevel** identifies how much virtual memory (above the line) is allocated to maintain the internal compression state. The valid value range is 1-9. Level 1 requires the least memory (1 KB). Level 9 requires the most memory (256 KB). The default is 4.

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

```
COMPRESS EXT = ( CMP=1  
                 WIN=12  
                 MEM=8 )
```

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.

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 specified first.

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.

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 z/OS: 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.

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 *IBM Sterling Connect:Direct for z/OS 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.

DCB =([model-file-name]
[,BLKSIZE = no.-bytes]
[,NCP] = no. of buffers for BSAM data transfers
[,DEN = 0 | 1 | 2 | 3 | 4]
[,DSORG = PS | PO | DA | VSAM]
[,KEYLEN = no.-bytes]
[,LIMCT = no.-blocks-or-tracks]
[,LRECL = no.-bytes]
[,OPTCD = W | Q | Z]
[,RECFM = record-format]
[,RKP = first-byte-position-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 maximum length depends on the device type. For most device types, the maximum length is 32,760 bytes, although Connect:Direct for z/OS supports a maximum length of 262,144 bytes for certain tape drives.

If you specify a block size other than zero, there is no minimum requirement for block size except for variable format data sets that have a minimum block size of 8. However, if a data check occurs on a magnetic tape device, any block shorter than 12 bytes in a read operation, or 18 bytes in a write operation, is treated as a noise record and lost. No check for noise is made unless a data check occurs.

BLKSIZE=0 allows the operating system to derive the block size. The system does not derive a block size for old or unmovable data sets, or when the RECFM is U. If the BLKSIZE remains 0 when the data set is opened, the Process fails with an SVSG005I message.

NCP (number of channel programs) specifies the number of buffers for reading data from or writing data to a sequential data set using BSAM. The default is 0, which lets the system determine the value and usually produces the best throughput results. The maximum is 255. When the number of BSAM buffers is greater than 1, IBM Connect:Direct interleaves BSAM and network I/O to maximize throughput. For more information on BSAM data processing, see *Performance Tuning* in the *IBM Sterling Connect:Direct for z/OS Administration Guide*.

Note: Here are some additional points you may need to know for your environment:

The NCP value may be limited by the MAXSTGIO initialization parameter. For more information on the MAXSTGIO initialization parameter, see *Chapter 16. Global Initialization Parameters* in the *IBM Sterling Connect:Direct for z/OS Administration Guide*.

When LU0 is used and checkpointing is requested, IBM Connect:Direct sets NCP to 1 (see the CKPT parameter for more information.)

If the TAPEIO initialization parameter is set to EXCP, the NCP value is ignored and only one buffer is acquired with the storage occurring below the line.

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 specified together, the DEN and TRTCH values are used to select a tape device for allocation by IBM 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 PO, PS, DA, 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=VB, the BLKSIZE value does not need to be an even multiple of LRECL.

An entry-sequenced file coming from HP NonStop to an IBM PS/VB file must be specified with an LRECL 4 bytes larger than the HP NonStop 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 HP NonStop.

When you are performing an MBCS conversion between two z/OS nodes, the receiving file LRECL must be larger than the sending file LRECL, due to possible data length increase during conversion and to avoid an SVSJO32I error during the Copy. Making the receiving file LRECL 10% larger than that of the sending file is usually adequate.

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.

An OpenVMS file with a record format of undefined (U) cannot be copied to z/OS.

When performing an MBCS conversion on a file created on a z/OS receiving node, you must specify the file record format (RECFM) as V (Variable), VB (Variable Block), or U (Undefined). If RECFM=VB, BLKSIZE for the output file must be at least as large as LRECL +4.

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 specified together, the TRTCH and DEN values are used to select a tape device for allocation by IBM 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.

DEBUG = trace bits

specifies the 8-position trace setting for the Process. This allows you to specify a trace for a specific Process. The following table shows the available function traces for Connect:Direct for z/OS, with their respective DEBU settings, and the DD names (or filenames) used for output. Specify these bits using hexadecimal notation. For example, X'80' plus X'10' results in X'90 and X'08' plus X'04' results in X'0E'.

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
00400000	TCQSH from DMCOPYRT	DMCOPYRT
00200000	Make each SVC dump unique	N/A
00100000	SECURITY Trace Control	SECURITY
00040000	GETMAIN/FREEMAIN trace	RADBDD16
00008000	I/O buffer trace	RADBDD21
00004000	WTO all dynamic allocation parameters	RADBDD22
00002000	IBM 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
00001000	Workload Balancing trace	CDPLXWLB
00000100	In-storage tracing only. WARNING: If specified, all SYSOUT (except WTOs) goes to in-storage only.	N/A
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

DSN = dsn[(member)]

For a Submit statement, this 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.

For a Run Job statement, this 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, specify the value for DSN in uppercase letters when submitting a Process from UNIX that runs a job on z/OS.

For a Copy statement using Version 5.0 or later, DSN=NULLFILE may be used. NULLFILE is a dummy data set.

DSNTYPE = LIBRARY | PDS | BASIC | LARGE | EXTPREF | EXTREQ

defines a specific data set organization for a sequential (BASIC or LARGE) or partitioned (LIBRARY or PDS) data set. The BASIC and LARGE parameters are only available for Connect:Direct for z/OS.

For platforms other than z/OS: The DSNTYPE parameter must be specified as a subparameter 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 a partitioned data set extended (PDSE).
- **PDS** specifies a partitioned data set.
- **BASIC** specifies a sequential data set that cannot contain more than 65535 tracks per volume.
- **LARGE** specifies a sequential data set which can contain more than 65535 tracks per volume.
- **EXTPREF** specifies a data set where the extended attribute is preferred but not required. The data set is created whether the extended attribute is obtained through this parameter or another such as DATACLAS or LIKE.
- **EXTREQ** specifies a data set which requires the extended attribute. If an extended format data set cannot be allocated, a data set is not created.

EIF

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

ELSE

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

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.

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.

EXIT

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

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 z/OS data set name conventions. If the data set name does not follow z/OS naming conventions, enclose the data set name in single quotation marks.

Use the relative generation number when copying a Generation Data Group (GDG) data set. Using the relative generation number ensures that no data is lost because you can specify only one Generation Data Set (GDS) in a COPY step.

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, z/OS issues a WTOR requesting the password when IBM Connect:Direct opens the data set. For example:

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

If the data set 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
```

(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.

GOTO

moves to a specific step within a Process.

HOLD = Y | YES | N | NO | CALL

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

Y or **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 (a Process submitted with HOLD=YES is placed in the Hold queue even if a start time is specified).

Specifying both HOLD=YES and MAXDELAY=YES is not valid. The PROCESS statement returns message SCBI219I.

N or **NO** specifies that the Process is to execute as soon as possible. HOLD=NO is the default.

CALL specifies that IBM 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:

- IBM Connect:Direct ignores the HOLD parameter if RETAIN=Y.
- When the SNODE is a Microsoft Windows operating system, a null or ENABLE Process is required from the Microsoft 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.

IF THEN

specifies that IBM 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.

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 *IBM Sterling Connect:Direct for z/OS 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.

label

The label is a user-assigned 1-8 character alphanumeric string that identifies the statement.

Labels must begin in column one. The first character must be alphabetic.

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.

**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 Microsoft 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 z/OS: 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 RECOG=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 z/OS: 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.

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 z/OS: 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 z/OS: 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.

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

indicates that the submit command waits until the submitted Process completes execution or the specified time interval expires. This parameter is optional. Do not use MAXDELAY for a submit within a Process—use only in SUBMIT commands.



CAUTION: If you use the MAXDELAY PROCESS statement keyword, you must specify ESF=NO or the message *SCBI221I MAXDELAY not supported for ESF submits* will display when the DTF is down and the Process will not submit.

Note: If the Process does not complete within the time interval specified by queued or hh:mm:ss, the API returns SSPA006I, RC=4 and DMBATCH terminates with RC=48 (x'30').

UNLIMITED specifies that the submit command waits until the Process completes execution. This is the default when no parameters are specified.

QUEUED specifies that the submit command waits until the Process completes or 30 minutes, whichever occurs first.

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.

0 specifies that the submit command attempts to start a session for the submitted Process to execute on immediately. If IBM Connect:Direct cannot establish a session, after all retries are exhausted, the Process is flushed and the submit command fails with the error SVTM118I RC=52(x'34').

If IBM Connect:Direct cannot establish a session after all retries are exhausted due to all available sessions on the remote node being in use, that is, when session attempts fail with error SVTM080I SESSION (nnn) REJECTED pname (pnum) SNODE=remote.node, the Process is flushed and the submit command fails with the error SVTM118I RC=12(x'0C').

MAXDELAY=0 Processes will not use the intelligent retry feature. When a transfer to a remote node times out and retries, subsequent transfers to the same remote node will also time out and retry rather than being added to the wait queue.

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 z/OS: 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.

NEWNAME = new-name

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

nn

specifies the numeric value to be used for comparison. If specified 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.

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.

NOTIFY = %USER | userid

specifies that the user receives Process completion messages.

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

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

NOTIFY is not supported across z/OS images in a sysplex environment.

PACCT = 'pnode-accounting-data'

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

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 z/OS 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 a z/OS to run on a HP NonStop 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.

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.

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.

PLEXCLASS = (pnode class, snode class)

specifies the class that directs the Process to only certain servers in a IBM Connect:Direct/Plex. This parameter is only used in a IBM Connect:Direct/Plex.

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

PNODE

For a Copy statement, this specifies the primary node, defining the direction of transfer (with SNODE). When PNODE is specified with the FROM parameter, a send takes place. When PNODE is specified with the TO parameter, a receive takes place. PNODE is the default for the FROM parameter.

For a Run Job statement, this 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.

For a Run Task statement, this specifies that the program will be executed on the PNODE, which is the default.

PNODE = primary-node-name | %PNODE

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 alphanumeric 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.

For more information about %PNODE, see the description for the &symbolicName1 Process parameter.

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 IBM 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.

PROcEss

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

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. The PROCESS keyword must be on the same line as the Process name.

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.

PRTY = nn

specifies the Process priority in the Transmission Control Queue (TCQ). The TCQ holds all Processes that have been submitted to IBM 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 for z/OS initialization parameters.

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 z/OS: 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.

REQUEUE = Y | YES | N | NO

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

Y or **YES** places the requeued Process 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. The Process must be explicitly released from the Hold queue when the status is HELD IN ERROR (HE).

N or **NO** enables 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 if REQUEUE=YES is specified.

RESGDG = S | SUB | R | RUN

allows users to specify submit or execution time for resolution input GDGs. The parameter, valid only for input GDGs, is specified on the FROM clause of the COPY statement.

S or **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.

R or **RUN** specifies GDG resolution at Process run or execution time within the DTF.

RESTART = YES | NO

specifies whether or not the subtask is restarted if interrupted.

RETAIN = Y | YES | N | NO | INITIAL

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

Y or **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 specified. Use RETAIN with STARTT to run a Process repeatedly at an interval.

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

RETAIN=YES is ignored when MAXDELAY is specified.

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

INITIAL specifies that the Process executes every time IBM Connect:Direct initializes. Processes submitted with RETAIN=INITIAL do not execute when submitted. STARTT should not be specified with RETAIN=INITIAL.

RETAIN=INITIAL is ignored when MAXDELAY is specified.

RUN JOB

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

RUN TASK

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

SACCT = 'snode-accounting-data'

specifies the accounting data for the SNODE. The maximum length of the accounting data is 256 characters. Enclose the string in single quotation marks if it contains special characters. 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 for i5/OS node.

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.

SECURE=OFF|SSL|TLS10|TLS11|TLS12

or

SECURE = ENCRYPT.DATA=Y | N

or

SECURE = (OFF | SSL | TLS10|TLS11|TLS12 | ENCRYPT.DATA=Y|N)

or

SECURE = (OFF | SSL | TLS10|TLS11|TLS12 | cipher_suite>|(cipher_suite_list),ENCRYPT.DATA=Y|N)
(for use in a PROCESS statement)

turns on security for a specific session by allowing you to select a protocol (SSL or TLS) and one or more ciphers when non-secure sessions are the default or turns off security when secure sessions are the default. In addition, you can specify the type of encryption you want performed. For more information, see the *IBM Sterling Connect:Direct Secure Plus for z/OS Implementation Guide*.

Parameter	Definition
,ENCRYPT.DATA=Y N or ENC=Y N	Depending on the option chosen, encrypts both the control block information contained in Function Management Headers (FMHs) and the files being transferred (ENC=Y) or encrypts only the FMHs (ENC=N). The type of encryption chosen will be performed if: <ul style="list-style-type: none"> • The SNODE's remote node entry in its Connect:Direct Secure Plus file for this node specifies OVERRIDE=Y. • ENCRYPT.DATA must be the last (or only) value specified on the SECURE= parameter. • Both sides of the connection support ENCRYPT.DATA= for SSL/TLS.
OFF	If the remote node in the Secure+ parameters file specifies OVERRIDE=Y, attempts to start the session as a non-secure session. The session will be successfully established if: <ul style="list-style-type: none"> • The SNODE has no remote node entry in its Secure+ parameters file. • The SNODE's remote node entry in its Secure+ parameters file for this node specifies that all Secure+ protocols are disabled. • The SNODE's remote node entry in its Secure+ parameters file for this node specifies OVERRIDE=Y.
SSL TLS	If the remote node in the Secure+ parameters file specifies OVERRIDE=Y, attempts to start the session as a secure session using the specified protocol. The session will be successfully established if: <ul style="list-style-type: none"> • The SNODE's remote node entry in its Secure+ parameters file for this node specifies that the specified protocol is to be used. • The SNODE's remote node entry in its Secure+ parameters file for this node specifies OVERRIDE=Y.
,Cipher Suite	Specifies the one cipher suite for IBM Connect:Direct to use when executing the Process overriding the cipher suite defined in the Secure+ parameters file, for example, SSL_RSA_WITH_3DES_EDE_CBC_SHA. Note: If System SSL is in FIPS mode, only certain ciphers are valid. See the <i>IBM Connect:Direct for z/OS Release Notes</i> for a list of valid FIPS-mode ciphers.

Parameter	Definition
,(Cipher suite list)	<p>Specifies a list of cipher suites for IBM Connect:Direct to use when executing the Process overriding the cipher suite defined in the Secure+ parameters file. The cipher list must be enclosed in parentheses and each cipher suite separated by a comma or space, for example, (TLS_RSA_AES_128_SHA,TLS_RSA_AES_256_SHA, TLS_RSA_WITH_DES_CBC_SHA)</p> <p>Note: If System SSL is in FIPS mode, only certain ciphers are valid. See the <i>IBM Connect:Direct for z/OS Release Notes</i> for a list of valid FIPS-mode ciphers.</p>

SECURE = ENCRYPT.DATA=Y | N

or

SECURE = ENC=Y | N

specifies whether to set data encryption (FMH control block information and file data being copied or FMH information only) using the SSL or TLS protocol. If both sides support ENCRYPT.DATA= for SSL/TLS, the PNODE governs whether the data is encrypted. For more information, see the *IBM Sterling Connect:Direct Secure Plus for z/OS Implementation Guide*.

ENCRYPT.DATA=Y|N or **ENC=Y|N** enables or disables copy file encryption. Default= Sterling Connect:Direct Secure Plus file value.

**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.

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 = secondary-node-name | SNODE = TCPNAME = tcpvalue;port | SNODE = UDT33NAM = udtvalue;port

specifies the secondary node in the Process.

Note: The default SNODE value is the value specified in the Process statement. The Process statement value can be overridden by the SNODE value specified in the SUBMIT command. If SNODE is specified in the Submit command, it is not required in the Process statement.

secondary-node-name is a 1-16 character alphanumeric name that is defined in the network map. The following characters are allowed:

A-Z, 0-9, !, @, #, \$, %, &, {, }, +, -, and ^

Connect:Direct for z/OS does not accept the following characters for the adjacent node:

(,) =, \, ", ', <, >, |, ||

Use **SNODE=TCPNAME=tcpvalue** or **SNODE=UDT33NAM=udtvalue** to specify TCP/IP or UDT connections that are not defined in the IBM Connect:Direct network map. **tcpvalue** or **udtvalue** can be a DNS name up to 255 characters or a 15-character IPv4 or 39-character IPv6 TCP/IP address. A TCP/IP default entry is required in the network map if a Process uses SNODE=TCPNAME= but is not required with SNODE=UDT33NAM=.

You can specify a port number by appending a semicolon followed by a 1-5 character port number. If you do not specify a port number, IBM Connect:Direct uses the port number from the TCP.IP.DEFAULT entry of the network map. If you do not specify a port number with the udtvalue, the default port value of 1366 is used.

If the DNS cannot fit on one line, concatenate it with the double-pipes (||) character. For example:

```
PROC1      PROCESS SNODE=TCPNAME=THIS.IS.A.LONG.DNS.NAME.THAT. ||      -
           WONT.FIT.ON.ONE.LINE                                     -
           HOLD=NO                                                -
```

If you use SNODE=TCPNAME=tcpvalue or SNODE=UDT33NAM=udtvalue, Processes in HO WC status do not automatically restart when the node becomes available and another Process is submitted to that node. However, if SNODE = secondary-node-name is used, Processes in HO WC status restart, automatically.

SNODE

For a Copy statement, this specifies the secondary node, defining the direction of transfer (with PNODE). When SNODE is specified with the FROM parameter, a receive takes place. When SNODE is specified with the TO parameter, a send takes place. SNODE is the default for the TO parameter.

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

For a Run Task statement, this 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 IBM Connect:Direct on the specified node.

For a Run Job statement, this specifies that the job is to be submitted on the secondary node (SNODE), which is the node that interacts with the PNODE.

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

Specifies security user IDs and passwords at the SNODE.

For z/OS and UNIX, each value can be 1-64 alphanumeric characters. For other operating environments, each value can be 1-8 alphanumeric characters, unless otherwise noted.

For Connect:Direct for i5/OS, if the SNODEID and password is not specified in the PROCESS statement, Connect:Direct for i5/OS uses the user ID and password of the Process submitter for the security ID and password check.

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

For Connect:Direct for HP NonStop, this subparameter specifies the HP NonStop group number and user number. These numbers can range from 0-255. When specifying an HP NonStop value, you must use a period (.) as a separator between the group number and the user number.

For Connect:Direct for i5/OS, this subparameter specifies the i5/OS user profile used for authorization checks during Process execution and is limited to 8 characters.

If the SNODE DTF can authenticate using a PassTicket password, then specifying an SNODE user ID 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 *IBM Sterling Connect:Direct for z/OS 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.

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 for HP NonStop, SAFEGUARD must be running on the HP NonStop.

This subparameter is ignored for Connect:Direct for i5/OS.

SPACE =(CYL | TRK | blk | 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 specified 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 specified 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.
- **blk** 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 specified, 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.

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 you do not specify date or day is not specified, precede the time with a comma.

Do not specify **STARTT** with **RETAIN=INITIAL**.

If you specify both **HOLD=YES** and a **STARTT** value in a Process, the **HOLD** specification takes precedence, and the Process is placed in the Hold queue.

date specifies that the Process starts on a specific date. Depending on the value of the **DATEFORM** initialization parameter, you can specify the date in one of the following formats:

DATEFORM Value	Formats			
DATEFORM=MDY (default)	mm/dd/yy	mm/dd/yyyy	mm.dd.yy	mm.dd.yyyy
DATEFORM=DMY	dd/mm/yy	dd/mm/yyyy	dd.mm.yy	dd.mm.yyyy
DATEFORM=YMD	yy/mm/dd	yyyy/mm/dd	yy.mm.dd	yyyy.mm.dd
DATEFORM=YDM	yy/dd/mm	yyyy/dd/mm	yy.dd.mm	yyyy.dd.mm

You can omit the period or slash separators for transfers between mainframe nodes.

If you only specify a date, the time defaults to 00:00.

If you specify **RETAIN=YES**, you cannot specify a date in the **STARTT** parameter.

Valid Julian date formats are **yyddd**, **yyyyddd**, **yy/ddd**, **yyyy/ddd**, **yy.ddd**, or **yyyy.ddd**.

day specifies the day of the week to release the Process for execution. Valid names include **MONday**, **TUesday**, **WEdnesday**, **THursday**, **FRiday**, **SATurday**, and **SUNday**. You can abbreviate the day value 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.

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.

You can specify the time of day using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00:00 to 24:00:00. If you do not use AM and PM, the 24-hour clock is assumed.

If you use the 12-hour clock, 01:00:00 hours can be expressed as 1:00AM, and 13:00 hours can be expressed as 1:00PM.

If you specify hh:mm:ssXM with RETAIN=YES, the Process executes at the same time every day. You do not need to specify minutes and seconds.

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

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 z/OS: 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.

SUBMIT

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

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.

SUBNODE=SNODE is not valid if communicating with a node running Connect:Direct for i5/OS.

SYMBOL

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

&symbolicName1 = variable string 1

&symbolicName2 = variable string 2

.

.

.

&symbolicName n = variable string n

Specifies the default value for a symbolic parameter in the Process. The value is substituted within the Process when the symbolic parameter is encountered. You can override this default in the SUBMIT command.

The symbolic_name parameter must begin with an ampersand. The maximum length of &symbolic_name (including the ampersand) is 9 characters. The maximum length of the variable-string is 256 characters.

Do not use identical symbolic_names. Enclose any parameters containing special characters in single quotation marks.

Note: Depending on how you use a variable string, you may need to include bracketing characters. This situation is often required when a SYSOPTS string is sent as a symbolic parameter and must be enclosed in quotation marks.

For example, to transfer a file to a UNIX system using a symbolic variable, you would type the SYSOPTS clause as follows:

```
&SYSOPTS="\":datatype=text:xlate=yes:"\
```

In this example, what the Process states:

```
SYSOPTS=&SYSOPTS
```

resolves to:

```
SYSOPTS="":datatype=text:xlate=yes:"
```

Connect:Direct for z/OS provides the following intrinsic symbolic variables that you can use to substitute user-defined values when a Process is executed. This flexibility lets you use the same Process for multiple applications when these values change.

Value	Description
%DD2DSN	Specifies an allocated DD statement, which references a DSN to be passed to a Process being submitted (for Connect:Direct for z/OS)
%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. 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.
%JOBID	Specifies the job number.
%JOBNM	Specifies the job name.
%JUSER	Specifies a variable that resolves to the USERID of the submitted job.
%NUM1	Specifies the submission time of the Process in minutes, seconds, and fraction of seconds in the format mmssth.
%NUM2	Specifies the submitted time of a Process as the low order 4 bits of the milliseconds of the time expressed as 1 hex digit (a value from 0 through 15 expressed as 0 through F).
%PNODE	PNODE name where the submit occurs
%PRAND	Pseudo-random number (6 hex digits)

Value	Description
%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. The value returned can be used to create a file name on the node receiving the file.
%SUBDATE1	Use this parameter to substitute the submitted date in the yyyyymmdd 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 ddmyyyy 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. The return value can be used to create a file name on the node receiving the file. 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	Specifies a variable that resolves to the user submitting the Process

SYSOUT=(sysout_keyword1, sysout_keyword2, . .)

Refer to the *IBM Sterling Connect:Direct for z/OS Facilities Guide* for a list and description of the SYSOUT keywords.

SYSOPTS="DBCS=(tablename,so,si, PAD | PAD=pc, LOGIC=A | B | (B,RC))"

"CODEPAGE=(from code set, to Unicode code set)"

"parameter1[parameter2,...]"

"DATATYPE=TEXT | BINARY"

"XLATE=NO | YES"

"STRIP.BLANKS=NO | YES"

"PRECOMP=Y | YES | N | NO"

"ZEDC=ON | OFF"

"ZIIP=ON | OFF"

specifies the FROM system operation parameters.

DBCS=(tablename, so, si, PAD|PAD=pc, LOGIC=A|B| (B,RC)) is used to invoke the double-byte character set (DBCS) 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. For an updated list of translation tables provided by IBM Connect:Direct, see *Supporting DBCS* in *IBM Sterling Connect:Direct for z/OS Administration Guide*.

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'.

LOGIC=A | B | (B, RC) tells IBM Connect:Direct how to process data when it encounters SO (SHIFT-OUT) and SI (SHIFT-IN) characters. Normal mainframe data processing expects a pairing of SO/SI characters, and generates an error and terminates a Process when these SO/SI characters are not equally matched. You specify the DBCS keyword within the SYSOPTS parameter, the parameters that define the translation table and the SO and SI characters, and whether padding characters are in use. For example, the following two examples of a SYSOPTS parameter mean the same thing and tell IBM Connect:Direct to use this normal method to process DBCS data (LOGIC=A):

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F,PAD=00)"
```

or

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F,PAD=00,LOGIC=A)"
```

The initial state of the data is SBCS. With normal logic translation processing, you are not required to specify the keyword, Logic=A. With normal logic, the mainframe system processes the data as SBCS until it encounters an SO character. After encountering the SO character, the system expects DBCS characters. IBM Connect:Direct generates an error and terminates the Process if it encounters one of the following conditions:

- An SI character before an SO character
- A combination of SI and SO characters without data between the SI and SO characters

An alternate processing method exists. When you specify LOGIC=B, special rules apply to the normal DBCS translation. The system tells IBM Connect:Direct to send the file before generating an error. LOGIC=B tells IBM Connect:Direct to continue processing and keep the state as SBCS if the system encounters one of the following conditions:

- An SI character before a SO character
- An invalid combination of SI and SO characters

The system processes the data in the state that was in effect prior to the invalid SI-SO combination and sends the entire file. After sending the file, the system generates an error.

Alternate Translation allows for the file to be completely transferred and an error (message SCPA074I with return code 8) noted at the completion of the transmission. You also have the option of using LOGIC=B,RC to specify a different return code within the range of 1-254 for this specific message. If you specify LOGIC=B without specifying a return code, the default return code of 8 is used. The following coding examples show how to specify a return code for SCPA074I and the expected results for each:

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F,LOGIC=(B,4))"
```

In the example above, if the DBCS transfer produces the SCPA074I message, the return code generated will be 4.

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F,LOGIC=(B,254))"
```

In the example above, if the DBCS transfer produces the SCPA074I message, the return code generated will be 254 or X'FE'.

CODEPAGE=(from code set, to Unicode code set) invokes code set conversion utilities supported by the z/OS system.

- **from code set** is the name of the code set of the original data. The code set name is required and can be any z/OS-compatible IBM code set, such as IBM-930, IBM-932, IBM-942, and IBM-943, or LOCALE to indicate the default code page set for the sending node.
- **to Unicode code set** is the name of the code set on the local node that will be used as the intermediate conversion format. The code set name is required and can be UTF-8 or UCS-2, or their equivalent on other operating systems. For example, 65001 is the UTF-8 equivalent on the Microsoft Windows system.

The code set specifications are only validated for basic syntax. An invalid code set specification will produce an error message on the node attempting conversion.

A code set value of LOCALE specifies the default code set for the node performing conversion.

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 for z/OS 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.

PRECOMP=YES|NO specifies that IBM Connect:Direct will automatically decompress a file that was precompressed with the CDSACOMP utility. See the *IBM Sterling Connect:Direct for z/OS User Guide* for more information on the CDSACOMP utility.

YES indicates that the FROM data set is precompressed and tells IBM Connect:Direct to decompress the file as part of the Process.

NO tells IBM Connect:Direct to send the file in compressed format. This value works only when copying to another Connect:Direct for z/OS system that can decompress the file using the CDSACOMP utility.

If SYSOPTS are not coded or if SYSOPTS='PRECOMP=NO', the file is sent in compressed format and the receiver must run CDSACOMP with MODE=DECOMP.

ZEDC=ON | OFF specifies whether the ZEDC hardware compression should be ON (enabled) or OFF (disabled) for this data set.

ZIIP=ON | OFF specifies whether the extended compression should be attempted to be offload to the ZIIP processor (ON) or not attempted (OFF).

SYSOPTS = "UNIQUE=YES"

"DBCS=(tablename, so, si, PAD | PAD=pc, LOGIC=A | B | (B, RC))"

"CODEPAGE=(from Unicode code set, to code set)"

"DATATYPE=TEXT | BINARY"
"XLATE=NO | YES"
"STRIP.BLANKS=NO | YES"
"PERMISS=nnn"
"SYSOUT=(sysout_keyword1, sysout_keyword2, . .)"
"ZEDC=ON | OFF"
"ZIIP=ON | OFF"

specifies the TO 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 z/OS 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, LOGIC=A|B| (B, RC)) 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. For an updated list of translation tables provided by IBM Connect:Direct, see *Supporting DBCS* in *IBM Sterling Connect:Direct for z/OS Administration Guide*.

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'.

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'.

LOGIC=A|B| (B, RC) tells IBM Connect:Direct how to process data when it encounters SO (SHIFT-OUT) and SI (SHIFT-IN) characters. Normal mainframe data processing expects a pairing of SO/SI characters, and generates an error and terminates a Process when these SO/SI characters are not equally matched. You specify the DBCS keyword within the SYSOPTS parameter, the parameters that define the translation table and the SO and SI characters, and whether padding characters are in use. For example, the following two examples of a SYSOPTS parameter mean the same thing and tell IBM Connect:Direct to use this normal method to process DBCS data (LOGIC=A):

```
SYSOPTS="DBCS=(EBCXKSC, 0E, 0F, PAD=00) "
```

or

```
SYSOPTS="DBCS=(EBCXKSC, 0E, 0F, PAD=00, LOGIC=A) "
```

The initial state of the data is SBCS. With normal logic translation processing, you are not required to specify the keyword, Logic=A. With normal logic, the mainframe system processes the data as SBCS until it encounters an SO character. After encountering the SO character, the system expects DBCS characters. IBM Connect:Direct generates an error and terminates the Process if it encounters one of the following conditions:

- An SI character before an SO character
- A combination of SI and SO characters without data between the SI and SO characters

An alternate processing method exists. When you specify LOGIC=B, special rules apply to the normal DBCS translation. The system tells IBM Connect:Direct to send the file before generating an error. LOGIC=B tells IBM Connect:Direct to continue processing and keep the state as SBCS if the system encounters one of the following conditions:

- An SI character before a SO character
- An invalid combination of SI and SO characters

The system processes the data in the state that was in affect prior to the invalid SI-SO combination and sends the entire file. After sending the file, the system generates an error.

Alternate Translation allows for the file to be completely transferred and an error (message SCPA074I with return code 8) noted at the completion of the transmission. You also have the option of using LOGIC=B,RC to specify a different return code within the range of 1-254 for this specific message. If you specify LOGIC=B without specifying a return code, the default return code of 8 is used. The following coding examples show how to specify a return code for SCPA074I and the expected results for each:

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F,LOGIC=(B,4))"
```

In the example above, if the DBCS transfer produces the SCPA074I message, the return code generated will be 4.

```
SYSOPTS="DBCS=(EBCXKSC,0E,0F,LOGIC=(B,254))"
```

In the example above, if the DBCS transfer produces the SCPA074I message, the return code generated will be 254 or X'FE'.

CODEPAGE=(from Unicode set, to code set) invokes code set conversion utilities supported by the z/OS C/C++ compiler.

- **from Unicode set** is the name of the Unicode set of the encoded data sent to the receiving node. The code set name is required and can be UTF-8 or UCS-2, or their equivalent on other operating systems. For example, 65001 is the UTF-8 equivalent on the Microsoft Windows system.
- **to code set** is the name of the final code set that will be used on the remote node. You can use LOCALE to indicate the default code page set relevant to the receiving 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.

Note: To honor the permission setting for HFS files using the PERMISS keyword, set the UNIX System Services UMASK to 000, either by default or by using the runtime environment variable, `_EDC_UMASK_DFLT`. To set the environment variable, define the `_EDC_UMASK_DFLT=000` variable

in a RECFM=VB type file and allocate the ENVIRON DD in the IBM Connect:Direct startup JCL. For example:

```
//ENVIRON DD DISP=SHR,DSN=$CD.ENVIRON(TZ)
```

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, **permis=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.

SYSOUT=(sysout_keyword1, sysout_keyword2, . . .) specifies various SYSOUT keywords. See the *IBM Sterling Connect:Direct for z/OS Facilities Guide* for a list and description of the SYSOUT keywords.

THEN

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

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 z/OS data set name conventions, enclose the data set name 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, z/OS issues a WTOR requesting the password when Connect:Direct for z/OS software opens the data set. For example:

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

If the data set 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) 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. The PNODE setting for the THIRD.DISP.DELETE initialization parameter controls how IBM Connect:Direct processes this subparameter's value when an ABEND occurs in the COPY step on the SNODE. 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. CATLG is the default.
- **DELETE** specifies the system deletes the file if the Process step terminates abnormally.

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 z/OS to OpenVMS copies: For z/OS 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 for OpenVMS COPY statement.
- For OpenVMS to z/OS 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 specified 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 IBM Connect:Direct does dynamic deallocation of data sets. However, if RETAIN is omitted, a comma must be specified. 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.

Chapter 3. Transmit Files between IBM Connect:Direct and B2B Integrator

If your company uses both B2B Integrator and IBM Connect:Direct to perform business activities, route data, and transfer files, you can configure the Connect:Direct Server Adapter on the B2B Integrator side to act as either a PNODE, SNODE, or both.

For more information about how to work within B2B Integrator and use both the Connect:Direct Server Adapter and its related services, see *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf, which is part of the B2B Integrator documentation library.

This webpage contains information about how to create IBM Connect:Direct Processes to communicate with B2B Integrator either by copying files to a B2B Integrator mailbox or business process. In addition, this page provides information for the reverse direction, that is, when B2B Integrator as the PNODE is either copying (pushing) a file to a IBM Connect:Direct node or retrieving (pulling) a file to a IBM Connect:Direct node. For detailed information, refer to the following topics in *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf:

- *Use B2B Integrator to Copy a File to IBM Connect:Direct*
- *Use B2B Integrator to Copy a File from IBM Connect:Direct*
- *Sample Business Processes*

Sterling B2B Integrator-Connect:Direct Server Adapter Syntax

Transmit Files between IBM Connect:Direct and B2B Integrator

If your company uses both B2B Integrator and IBM Connect:Direct to perform business activities, route data, and transfer files, you can configure the Connect:Direct Server Adapter on the B2B Integrator side to act as either a PNODE, SNODE, or both.

For more information about how to work within B2B Integrator and use both the Connect:Direct Server Adapter and its related services, see *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf, which is part of the B2B Integrator documentation library.

This webpage contains information about how to create IBM Connect:Direct Processes to communicate with B2B Integrator either by copying files to a B2B Integrator mailbox or business process. In addition, this page provides information for the reverse direction, that is, when B2B Integrator as the PNODE is either copying (pushing) a file to a IBM Connect:Direct node or retrieving (pulling) a file to a IBM Connect:Direct node. For detailed information, refer to the following topics in *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf:

- *Use B2B Integrator to Copy a File to IBM Connect:Direct*
- *Use B2B Integrator to Copy a File from IBM Connect:Direct*
- *Sample Business Processes*

COPY Statement Keywords for Communicating with B2B Integrator

After you have set up a B2B Integrator mailbox or business process to which you want to send a file, you can create the IBM Connect:Direct Process to perform the file transfer.

(For instructions on how to create a B2B Integrator mailbox or business process, see *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf.)

Send to a B2B Integrator Mailbox

To send to a B2B Integrator mailbox, use the following general syntax:

```
COPY FROM source file information
COPY TO FILE=/mailbox/yourMailbox/yourDestinationFile
```

The following IBM Connect:Direct Process example copies *yourSourceFile* from your IBM Connect:Direct server to the B2B Integrator-Connect:Direct Server Adapter, *your CDSANode*, which passes the document to your mailbox. The SNODEID, *yourUserID*, is your B2B Integrator User ID.

```
COPY2MB PROCESS
STEP1 COPY SNODE=yourCDSANode
          SNODEID=(yourUserID,yourPassword)
          FROM (
              FILE=yourSourceFile
            )
          TO (
              FILE=/mailbox/yourMailbox/yourDestinationFile
              DISP=RPL
            )
```

There are three parts to the destination FILE name in this Copy step:

- *mailbox* is a reserved word and signals to B2B Integrator that you are copying to a Mailbox.
- *yourMailbox* includes the name of your B2B Integrator Mailbox.
- *yourDestinationFile* is the document that is passed to your mailbox by the Connect:Direct Server Adapter. This document is stored in the mailbox. This document can be retrieved by an SFTP client, business process or another IBM Connect:Direct process.

After you submit the IBM Connect:Direct Process, you can use IBM Connect:Direct Select Statistics to monitor your IBM Connect:Direct to B2B Integrator processes. For more information on how to use the SELECT STATISTICS command to determine the outcome of a completed Process, refer to the IBM Connect:Direct documentation for the platform you are using.

You can also view *yourDestinationFile* in your B2B Integrator mailbox. See *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf for more information on this B2B Integrator procedure.

When setting up and organizing your B2B Integrator mailboxes, you may want to consider virtual roots. For more information, refer to *Using Virtual Roots in a Mailbox Hierarchy* in *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf.

For more IBM Connect:Direct Process examples, see [B2B Integrator Connect:Direct Server Adapter Examples](#).

Send to a B2B Integrator Business Process

To send to a B2B Integrator business process, use the following general syntax:

```
COPY FROM source file information
COPY TO FILE=/businessprocess/yourBusinessProcess/yourDestinationFile
```

The following IBM Connect:Direct Process example copies *yourSourceFile* from your IBM Connect:Direct server to the B2B Integrator-Connect:Direct Server Adapter, which invokes your business process. The SNODEID, *yourUserID*, is your B2B Integrator User ID.

```

COPY2SI  PROCESS
          SNODE=yourCDSANode
          SNODEID=(yourUserID,yourPassword)
STEP1    COPY
          FROM (
              FILE=yourSourceFile
            )
          TO   (
              FILE=/businessprocess/yourBusinessProcess/yourDestinationFile
              DISP=RPL
            )

```

There are three parts to the destination FILE name in this Copy step:

- */businessprocess/* is a reserved word and signals to B2B Integrator that you are copying to a business process.
- *yourBusinessProcess* is the name of the B2B Integrator business process that the Connect:Direct Server Adapter invokes.
- *yourDestinationFile* is the name of the document that is passed to the business process by the Connect:Direct Server Adapter. This document becomes the Primary Document in the *yourBusinessProcess* business process. The primary document is the document that the services in a business process act on or in relation to; this is usually the document passed to a business process by the initiating Adapter.

After you submit your IBM Connect:Direct Process, you can use both IBM Connect:Direct and B2B Integrator to monitor this process. See *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf for more information.

For more IBM Connect:Direct Process examples, see [B2B Integrator Connect:Direct Server Adapter Examples](#).

Retrieve from a B2B Integrator Business Process

To retrieve a file from a B2B Integrator business process using IBM Connect:Direct, use the following general syntax:

```

COPY FROM /businessprocess/yourBusinessProcess/yourSourceFile
COPY TO  FILE=yourDestinationFile

```

The following IBM Connect:Direct Process example retrieves *yourSourceFile* from the B2B Integrator-Connect:Direct Server Adapter which receives the file from your business process.

```

PULLFILE  PROCESS
          SNODE=yourCDSANode
          SNODEID=(yourUserID,yourPassword)
STEP1     COPY
          FROM (
              FILE=/businessprocess/yourBusinessProcess/yourSourceFile
              SNODE
            )
          TO   (
              FILE=yourDestinationFile
              PNODE
              DISP=RPL
            )
PEND

```

There are three parts to the FROM FILE name in the Copy step:

- */businessprocess/* is a reserved word and signals to B2B Integrator that you are invoking a business process.
- *yourBusinessProcess* is the name of the B2B Integrator business process that the IBM Connect:Direct Server Adapter invokes.

- *yourSourceFile* is the name of the document that is passed from the B2B Integrator-Connect:Direct Server Adapter to your IBM Connect:Direct node. This document is the Primary Document in your business process.

After you submit your IBM Connect:Direct Process, you can use both IBM Connect:Direct and B2B Integrator to monitor this process. See *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf for more information.

For more IBM Connect:Direct Process examples, see [B2B Integrator Connect:Direct Server Adapter Examples](#).

Retrieve from a B2B Integrator Mailbox

To retrieve a file from a B2B Integrator mailbox using IBM Connect:Direct, use the following general syntax:

```
COPY FROM /mailbox/yourMailbox/yourSourceFile
COPY TO FILE=yourDestinationFile
```

The following IBM Connect:Direct process example retrieves *yourSourceFile* from the B2B Integrator-Connect:Direct Server Adapter which picks up the file in your B2B Integrator mailbox.

```
PULLFILE PROCESS
STEP1 COPY SNODE=yourCDSANode
          SNODEID=(yourUserID,yourPassword)
          FROM (
                FILE=/mailbox/yourMailbox/yourSourceFile
                SNODE
                )
          TO (
              FILE=yourDestinationFile
              PNODE
              DISP=RPL
              )
```

There are three parts to the FROM FILE name in the Copy step:

- */mailbox/* is a reserved word and signals to B2B Integrator that you are retrieving from a mailbox.
- *yourMailbox* includes the name of the B2B Integrator mailbox.
- *yourSourceFile* is the name of the document that is passed from the B2B Integrator mailbox by the B2B Integrator-Connect:Direct Server Adapter to your IBM Connect:Direct node.

After you submit the IBM Connect:Direct Process, you can use IBM Connect:Direct Select Statistics to monitor your IBM Connect:Direct Process, including the source file. For more information on how to use the SELECT STATISTICS command to determine the outcome of a completed Process, refer to the IBM Connect:Direct documentation for the platform you are using.

In B2B Integrator, you can also view *yourSourceFile*, which is stored as a message in your mailbox. See *IBM Sterling Connect:Direct Interoperability* at ftp://public.dhe.ibm.com/software/commerce/doc/mft/cdsa/si_cdsa_user_guide.pdf for more information on this B2B Integrator procedure.

For more IBM Connect:Direct Process examples, see [B2B Integrator Connect:Direct Server Adapter Examples](#).

Specify File Formats using SYSOPTS

To specify the format of a file including operations to be performed on the file before sending to B2B Integrator, identify all customizing features by using SYSOPTS parameters in the COPY FROM clause. By specifying the SYSOPTS parameters appropriate for the platform from which you are sending the source file, the destination file will arrive on the B2B Integrator side the way you want it to.

For example, in the following scenario, the file, SourceFileABC, is being copied from the Connect:Direct for UNIX node to the B2B Integrator mailbox Inbox directory for the SI_UserID. The file is to be sent as is (datatype=binary) with no character translation and will be named DestinationFileXYZ on the B2B Integrator side.

```
copy_file process snode=SI_CDSA_Node snodeid=(SI_UserID,yourPassword)
step01    copy
          from
            (
              file = /CDUNIX_node/SourceFileABC
              pnode
              sysopts=":datatype=binary:xlate=no:"
            )
          ckpt = 2M
          compress ext
          to
            (
              file = /mailbox/Inbox/DestinationFileXYZ
              snode
              disp = new
            )
          pend
```

Submit a B2B Integrator Business Process from IBM Connect:Direct

To submit a B2B Integrator business process from a IBM Connect:Direct PNODE to an SNODE B2B Integrator-Connect:Direct Server Adapter, use the following syntax:

```
SUBMIT business process name
```

For more IBM Connect:Direct Process examples, see [B2B Integrator Connect:Direct Server Adapter Examples](#).

IBM Connect:Direct Parameters for Communicating with B2B Integrator

(FROM) FILE=/businessprocess/*yourBusinessProcess*/*yourSourceFile* | /mailbox/*yourMailbox*/*yourSourceFile*

specifies the source document in a B2B Integrator business process or mailbox. Specify a slash (/), followed by the keyword, businessprocess or mailbox, followed by the name of your business process or mailbox, a slash (/), and the name of the source document file.

(TO) FILE=/businessprocess/*yourBusinessProcess*/*yourDestinationFile* | /mailbox/*yourMailbox*/*yourDestinationFile*

specifies the destination document in a B2B Integrator business process or mailbox. Specify a slash (/), followed by the keyword, businessprocess or mailbox, followed by the name of your business process or mailbox, a slash (/), and the name of the destination document file.

business process name

indicates the specific B2B Integrator business process being submitted from a IBM Connect:Direct node.

Sterling B2B Integrator-Connect:Direct Server Adapter Examples

B2B Integrator Connect:Direct Server Adapter Examples

This topic provides some examples of IBM Connect:Direct Processes that interoperate with B2B Integrator.

Copy a File from Connect:Direct for z/OS to a B2B Integrator Mailbox

In this example, the PNODE is a Connect:Direct for z/OS server (LA.ZOS) copying its source file (the LA.3Q.SALES data set) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named ATLANTA_CDSA. The destination file to be created on the B2B Integrator side will be called West3QRevenue and placed in the AtlantaHQ mailbox.

```
COPYTOMB    PROCESS    SNODE=ATLANTA_CDSA          -
              SNODEID=(Atlanta_UserID,Atlanta_Password)          -
              PNODE=LA.ZOS                                         -
STEP1       COPY FROM (DSN=LA.3Q.SALES)                            -
              TO (FILE=/mailbox/AtlantaHQ/West3QRevenue           -
                  DISP=(NEW))                                     -
```

Copy a File from Connect:Direct for UNIX to a B2B Integrator Mailbox

In this example, the PNODE is a Connect:Direct for UNIX server (NY_UNIX1) copying its source file (daily.txt) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA2. The destination file to be created on the B2B Integrator side will be called orders.dly and placed in the NYInbox mailbox.

```
COPY2UX PROCESS
              SNODE=CDSA2
              SNODEID=(myUserID,myPassword)
              PNODE=NY_UNIX1
STEP1  COPY  FROM (
              FILE=/orders/daily.txt
              )
              TO (
              FILE=/mailbox/NYInbox/orders.dly
              DISP=NEW
              )
PEND
```

Copy a File from Connect:Direct for Microsoft Windows to a B2B Integrator Mailbox

In this example, the PNODE is a Connect:Direct for Microsoft Windows server (WEST_WINA) copying its source file (branch3.txt) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server

Adapter named SI_REGION1. The destination file to be created on the B2B Integrator side will be called branch3.inv and placed in the accounting mailbox.

```

COPY2WIN PROCESS
          SNODE=SI_REGION1
          SNODEID=(myUserID,myPassword)
          PNODE=WEST_WINA
STEP1    COPY
          FROM (
              FILE=C:\invoices\branch3.txt
            )
          TO (
              FILE=/mailbox/accounting/branch3.inv
              DISP=NEW
            )
PEND

```

Copy a File from Connect:Direct for HP NonStop to a B2B Integrator Mailbox

In this example, the PNODE is a Connect:Direct for HP NonStop server copying datafile in \$B.smith to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA_NODE. The destination file to be created on the B2B Integrator side will be called trans and placed in the daily mailbox.

```

cdsa1 process      snode=CDSA_NODE snodeid=(user,user1)
step1 copy from   (file=$B.smith.datafile          -
                  pnode)                          -
                to (file='/mailbox/daily/trans '    -
                  snode                            -
                  disp = new)                      -
                  ckpt=128K

```

Copy a File from Connect:Direct for i5/OS to a B2B Integrator Mailbox

In the following CDSND command example, the PNODE is a Connect:Direct for i5/OS system copying DALLIB/ACCTDATA to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named DALLAS_CDSA. The destination file to be created on the B2B Integrator side will be called daily.dat and placed in the invoice mailbox.

```

CDSND  PNAME(CDSEDMB)  SNODE(DALLAS_CDSA)
        SNODENVIRN(UNIX)
        FDSN('DALLIB/ACCTDATA'(MBR1'))
        TDSN('/mailbox/invoice/daily.dat')
        FMSYSOPTS('TYPE(MBR)')

```

Copy a File from Connect:Direct for z/OS to a B2B Integrator Business Process

In this example, the PNODE is a Connect:Direct for z/OS server (CD.MF6) copying its source file (the ORDERS.BATCH data set) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named SI_BRANCH. The destination file to be created on the B2B Integrator side will be called branch.ord and copied to the SendToBilling business process as the Primary Document.

```

COPY2BP PROCESS SNODE=SI_BRANCH -
                SNODEID=(myUserID,myPassword) -
                PNODE=CD.MF6 -
STEP1 COPY FROM (DSN=ORDERS.BATCH) -
                TO (FILE=/businessprocess/SendToBilling/branch.ord -
                   DISP=(NEW))

```

Copy a File from Connect:Direct for UNIX to a B2B Integrator Business Process

In this example, the PNODE is a Connect:Direct for UNIX server (NY_UNIX1) copying its source file (daily.txt) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA_V51. The destination file to be created on the B2B Integrator side will be called orders.dly and copied to the ProcessNYOrders business process as the Primary Document.

```
COPYFMUX PROCESS
                SNODE=CDSA_V51
                SNODEID=(myUserID,myPassword)
                PNODE=NY_UNIX1
STEP1  COPY
        FROM (
            FILE=/orders/daily.txt
        )
        TO   (
            FILE=/businessprocess/ProcessNYOrders/orders.dly
            DISP=RPL
        )
PEND
```

Copy a File from Connect:Direct for Microsoft Windows to a B2B Integrator Business Process

In this example, the PNODE is a Connect:Direct for Microsoft Windows server (CD_WIN_CENTRAL) copying its source file (merch.txt) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named SICDSA. The destination file to be created on the B2B Integrator side will be called returns.dly and copied to the UpdateInventory business process as the Primary Document.

```
CPYFMWIN PROCESS
                SNODE=SICDSA
                SNODEID=(myUserID,myPassword)
                PNODE=CD_WIN_CENTRAL
STEP1  COPY
        FROM (
            FILE=C:\returns\merch.txt
        )
        TO   (
            FILE=/businessprocess/UpdateInventory/returns.dly
            DISP=RPL
        )
PEND
```

Copy a File from Connect:Direct for HP NonStop to a B2B Integrator Business Process

In this example, the PNODE is a Connect:Direct for HP NonStop server copying datafile in \$B.smith to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA_NODE. The destination file to be created on the B2B Integrator side will be called orders and copied to the ProcessOrders business process as the Primary Document.

```
cdsa2 process      snode=CDSA_NODE snodeid=(user,user1)
step1 copy from    (file=$B.smith.datafile
                    pnode)
                    to (file='/businessprocess/ProcessOrders/orders
                        snode
                        disp = new)
                    ckpt=128K
```

Copy a File from Connect:Direct for i5/OS to a B2B Integrator Business Process

In the following CDSND command example, the PNODE is a Connect:Direct for i5/OS system copying DALLIB/ACCTDATA to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named DALLAS_CDSA. The destination file to be created on the B2B Integrator side will be called daily.dat and copied to the ProcessInvoice business process as the Primary Document.

```
CDSND  PNAME(CDSENBP)  SNODE(DALLAS_CDSA)
       SNODENVIRN(UNIX)
       FDSN('DALLIB/ACCTDATA')
       TDSN('/businessprocess/ProcessInvoice/daily.dat')
       FMSYSOPTS('TYPE(MBR)')
```

Retrieve a File from a B2B Integrator Business Process to Connect:Direct for z/OS

In this example, the PNODE is a Connect:Direct for z/OS server (CD.ZOS.PROD) retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named MAIN_CDSA. The source file (outbound.billing) is copied from the primary document of the TransferBilling business process. The destination file to be created on the IBM Connect:Direct server is called BILLING.DAT.

```
SENBILL  PROCESS          SNODE=MAIN_CDSA          -
          SNODEID=(myUserID,myPassword)      -
          PNODE=CD.ZOS.PROD                    -
STEP1    COPY FROM (FILE=/businessprocess/TransferBilling/outbound.billing -
          SNODE )                               -
          TO (DSN=BILLING.DAT                   -
          PNODE                                 -
          DISP=(NEW)                            -
```

Retrieve a File from a B2B Integrator Business Process to Connect:Direct for Microsoft Windows

In this example, the PNODE is a Connect:Direct for Microsoft Windows server (CDWIN_9) retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named SI51CDSA. The source file (daily.txt) is copied from the primary document of the RespondToCD business process. The destination file to be created on the IBM Connect:Direct server is called orders.dat.

```
PULL2WIN  PROCESS
          SNODE=SI51CDSA
          SNODEID=(myUserID,myPassword)
          PNODE=CDWIN_9
STEP1     COPY
          FROM (
              FILE=/businessprocess/RespondToCD/daily.txt
              SNODE
            )
          TO (
              FILE=C: ransactions\orders.dat
              PNODE
              DISP=RPL
            )
PEND
```

Retrieve a File from a B2B Integrator Business Process to Connect:Direct for UNIX

In this example, the PNODE is a Connect:Direct for UNIX server (UNIX_NODEA) retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA2. The

source file (trans.dat) is copied from the primary document of the SendTransactions business process. The destination file to be created on the IBM Connect:Direct server is called orders.txt.

```

PULLTRAN  PROCESS
          SNODE=CDSA2
          SNODEID=(myUserID,myPassword)
          PNODE=UNIX_NODEA
STEP1     COPY
          FROM (
              FILE=/businessprocess/SendTransactions/trans.dat
              SNODE
              )
          TO   (
              FILE=/cust1/orders.txt
              PNODE
              DISP=RPL
              )
PEND

```

Retrieve a File from a B2B Integrator Business Process to Connect:Direct for HP NonStop

In this example, the PNODE is a Connect:Direct for HP NonStop server retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA_NODE. The source file (orders) is copied from the primary document of the ProcessOrders business process. The destination file to be created on the IBM Connect:Direct server is datafile in \$B.smith.

```

cdsa3  process  snode=CDSA_NODE snodeid=(user,user1)
step1  copy    from (file='/businessprocess/ProcessOrders/orders ' -
                  snode) -
          to     (file=$B.smith.datafile -
                  pnode -
                  disp=(RPL, ,DELETE)) -
                  ckpt=128K

```

Retrieve a File from a B2B Integrator Business Process to Connect:Direct for i5/OS

In the following CDRCV command example, the PNODE is a Connect:Direct for i5/OS system retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named DALLAS_CDSA. The source file (daily.dat) is copied from the primary document of the ProcessInvoice business process. The destination file to be created on the IBM Connect:Direct server is DALLIB/ACCDATA.

```

CDRCV  PNAME(CDRCVBP)  SNODE(DALLAS_CDSA)
       SNODENVIRN(UNIX)
       FDSN('/businessprocess/ProcessInvoice/daily.dat')
       TDSN('DALLIB/ACCDATA'(DAILY)'))
       TOSYSOPTS('TYPE(MBR)')

```

Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for z/OS

In this example, the PNODE is a Connect:Direct for z/OS system (CD.ZOS.PROD) retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named SICDSA. The source file (mainframe.txt) is copied from the Outbound mailbox on B2B Integrator. The destination file to be created on the IBM Connect:Direct system is called DAILY.DAT.

```

GETDAILY  PROCESS          SNODE=SICDSA                -
          SNODEID=(myUserID,myPassword) -
          PNODE=CD.ZOS.PROD
STEP1     COPY  FROM      (FILE=/mailbox/Outbound/mainframe.txt -
                          SNODE ) -
          TO      (DSN=DAILY.DAT -
                  PNODE -
                  DISP=(NEW) ) -

```

Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for Microsoft Windows

In this example, the PNODE is a Connect:Direct for Microsoft Windows server (WIN_CD2) retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CENTRAL_CDSA. The source file (annual.rpt) is copied from the Chicago mailbox on B2B Integrator. The destination file to be created on the IBM Connect:Direct server is called annual.txt.

```

GETANRPT  PROCESS          SNODE=CENTRAL_CDSA
          SNODEID=(myUserID,myPassword)
          PNODE=WIN_CD2
STEP1     COPY  FROM      (
                          FILE=/mailbox/Chicago/annual.rpt
                          SNODE
                          )
          TO      (
                  FILE=C:\acctg\annual.txt
                  PNODE
                  DISP=RPL
                  )
PEND

```

Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for UNIX

In this example, the PNODE is a Connect:Direct for UNIX server (CD_UX_TEST) retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named SI50. The source file (contact.lst) is copied from the Sales mailbox on B2B Integrator. The destination file to be created on the IBM Connect:Direct server is called prospect.txt.

```

RTRVLIST  PROCESS          SNODE=SI50
          SNODEID=(myUserID,myPassword)
          PNODE=CD_UX_TEST
STEP1     COPY  FROM      (
                          FILE=/mailbox/Sales/contact.lst
                          SNODE
                          )
          TO      (
                  FILE=/contacts/prospect.txt
                  PNODE
                  DISP=RPL
                  )
PEND

```

Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for HP NonStop

In this example, the PNODE is a Connect:Direct for HP NonStop server retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA_NODE. The source file (orders) is copied from the DailyTransactions mailbox on B2B Integrator. The destination file to be created on the IBM Connect:Direct server is \$B.smith.datafile.

```

cdsa4 process      snode=CDSA_NODE snodeid=(user,user1)
step1 copy from   (file='/mailbox/DailyTransactions/orders ' -
                  snode) -
                  to   (file=$B.smith.datafile -
                        pnode -
                        disp=(RPL, ,DELETE)) -
                        ckpt=128K

```

Retrieve a File from a B2B Integrator Mailbox to Connect:Direct for i5/OS

In the following CDRCV command example, the PNODE is a Connect:Direct for i5/OS system retrieving a file from the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named DALLAS_CDSA. The source file (daily.dat) is copied from the invoice mailbox on B2B Integrator. The destination file to be created on the IBM Connect:Direct server is DALLIB/ACCTDATA.

```

CDRCV  PNAME(CDRCVMB)  SNODE(DALLAS_CDSA)
       SNODENVIRN(UNIX)
       FDSN('mailbox/invoice/daily.dat')
       TDSN('DALLIB/ACCTDATA(DAILY)')
       TOSYSOPTS('TYPE(MBR)')

```

Submit a B2B Integrator Business Process from Connect:Direct for z/OS

In this example, the PNODE is a Connect:Direct for z/OS system (CD.NY.ZOS) submitting a B2B Integrator business process (the PROCESS.DAILY.ORDER member) located in the NYLIBRARY to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named LA_SI.

```

SUBBP  PROCESS  SNODE=LA_SI -
              SNODEID=(myUserID,myPassword) -
              PNODE=CD.NY.ZOS
STEP1  SUBMIT   DSN=NYLIBRARY(PROCESS.DAILY.ORDER)

```

Submit a B2B Integrator Business Process from Connect:Direct for UNIX

In this example, the PNODE is a Connect:Direct for UNIX server (CD_UX_PROD) submitting a B2B Integrator business process (UpdateInvoices) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CD_ACCTG.

```

SUBUPDT PROCESS
              SNODE=CD_ACCTG
              SNODEID=(myUserID,myPassword)
              PNODE=CD_UX_PROD
STEP1  SUBMIT  FILE=UpdateInvoices
PEND

```

Submit a B2B Integrator Business Process from Connect:Direct for Microsoft Windows

In this example, the PNODE is a Connect:Direct for Microsoft Windows server (WINCD_BRANCH) submitting a B2B Integrator business process (ProcessOrders) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA_TULSA.

```

SUBUPDT PROCESS
              SNODE=CDSA_TULSA
              SNODEID=(myUserID,myPassword)
              PNODE=WINCD_BRANCH
STEP1  SUBMIT  FILE=ProcessOrders
PEND

```

Submit a B2B Integrator Business Process from Connect:Direct for HP NonStop

In this example, the PNODE is a Connect:Direct for HP NonStop system (CD.HPNS.PROD) submitting a B2B Integrator business process (ProcessDuplicates) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA.NODE.

```
cdsa5      process      pnode=CD.HPNS.PROD
                        snode=CDSA_NODE
step1      submit      snodeid=(user,user1)
                        DSN=ProcessDuplicates
                        SUBNODE=SNODE
```

Submit a B2B Integrator Business Process from Connect:Direct for i5/OS

In this CDSUBMIT command example, the PNODE is a Connect:Direct for i5/OS system submitting a B2B Integrator business process (ProcessInvoice) to the SNODE, which is, in this case, a B2B Integrator-Connect:Direct Server Adapter named CDSA_TULSA. The process name is CDSUBSI.

```
CDSUBMIT SNODE(CDSA_TULSA) SNODENVIRN(UNIX) PROCFILE(ProcessInvoice)
PNAME(CDSUBSI)
```


Chapter 4. Basic PROCESS Statement

This example illustrates the minimum requirements of a PROCESS statement. The label CD1 must begin in column one. PROCESS is the statement identifier and is required. The only required parameter is SNODE, which specifies the secondary node to be used in the IBM Connect:Direct Process.

```
CD1  PROCESS  SNODE=CD.VM.NODE
```

Basic PROCESS Statement

This example illustrates the minimum requirements of a PROCESS statement. The label CD1 must begin in column one. PROCESS is the statement identifier and is required. The only required parameter is SNODE, which specifies the secondary node to be used in the IBM Connect:Direct Process.

```
CD1  PROCESS  SNODE=CD.VM.NODE
```

Detailed PROCESS Statement Using the RETAIN Parameter

According to parameters specified, this Process runs in CLASS 4 and is assigned the highest priority, 15. The parameter RETAIN=INITIAL causes the Process to remain in the queue (TCQ) after execution and run every time the IBM Connect:Direct system is initialized. In addition, accounting data is specified for both nodes (PACCT and SACCT).

```
CDACCT  PROCESS  SNODE=CD.OS390.NODE  -  
          CLASS=4  -  
          PRTY=15  -  
          PACCT='ACCOUNTING DATA FOR PNODE' -  
          RETAIN=INITIAL -  
          SACCT='INFORMATION FOR SNODE'
```

Detailed PROCESS Statement Using the NOTIFY Parameter

The priority for this Process is set to 8 and runs in CLASS 4. Because of the NOTIFY parameter, USER1 will be notified upon Process completion.

For Connect:Direct for UNIX, Connect:Direct for VSE, Connect:Direct for OpenVMS, and Connect:Direct for HP NonStop nodes, NOTIFY is ignored.

```
PROC1  PROCESS  -  
        SNODE=CD.AS400 -  
        PRTY=8  -  
        NOTIFY=USER1 -  
        CLASS=4  -  
        SNODEID=(USER1,PWD)
```

Detailed PROCESS Statement Using Long Passphrases

This example illustrates a sample PROCESS statement for entering long passphrases when IBM Connect:Direct for z/OS delimiter characters are not used.

```
CD1 PROCESS SNODE=CD.ZOS.NODE -  
  PNODEID=(USERID, -  
  This.is.a.long.passphrase.and.there.must.be.room.to.specify.it) -  
  SNODEID=(RUSERID, -  
  This.is.another.long.passphrase.and.it.needs.room.also)
```

Use %PNODE

This Process shows using a variable to substitute to %PNODE. %PNODE is set to the node name from which the Process is submitted. (%PNODE is only valid for Connect:Direct for z/OS nodes.)

```
PROC1  PROCESS      &NODE=%PNODE
STEP1  COPY FROM    (PNODE DSN=JSMITH.FILE DISP=SHR) -
        TO          (SNODE DSN=JSMITH.FILE DISP=NEW)
        IF (STEP1 EQ 0)
            RUN TASK (PGM=USERPGM                -
                    PARM='&NODE, &DSN')
        EIF
```

Specify a List of Ciphers in a Particular Process

This example involves overriding default settings in the Connect:Direct Secure Plus parameter files used to establish a connection between two business partners. The business partners agreed by default all sessions are non-secure but that when a secure communication line is required for a particular session, they would use the SSL protocol and a list of cipher suites in a specific order.

Although the SSL protocol is not enabled in the Connect:Direct Secure Plus parameter files, the remote node records specify OVERRIDE=Y, and all other parameters required to perform the handshake to establish an SSL session are defined.

To specify that the session for this PROCESS is to be secure using SSL and to tell IBM Connect:Direct to use a specific list of cipher suites, the business partners use the following PROCESS statement:

```
SSLCIPHERS PROCESS SNODE=OTHERBP
SECURE=(SSL,(SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_RSA_AES_128_SHA,
SSL_RSA_AES_256_SHA,SSL_RSA_WITH_DES_CBC_SHA) )
```

The four cipher suites are listed in the order of preference, and the first one that matches a cipher suite defined for the other node is used to establish a session.

Encrypt Only Control Block Information—No Data Being Sent

This example involves overriding default settings in the Connect:Direct Secure Plus parameter files used to establish a connection between two business partners. The business partners agreed by default all sessions are secure and that everything should be encrypted, that is, both the information sent during the handshake to set up communication sessions and the actual files being transferred.

Both partners specified the following configuration in their Connect:Direct Secure Plus parameter files:

- Specified ENCRYPT=Y in both the Local and Remote Node records
- Specified OVERRIDE=Y in both the Local and Remote Node records

To avoid going through the expense of encrypting and decrypting data being transferred, the business partners use the following PROCESS statement when transferring a particular file:

```
ENCNO PROCESS SNODE=OTHERBP SECURE=ENCRYPT.DATA=N
```

In this scenario, both business partners are more concerned with increasing throughput and using less CPU while protecting the information being exchanged to establish the session.

Note: Both sides must have support for ENCRYPT.DATA=N or the Process fails.

Use %DD2DSN to Pass DSN from JCL to a Process

In this example, IBM Connect:Direct uses the DSN specified on the allocated FROMDD statement in the JCL and a randomly generated six- digit number to copy to a data set with a unique name.

```
TEST  PROCESS  SNODE=S_NODE          -
                &RAN = %PRAND         -
                &DSN = %DD2DSN(FROMDD)
STEP  COPY    FROM (DSN=&DSN) TO (XXX.T&RAN)
```

Use a TCP/IP Address for the SNODE Keyword (IPv4)

This Process shows how to code the TCP/IP address using the IPv4 standard dotted-name format when the real address is known to the user.

```
PROC1  PROCESS  PNODE=CD.OS390.NODE    -
                SNODE=TCPNAME=111.222.333.444 -
                NOTIFY=USER1
```

Use a TCP/IP Address for the SNODE Keyword (IPv6)

This Process shows how to code the TCP/IP address using the IPv6 format with colons when the real address is known to the user.

```
PROC1  PROCESS  PNODE=CD.OS390.NODE    -
                SNODE=TCPNAME=1111::6666:7777:8888 -
                NOTIFY=USER1
```

Use TCPNAME to Identify the PNODE/SNODE Sites

This Process shows how the user can identify the TCP/IP nodes by name when the real network addresses for the nodes are unknown. The names specified must be identified to the TCP/IP network name resolution task.

```
PROC1  PROCESS  PNODE=CD.OS390.NODE    -
                SNODE=RESTON.SCENTER   -
                NOTIFY=USER1
```

HP NonStop PROCESS Statement

The following is an example HP NonStop PROCESS statement.

```
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 user IDs 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.

Example OpenVMS PROCESS Statement

The following is a example PROCESS statement.

```
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 user IDs and passwords (SNODEID) are 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.

After the Process executes it will be deleted because the RETAIN parameter is set to NO.

Example VMESA Process

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

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

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.

After the Process executes, it is deleted because the RETAIN parameter is set to NO.

VSE PROCESS Statement

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

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

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.

After the Process executes, it is deleted because the RETAIN parameter is set to NO.

Use Symbolics in a UNIX Process

This example shows a UNIX Process that uses symbolics to specify the file and data set names at 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
```

Copy a File from Connect:Direct for HP NonStop to Connect:Direct for z/OS

This Process copies a file from a Connect:Direct for HP NonStop node to a new SMS-controlled file at a Connect:Direct for z/OS node.

```
SMS1  PROCESS  PNODE=CD.HPNONSTOP      -
              SNODE=CD.OS390.NODE      -
STEP1  COPY    FROM  (PNODE             -
                    DSN=$VOL.SUBVOL.TANFILE -
                    DISP=SHR             -
                    TO  (SNODE           -
                    DISP=(NEW,CATLG)     -
                    DSN=DATA1.SEQ.OS390FILE -
                    DCB=(DSORG=PS, LRECL=80) -
                    SPACE=(80,(1500,100),RLSE) -
                    UNIT=SYSDA           -
                    SYSOPTS="AVGREC=U    -
                    MGMTCLAS=TEST01     -
                    STORCLAS=TEST01     -
                    DATACLAS=TEST01")
```

Copy a File from Connect:Direct for HP NonStop to Connect:Direct for z/OS

This Process copies a file from a Connect:Direct for HP NonStop node to a new SMS-controlled file at a Connect:Direct for z/OS node.

SMS1	PROCESS	PNODE=CD.HPNONSTOP	-
		SNODE=CD.OS390.NODE	
STEP1	COPY	FROM (PNODE	-
		DSN=\$VOL.SUBVOL.TANFILE	-
		DISP=SHR	-
		TO (SNODE	-
		DISP=(NEW,CATLG)	-
		DSN=DATA1.SEQ.OS390FILE	-
		DCB=(DSORG=PS, LRECL=80)	-
		SPACE=(80,(1500,100),RLSE)	-
		UNIT=SYSDA	-
		SYSOPTS="AVGREC=U	-
		MGMTCLAS=TEST01	-
		STORCLAS=TEST01	-
		DATACLAS=TEST01")	-

Copy a File from Connect:Direct for HP NonStop to Connect:Direct for z/OS

This Process copies a file from a Connect:Direct for HP NonStop node to a new SMS-controlled file at a Connect:Direct for z/OS node.

SMS1	PROCESS	PNODE=CD.HPNONSTOP	-
		SNODE=CD.OS390.NODE	
STEP1	COPY	FROM (PNODE	-
		DSN=\$VOL.SUBVOL.TANFILE	-
		DISP=SHR	-
		TO (SNODE	-
		DISP=(NEW,CATLG)	-
		DSN=DATA1.SEQ.OS390FILE	-
		DCB=(DSORG=PS, LRECL=80)	-
		SPACE=(80,(1500,100),RLSE)	-
		UNIT=SYSDA	-
		SYSOPTS="AVGREC=U	-
		MGMTCLAS=TEST01	-
		STORCLAS=TEST01	-
		DATACLAS=TEST01")	-

Submit a Process from a Connect:Direct for HP NonStop Node that Copies a File from z/OS to HP NonStop

This Process, submitted from Connect:Direct for HP NonStop, copies (pulls) a file from Connect:Direct for z/OS to the HP NonStop node. The TO file specification contains a "SET TYPE E" parameter in SYSOPTS, in case the file is not present and must be created. The FROM file normally needs no file attributes (DCB or SYSOPTS). The XLATE option is used to specify translation. This must always be specified in the SYSOPTS clause associated with the HP NonStop dataset. A checkpoint interval of 10 MB is specified.

EPROC	PROCESS	SNODE=CD.OS390	-
		SNODEID=(UID,USER_PASSWORD)	
PULL	COPY TO	(PNODE DISP=RPL	-
		DSN=\$WORK.DATA.EFILE	-
		SYSOPTS="SET TYPE E XLATE ON")	-
		FROM (SNODE DISP=SHR	-
		DSN=OFILES.ACH.D120897)	-
		CKPT=10M	

Copy a Code 0 Unstructured file from HP NonStop to z/OS

This Process copies an unstructured file to a z/OS node. Because an Enscribe unstructured file has no intrinsic record length, the DCB parameter is used on the FROM file specification to define the record length with which the file is read.

When an unstructured file is sent from an HP NonStop system, the default LRECL is the "buffersize" attribute of the file, unless overridden by the DCB parameter.

```
UNSTR    PROCESS    SNODE=CD.OS390
STEP1    COPY FROM  (PNODE DISP=SHR          -
                    DSN=$WORK02.HPDATA.UNST1 -
                    DCB=(LRECL=100))         -
                    TO    (SNODE DISP=NEW     -
                    DSN=DATA1.VSAME001.DATA -
                    DCB=(DSORG=PS          -
                        ,RECFM=FB          -
                        ,LRECL=100        -
                        ,BLKSIZE=10000))    -
                    CKPT=1M
```

Copy a File from HP NonStop to z/OS After Running DMRTDYN on z/OS

In this Process, STEP1 will invoke DMRTDYN to delete and unallocate DATA1.SEQ.KSDSFILE at the SNODE. Then \$B.FILETEST.KSDSFILE will be copied from the HP NonStop node to DATA1.SEQ.VSAMKSDS at the HP NonStop node.

```
VSAMKS   PROCESS    PNODE=CD.HPNONSTOP          -
STEP1     RUN TASK   (SNODE=CD.OS390.NODE          -
                    (PGM=DMRTDYN,                -
                    PARM=(C'ALLOC DSN=DATA1.SEQ.KSDSFILE -
                    DISP=(OLD,DELETE)'           -
                    F'-1'                         -
                    C"UNALLOC DSN=DATA1.SEQ.KSDSFILE") -
                    SNODE)                       -
STEP2     COPY FROM (DSN=$B.FILETEST.KSDSFILE     -
                    PNODE                       -
                    DISP=SHR)                   -
                    TO    (DSN=DATA1.SEQ.VSAMKSDS -
                    DISP=NEW                     -
                    SPACE=(2048,(2048,10))      -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=1024,BLKSIZE=2048))
```

Copy a File from an HP NonStop Spooler to a z/OS Node

In this multi-step Process, STEP1 will execute a RUN TASK to invoke DMRTDYN at the SNODE to delete DATA1.SEQ.EXTOS390F. STEP2 will copy a file from the HP NonStop spooler to DATA1.SEQ.EXTOS390F at the SNODE. Because a spooler job number was not specified, the most recent job in the spooler for the job owner will be copied. A record format of FBA is used to maintain ANSI control characters.

```
EXTOS390 PROCESS    PNODE=CD.HPNONSTOP          -
STEP1     RUN TASK   (SNODE=SS.CD.OS390          -
                    (PGM=DMRTDYN,                -
                    PARM=(C'ALLOC DSN=DATA1.SEQ.EXTOS390F, -
                    DISP=(OLD,DELETE)'           -
                    F'-1'                         -
                    C"UNALLOC DSN=DATA1.SEQ.EXTOS390F")) -
                    SNODE                       -
STEP2     COPY FROM (DSN=\TANEXT.$S.#EXTTANF     -
                    PNODE                       -
                    DISP=SHR)                   -
                    TO    (DSN=DATA1.SEQ.EXTOS390F -
                    SNODE                       -
                    DISP=(NEW,CATLG)            -
                    DCB=(DSORG=PS,RECFM=FBA,LRECL=132, -
                    BLKSIZE=13200)             -
                    SPACE=(13200,(10,2))      -
                    UNIT=SYSDA)
```

Copy Between a z/OS Node and a Remote HP NonStop Spooler on an EXPAND Network

This multi-step Process copies between a z/OS node and a remote HP NonStop spooler on an EXPAND network. The Process is submitted from the HP NonStop node. STEP01 copies a file from a DSN on the z/OS node to the HP NonStop spooler. STEP02 copies a file from the HP NonStop spooler to a z/OS node.

Note that \SYSEXT is an EXPAND node other than the one that the Connect:Direct for HP NonStop server resides on. Because the HP NonStop files are spooler files, the SYSOPTS SET XLATE subparameter does not have to be specified for translation; spooler files and edit files (unstructured, code 101) are translated automatically. A record format of FBA is used to maintain ANSI control characters.

```

SPL    PROCESS    SNODE=CD.SMITH
STEP01 COPY FROM (DSN=JSMITH.CDACT SNODE -
                DISP=SHR) -
                TO (FILE='\SYSEXT.$$.#SPL1' PNODE -
                DISP=RPL) -
STEP02 COPY FROM (FILE='\SYSEXT.$$.#SPL2' PNODE -
                DISP=SHR) -
                TO (DSN=RJONES.ACCT SNODE -
                DCB=(RECFM=FBA,DSORG=PS,LRECL=132, -
                BLKSIZE=13200) -
                SPACE=(13200,(10,2)) -
                DISP=RPL)

```

Copy Files Between the HP NonStop Spooler System and a z/OS Node Using Job Number

This Process transfers a file from the HP NonStop spooler \$\$S to a z/OS sequential file. The HP NonStop file has a job number of 3722 and a location (name) of #SPLFILE. The SPOOLER command is used to specify a supervisor other than the default of \$SPLS. A record format of FBA is used to maintain ANSI control characters.

```

SPLPROC PROCESS    SNODE=CD.OS390.JSMITH
STEP01  COPY FROM (DSN=\SYSEXT.$$.#SPLFILE -
                SYSOPTS=("SET SPOOLER=$SPLA" -
                "SET SPOOLNUM=3722") -
                DISP=SHR) -
                TO (DSN=RJONES.SPLFILE -
                DCB=(RECFM=FBA,DSORG=PS,LRECL=132, -
                BLKSIZE=13200) -
                SPACE=(1320,(10,2)) -
                DISP=RPL)

```

Copy a Disk File from HP NonStop to a Tape Device at z/OS

This Process is submitted on the HP NonStop node to copy a disk file on HP NonStop to a tape device at the z/OS node. Because NL (no labels) is specified, DCB attributes must be specified to identify the file on the volume.

```

PROCESS1 PROCESS    PNODE=CD.HPNONSTOP -
                SNODEID=(RJONES,ROGER) -
                SNODE=CD.OS390
STEP01  COPY FROM (DSN=$C.BILLPROC.ACCTDATA -
                DISP=SHR) -
                TO (DSN=RJONES.ACCTTAPE -
                DISP=RPL -
                DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=4096) -
                VOL=(, , 1) -
                LABEL=(,NL, ,EXPDT=91044) -
                UNIT=REEL)

```


Copy a Tape File from z/OS to a Disk File on HP NonStop

This Process is submitted at the HP NonStop node to copy a tape file from the z/OS node to a disk file on the HP NonStop node. Because NL (no labels) is specified, DCB attributes must be specified to identify the file on the volume. SYSOPTS is used to specify file creation parameters specific to HP NonStop.

PROCESS1	PROCESS	PNODE=CD.HPNONSTOP	-
		SNODEID=(RJONES,ROGER)	-
		SNODE=CD.OS390	-
STEP01	COPY FROM	(DSN=RJONES.ACCTTAPE	-
		SNODE	-
		UNIT=REEL	-
		VOL=SER=010196	-
		LABEL=(,NL,,EXPDT=91044)	-
		DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=4096)	-
		DISP=SHR)	-
	TO	(PNODE	-
		DSN=\$C.BILLPROC.ACCTDATA	-
		DISP=RPL	-
		SYSOPTS=("SET TYPE U"	-
		"SET BLOCK 4096"	-
		"SET EXT (5,5)")	-

Copy a File from z/OS to HP NonStop Using the FASTLOAD Option

This Process is submitted at the HP NonStop node to copy an entry-sequenced file from z/OS to a key-sequenced file at the HP NonStop node. The SYSOPTS subparameter SET FASTLOAD SORTED sets FASTLOAD and indicates to FUP that the data is sorted.

This option (particularly useful for key-sequenced files) will bypass invocation of FASTSORT by FUP. The FASTLOAD option can be used to reduce disk I/O overhead. The XLATE subparameter is included in the Process to turn on the text conversion utility and translate from EBCDIC to ASCII.

FASTLOAD	PROCESS	PNODE=CD.HPNONSTOP	-
		SNODE=CD.OS390	-
STEP01	COPY FROM	(DSN=OS390.ESDS.FILE	-
		DISP=SHR SNODE)	-
	TO	(DSN=\$C.DATA.KSDS	-
		DISP=RPL	-
		SYSOPTS=("SET FAST.LOAD.SORTED Y"	-
		"SET XLATE ON") PNODE)	-

Allocate a VSAM Data Set and Copying a File from HP NonStop to z/OS

This Process invokes IBM Connect:Direct DMRTAMS for VSAM file allocation. DMRTAMS (using IDCAMS commands specified in the Process) first deletes the old data set if it exists. It then defines and allocates it for use in the subsequent COPY statement. The file transmitted is an HP NonStop key-sequenced file.

VSAM01	PROCESS	SNODE=SC.OS390.RSMITH PNODE=CDCLX	-
		SNODEID=(RSMITH,RSMITH)	-
		SACCT='CHARGE TO ACCOUNTING'	-
STEP1	RUN TASK	(PGM=DMRTAMS,	-
		PARM=(C"FREE=CLOSE,RETURN=(DD) SYSOUT=X"	-
		C" DELETE (RSMITH.VSAM01.OUTPUT) CLUSTER	-
		C" DEFINE CLUSTER	-
		C" (NAME (RSMITH.VSAM01.OUTPUT)	-
		C" RECORDS(2500 100)	-
		C" VOLUMES(M80003)	-
		C" INDEXED	-
		C" FREESPACE(0 0)	-
		C" NOIMBED	-
		C" KEYS (8 6)	-
		C" RECORDSIZE(262 880)	-
		C" NOREPLICATE	-
		C" SHAREOPTIONS(2)	-
		C" DATA	-
		C" (CONTROLINTERVALSIZE(4096)	-
		C" NAME (RSMITH.VSAM01.OUTPUT.DATA2)	-
		C" INDEX	-
		C" (CONTROLINTERVALSIZE(512)	-
		C" NAME (RSMITH.VSAM01.OUTPUT.INDEX2))	-
		SNODE	-
STEP2	COPY FROM	(DSN=\$B.CDFILES.KDSDFIL DISP=SHR	-
		PNODE)	-
	TO	(DSN=RSMITH.VSAM01.OUTPUT DISP=SHR	-
		SNODE)	-

Copy to an Entry-Sequenced File (HP NonStop to HP NonStop)

In this multi-step Process, STEP01 will execute FUP to purge \$B.FILERESO.A11025 on the PNODE. A message will be sent to the spooler (\$S.#FUPTTEST) that indicates whether FUP executed successfully.

STEP02 will copy \$B.FILEDATA.ETYPE from the PNODE to the entry-sequenced file, \$B.FILERESO.A110257, at the SNODE. Specifying the SYSOPTS subparameter TYPE=E ensures that file attribute defaults defined in the type file E will be used when creating the file.

A110257	PROCESS	PNODE=CD.TANA	-
		SNODE=CD.TANB	-
STEP01	RUN TASK	(PGM=FUP	-
		SYSOPTS=(' /OUT \$S.#FUPTTEST/'	-
		' VOLUME \$B.FILERESO',	-
		' PURGE A110257 ')	-
		PNODE)	-
STEP02	COPY FROM	(DSN=\$B.FILEDATA.ETYPE	-
		PNODE	-
		DISP=SHR)	-
	TO	(DSN=\$B.FILERESO.A110257	-
		SNODE	-
		DISP=NEW	-
		SYSOPTS="SET TYPE E")	-

Copy Files Between HP NonStop Spooler Systems

This Process selects a spooler job by job number and copies it to another HP NonStop spooler system at a remote node. The HP NonStop file being copied has a job number of 3722 and a location (name) of #SPLFILA. The SPOOLER command is used to specify a supervisor other than the default of \$SPLS.

```
SPLPROC  PROCESS      PNODE=CD.TANA          -
          SNODE=CD.TANB
STEP01   COPY  FROM    (DSN=\SYSA.$S.#SPLFILA          -
                      SYSOPTS=("SET SPOOLER=$SPLA"    -
                                "SET SPOOLNUM=3722")   -
                      DISP=SHR PNODE)                 -
          TO           (DSN=\SYSB.$S.#SPLFILB          -
                      DISP=NEW SNODE)
```

Copy Text Files from HP NonStop to UNIX

This Process, submitted from the Connect:Direct for HP NonStop node, will copy datafile in \$B.smith to / payroll/monthly/jan on the Connect:Direct for UNIX node. For Connect:Direct for UNIX nodes, the security user IDs and passwords are case sensitive.

```
unix1  process      snode=unix.node snodeid=(user,user1)
step1  copy  from    (file=$B.smith.datafile          -
                    pnode)                            -
                    to  (file='/payroll/monthly/jan '  -
                    snode)                            -
                    sysopts=":datatype=text:"        -
                    disp = rpl)                       -
                    ckpt=128K
```

Copy Binary Files from HP NonStop to UNIX

This Process, submitted from the Connect:Direct for HP NonStop node, will copy a file in \$user.unixdata.cdcom to a binary file on the Connect:Direct for UNIX node.

```
TOBIN01 PROCESS      SNODE=cd.v1200          -
                    snodeid=(user,pswd)
STEP01  COPY  FROM    (DSN=$user.unixdata.cdcom        -
                    PNODE)                            -
                    SYSOPTS=('SET XLATE OFF')         -
                    DISP=SHR)                        -
                    TO  (dsn='tdata/cdcomu'           -
                    SNODE)                            -
                    SYSOPTS=":datatype=binary:"      -
                    DISP=RPL)
```

Copy Binary Files from HP NonStop (OSS) to UNIX

This Process illustrates how to ensure data integrity when copying between a IBM Connect:Direct HP NonStop (OSS) node and a UNIX node. The destination in STEP01 and source in STEP02 both have sysopts specified to denote a binary transfer.

OUOBIN	PROCESS	SNODE=qa160sun3600	-
		SNODEID=(qatest,qatest)	
	SYMBOL	&FILE1='/home/qatest/input/ndmsrvr'	
	SYMBOL	&FILE2='/export/home/users/qatest/tantest/binary'	
	SYMBOL	&FILE3='/home/qatest/output/unix/binary'	
RUNUNIX	RUN TASK	(PGM="UNIX") SNODE	-
		SYSOPTS=("rm -rf &FILE2")	
STEP01	COPY TO	(DSN=&FILE2 SNODE	-
		DISP = RPL	-
		SYSOPTS=":datatype=binary:")	-
	FROM	(DSN=&FILE1 PNODE	-
		DISP = SHR)	
STEP02	COPY FROM	(DSN=&FILE2 SNODE	-
		DISP = SHR	-
		SYSOPTS=":datatype=binary:")	-
	TO	(DSN=&FILE3 PNODE	-
		DISP = RPL)	

Copy Files from HP NonStop to VM

This Process illustrates the transmission of a file from an HP NonStop node to a VM node. The Process is submitted on the HP NonStop node. EBCDIC-to-ASCII translation is requested with the SYSOPTS parameter SET XLATE ON. All SYSOPTS keyword values must be enclosed in parentheses, and the entire SYSOPTS string must be enclosed in double quotation marks.

Use this Process when you copy files from HP NonStop to VM.

TANTOVM	PROCESS	PNODE=DALL.TX	-
		SNODE=CD.VM.BOSTON	-
		SNODEID=(IDXXXX,PASSWD)	-
		SACCT='TRANSFERRING FROM HP NONSTOP TO VM.'	-
STEP01	COPY FROM	(DSN=\$B.SMITH.DATAFILE	-
		DISP=(SHR)	-
		SYSOPTS=("SET XLATE ON")	-
		PNODE)	-
	TO	(DSN='TEST FILE'	-
		LINK=(TRA,WPSWD,W,191)	-
		DISP=(RPL)	-
		DCB=(RECFM=F, LRECL=80, BLKSIZE=80)	-
		SNODE)	

Copy an HP NonStop Key-Sequenced File to a Connect:Direct for OpenVMS Node

This Process, submitted from the Connect:Direct for HP NonStop node, copies a key-sequenced file to the Connect:Direct for OpenVMS node. When Processes are submitted from Connect:Direct for HP NonStop to Connect:Direct for OpenVMS nodes, OpenVMS file names must be in single quotation marks. Note that COMPRESS is coded between the FROM and TO clauses of the COPY statement.

SEND1	PROCESS	PNODE=CD.HPNONSTOP	-
		SNODE=CD.VMS	
STEP01	COPY FROM	(PNODE FILE=\$B.TESTF.KSDS	-
		DISP=SHR)	-
		COMPRESS PRIMECHAR=C'*'	-
	TO	(SNODE FILE=' [DUC4:-DATA.FILE]KSDS1.DAT'	-
		DCB=(DSORG=KSDS	-
		LRECL=100	-
		KEYLEN=10	-
		RECFM=F)	-
		DISP=RPL)	

Copy Files from HP NonStop to VSE

This Process illustrates the transmission of files from an HP NonStop node to a VSE node. The parameter XLATE must be set to ON and positioned in the Process where the HP NonStop file is specified. XLATE translates the file from ASCII to EBCDIC. Symbolics will be resolved at submission.

SEND2VSE	PROCESS	PNODE=BOSTON.NODE	-
		SNODE=CD.VSE.NODE	-
		SNODEID=(IDXXXX,PSWD)	-
		SACCT='CHARGE TO GROUP 199'	-
TAN_VSE	COPY FROM	(DSN=&FROM	-
		SYSOPTS=("SET XLATE ON")	-
		PNODE)	-
	TO	(DSN=&TO	-
		UNIT=DLBONLY	-
		DISP=(OLD)	-
		DCB=(RECFM=FB, LRECL=80, BLKSIZE=80, DSORG=PS)	-
		SNODE)	-

Copy Binary Files from Microsoft Windows to HP NonStop (OSS)

This Process illustrates how to ensure data integrity when copying between a Microsoft Windows system and an HP NonStop (OSS) node. The destination in STEP01 and source in STEP02 both have sysopts specified to denote a binary transfer.

BINARY	PROCESS	SNODE=w2S.4100.Cdwops8	-
		SNODEID=(qatest,qatest)	-
	SYMBOL	&FILE1='/home/qatest/input/ndmsrvr'	-
	SYMBOL	&FILE2='c:\output\binary'	-
	SYMBOL	&FILE3='/home/qatest/output/win/binary'	-
RUNWIN	RUN TASK	(PGM="WindowsNT")	-
		SNODE	-
		SYSOPTS="CMD(ERASE &FILE2)"	-
STEP01	COPY TO	(DSN=&FILE2	-
		SNODE	-
		DISP = RPL	-
		SYSOPTS="datatype(binary)")	-
	FROM	(DSN=&FILE1	-
		PNODE	-
		DISP = SHR)	-
STEP02	COPY FROM	(DSN=&FILE2	-
		SNODE	-
		DISP = SHR	-
		SYSOPTS="datatype(binary)")	-
	TO	(DSN=&FILE3	-
		PNODE	-
		DISP = RPL)	-

Copy a File from HP NonStop to VM

This example copies a file from HP NonStop 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
STEP01  COPY FROM    (FILE=$B.ABC.FILEA PNODE
                   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

```

Copy a Data Set from a Spooler File on IBM Connect:Direct HP NonStop to IBM Connect:Direct z/OS

This example statement copies a data set from a spooler file on a Connect:Direct for HP NonStop node to a IBM Connect:Direct z/OS 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)

```

Copy a File From HP NonStop to HP NonStop and Overriding SENDOPENFILE with OPENFILEXMT

```

COPY FROM (DSN=$dev.datain.openes -
           SYSOPTS="SET OPENFILEXMT=Y" -
           PNODE DISP=SHR) -
TO (DSN=$dev.dataout.openfres SNODE DISP=RPL) -
   CKPT=20k

```

Copy an HP NonStop File to an i5/OS Node

This Process copies an HP NonStop file to a member of a physical data base file on i5/OS. The i5/OS file is created with maximum members specified as 100. The HP NonStop file is translated from ASCII to EBCDIC.

```

TESTPROC PROCESS      PNODE=CD400 -
                   SNODE=CDTAN -
                   SNODEID=(USER1,PASSWRD)
STEP01   COPY FROM    (DSN=$SYSTEM.SOURCE.FILE -
                   DISP=SHR PNODE) -
                   TO    (DSN='CDTAN/SOURCE(FILE)' -
                   DISP=NEW SNODE -
                   SYSOPTS=("TYPE(MBR) -
                              MAXMBS(100)")

```

Copy an HP NonStop File to an i5/OS Node

This Process copies an HP NonStop file to a member of a physical data base file on i5/OS. The i5/OS file is created with maximum members specified as 100. The HP NonStop file is translated from ASCII to EBCDIC.

```
TESTPROC  PROCESS      PNODE=CD400          -
                               SNODE=CDTAN                  -
                               SNODEID=(USER1,PASSWRD)      -
STEP01    COPY  FROM    (DSN=$SYSTEM.SOURCE.FILE          -
                               DISP=SHR PNODE)              -
                               SYSOPTS=("SET XLATE ON")      -
                               TO    (DSN=' CDTAN/SOURCE(FILE)' -
                               DISP=NEW SNODE                -
                               SYSOPTS=("TYPE(MBR)         -
                               MAXMBS(100)"))
```

Copy a File from Microsoft Windows to z/OS

This Process copies a text file from a remote Microsoft Windows system (where IBM Connect:Direct is not installed) to a z/OS partitioned data set (PDS). When copying a file to or from a remote Microsoft Windows computer, you must use the Universal Naming Convention (UNC) method for specifying the file name. Do not use a drive letter. The UNC consists of two backslashes, the computer name, a single backslash, and the share name. The share, ROOT_C, must be accessible to Connect:Direct for Microsoft Windows server. In this example, the computer name is WIN_SYS1 and the share name is ROOT_C.

```
NT20S390  PROCESS      SNODE=SS.0S390/*$0S390$*/
                               HOLD=NO
                               CLASS=1
                               PRTY=10
                               EXECPRTY=10
                               RETAIN=NO
STEP01    COPY  FROM    (FILE=\WIN_SYS1\ROOT_C\DATA\OUT\SALESJAN.DAT
                               PNODE/*$WINDOVS$*/
                               SYSOPTS="datatype(text)")
                               TO    (SNODE/*$0S390$*/
                               FILE=SALES.DATA.JAN(MBR99)
                               DISP=(RPL,CATLG))
PEND
```

Copy a File from Microsoft Windows to z/OS

This Process copies a text file from a remote Microsoft Windows system (where IBM Connect:Direct is not installed) to a z/OS partitioned data set (PDS). When copying a file to or from a remote Microsoft Windows computer, you must use the Universal Naming Convention (UNC) method for specifying the file name. Do not use a drive letter. The UNC consists of two backslashes, the computer name, a single backslash, and the share name. The share, ROOT_C, must be accessible to Connect:Direct for Microsoft Windows server. In this example, the computer name is WIN_SYS1 and the share name is ROOT_C.

```
NT20S390  PROCESS      SNODE=SS.0S390/*$0S390$*/
                               HOLD=NO
                               CLASS=1
                               PRTY=10
                               EXECPRTY=10
                               RETAIN=NO
STEP01    COPY  FROM    (FILE=\WIN_SYS1\ROOT_C\DATA\OUT\SALESJAN.DAT
                               PNODE/*$WINDOVS$*/
                               SYSOPTS="datatype(text)")
                               TO    (SNODE/*$0S390$*/
                               FILE=SALES.DATA.JAN(MBR99)
                               DISP=(RPL,CATLG))
PEND
```

Copy a File from Microsoft Windows to HP NonStop

This Process copies a text file from a Microsoft Windows system to an HP NonStop node. Each system operation is enclosed in single quotation marks. Double quotation marks enclose the entire SYSOPTS statement.

```
NT2TAN  PROCESS      SNODE=SS.TAN/*$HP NONSTOP$*/
                                HOLD=NO
                                CLASS=1
                                PRTY=10
                                EXECPRTY=10
                                RETAIN=NO
                                PNODEID=(USR1,PSWD1)
                                SNODEID=(SS.USER01,PSWD01)
STEP01  COPY  FROM  (FILE=C:\OUTPUT\BINARY\SALES.JAN
                                PNODE/*$WINDOWS$*/
                                SYSOPTS="DATATYPE(BINARY)")
                                TO  (SNODE/*$HP NONSTOP$*/
                                FILE=$SALES.DATA.JAN)
                                DISP=(RPL, ,DELETE)
                                SYSOPTS="'SET CODE 0' 'SET TYPE U' 'SET EXT(700 300'
                                'SETBLOCK 4096' 'SETMAXEXTENTS600'")
                                COMPRESS PRIMECHAR=X'20'

PEND
```

Wildcard Copies from Microsoft Windows to UNIX

In the following example, a Connect:Direct for Microsoft Windows PNODE directory C:\financial\accounts contains the files customer1, customer2, customer3, supplier1, and supplier2. A Connect:Direct for UNIX SNODE has the directory /financial/accounts. The following wildcard copy command copies the files called customer1, customer2, and customer3 from the PNODE to the /financial/accounts directory on the SNODE. The source file names and the destination file names are identical.

```
WILDCOPY  COPY
                                FROM  (FILE=C:\financial\accounts\customer?)
                                TO      (FILE=/financial/accounts/
                                DISP=RPL)
```

When specifying a path and filename on Microsoft Windows, you can use the standard Microsoft Windows format when sending from C: or D: as show in the FROM parameter above. If you send from any other drive, you must use the UNC format such as \servername\financial\accounts\customer?.

You must include the ending forward slash (/) for the destination directory.

The following wildcard copy step copies customer1, customer2, customer3, supplier1, and supplier2 into the /financial/accounts directory on the SNODE. The source file names and the destination file names are identical.

```
WILDCOPY  COPY
                                FROM  (FILE=C:\financial\accounts\*)
                                TO      (FILE=/financial/accounts/
                                DISP=RPL)
```

Wildcard Copy from Microsoft Windows to Microsoft Windows

To copy send to a Connect:Direct for Microsoft Windows node, you must include an ending backslash (\) in the **TO FILE=** parameter. Following is an example:

```
WILDCOPY  COPY
                                FROM  (FILE=C:\financial\accounts\customer?)
                                TO      (FILE=\\server1\financial\accounts\
                                DISP=RPL)
```


You can use the standard Microsoft Windows format when sending from C: or D: as show in the FROM parameter above. If you send from any other drive, you must use the UNC format, as shown in the TO parameter above.

You must include the ending backslash (\) for the destination directory.

Wildcard Copy from Microsoft Windows to a z/OS Node

To copy send to sequential files on a Connect:Direct for z/OS node, you must include an ending period (.) in the **TO FILE=** parameter. Following is an example:

```
WILDCOPY COPY
  FROM (FILE=C:\financial\accounts\*)
  TO   (FILE=FINANCIAL.ACCOUNTS.
        DISP=RPL)
```

To copy send to a PDS on a Connect:Direct for z/OS node, you must use an asterisk (*) for the PDS member name. Following is an example:

```
WILDCOPY COPY
  FROM (FILE=C:\financial\records\*)
  TO   (FILE=FINANCIAL.RECORDS(*)
        DISP=RPL)
```

When specifying a path and filename on Microsoft Windows, you can use the standard Microsoft Windows format when sending from C: or D: as show above. If you send from any other drive, you must use the UNC format such as \servername\financial\accounts\customer?.

Wildcard Copy from Microsoft Windows to a Node with Download Restrictions

To copy send to a IBM 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=C:\financial\records\*)
  TO   (FILE=*
        DISP=RPL)
```

Copy a File from Microsoft Windows to z/OS

This Process copies a file from a Microsoft Windows node to a z/OS 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=0S390.v4200 /*$0S390$*/
              hold=no
              class=1
              prty=10
              execprty=10
              retain=no
              pnodeid=(user01,pw01)
              snodeid=(user01,pw01)
step01     copy    from  (
              file=\srv-one\c_drive      emp\ntfile.txt
              pnode/*$WindowsNT$*/
              sysopts="datatype(text) strip.blanks(no)
              xlate(yes)")
              ckpt=1k
              compress extended
              to      (
              snode /*$0S390$*/
              file=user01.newfile2
              disp=(rpl,catlg))
```

Use Symbolics In a Microsoft Windows Copy

This Process (named copyseq) copies a file from Microsoft Windows to z/OS. It uses symbolics to specify the file names to be copied. It also specified accounting data 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)
```

This submit command submits the copyseq Process and specifies the file names to use in the copy.

```
submit     file=copyseq
           &homefile="c:\mydir\myfile.txt"
           &dalfile="user01.newfile"
```

CODEPAGE Conversion During a File Copy (Microsoft Windows to z/OS)

The following example shows how to specify the CODEPAGE SYSOPTS parameter when copying a file from an ANSI Latin Connect:Direct for Microsoft Windows system (UTF-8 codepage) to a U.S. or Canadian Connect:Direct for z/OS system (IBM-1047 codepage).

```
CODEPGNT  PROCESS    SNODE=SNBEACH.OS390
                    HOLD=NO
CODEPGCP  COPY    FROM (PNODE
                    DSN='C:\SRN_TESTDATA\MIXED.TXT'
                    DISP=SHR
                    SYSOPTS="CODEPAGE=(1252,UTF-8)")
                    CKPT=5M
                    TO   (SNODE
                    DSN=SNBEAC1.DATAFILE.MIXED
                    DISP=RPL
                    SYSOPTS="CODEPAGE(UTF-8,IBM-1047)")
```

Copy PDS Members from z/OS to OpenVMS

This Process copies a PDS and all of its members from z/OS to a text library on OpenVMS. Note that the string of SYSOPTS parameters is enclosed in double quotation marks.

```
PDSCPY1  PROCESS    SNODE=CD.OS390.NODE
STEP01   COPY    FROM (DSN=SMITH.INPUT83.MEMBERS SNODE)      -
                    TO   (DSN=DUA0:[RJONES.CDTEST]PDSTEST.TLB -
                    DISP=OLD -
                    DCB=(DSORG=PO) -
                    SYSOPTS="LIBRARY='TEXT' REPLACE" PNODE)
```

Copy PDS Members from z/OS to OpenVMS

This Process copies a PDS and all of its members from z/OS to a text library on OpenVMS. Note that the string of SYSOPTS parameters is enclosed in double quotation marks.

```
PDSCPY1  PROCESS    SNODE=CD.OS390.NODE
STEP01   COPY    FROM (DSN=SMITH.INPUT83.MEMBERS SNODE)      -
                    TO   (DSN=DUA0:[RJONES.CDTEST]PDSTEST.TLB -
                    DISP=OLD -
                    DCB=(DSORG=PO) -
                    SYSOPTS="LIBRARY='TEXT' REPLACE" PNODE)
```

Copy a File from Disk to Tape (OpenVMS)

The following example copies an OpenVMS file from disk to tape. Specifying the /OVERRIDE qualifier causes the name of the tape volume to be ignored. The /OVERRIDE qualifier can be added either to the MOUNT command or to the tape label parameter. The example shows the qualifier added to the parameter.

```
T1      PROCESS      SNODE=SC.VMS.JOEUSER  SNODEID=(JOEUSER,USER_PASSWORD)
STEP01  COPY FROM    (DSN=DUXX:<DIRECTORY>TESTFILE.DAT)      -
        TO           (DSN=MUXX:TESTFILE.DAT)              -
        SYSOPTS="MOUNT='MUXX: TAPE /OVERRIDE=ID' "
```

Copy a File from Tape to Disk (OpenVMS)

The following example copies an OpenVMS file from tape to disk. Specifying the /OVERRIDE qualifier causes the name of the tape volume to be ignored.

```
T2      PROCESS      SNODE=SC.VMS.JOEUSER  SNODEID=(JOEUSER,USER_PASSWORD)
STEP01  COPY FROM    (DSN=MUXX:TESTFILE.DAT)              -
        TO           (DSN=DUXX:<DIRECTORY>TESTFILE.DAT)    -
        SYSOPTS="MOUNT='MUXX: TAPE /OVERRIDE=ID' "
```

Copy from z/OS to OpenVMS and Specifying a User-Defined Translation Table

This Process, submitted from the OpenVMS node, copies from a Connect:Direct for z/OS node to a Connect:Direct for OpenVMS node. It illustrates how a user can specify a user-defined translation table. Using the XLATE keyword in a Process overrides the default translation table; the IBM Connect:Direct system will extract a module from the OpenVMS text library CD_1.TBL.

```
RECV_VB PROCESS      SNODE=CD.OS390.NODE
STEP01  COPY FROM    (DSN=SMITH.CDTEST SNODE              -
                    DISP=SHR )                            -
        TO           (DSN=$DISK:[JONES.DATA]ACCT.TXT      -
                    SYSOPTS="NOBINARY                    -
                    XLATE='$DSK:[JS.TXT]CD_1.TBL{JS.DEF}' -
                    PROTECTION='S:RW,W:RWED'            -
                    LIBRARY='TEXT' STREAM" PNODE)
```

Copy a Single Entry from the OpenVMS Text Library to a z/OS Member

This Process sends a single entry from an OpenVMS text library to a member of a PDS on z/OS.

```
SINGENT1 PROCESS      SNODE=CD.OS390.NODE
STEP01  COPY FROM    (DSN=DUAO:[SMITH.CDTEST]PDSTEST.TLB PNODE -
                    SELECT=(BOOLEAN)                      -
                    SYSOPTS="LIBRARY='TEXT' "             -
                    DISP=OLD                              -
                    DCB=(DSORG=PO))                      -
        TO           (DSN=SMITH.MEMBERS(DATA)             -
                    DISP=RPL SNODE)
```

Copy All Entries from an OpenVMS Text Library to z/OS

This Process copies all the modules of a text library to respective members of a PDS on z/OS. Note the use of the asterisk (*) with the SELECT parameter.

```
ALLENT1  PROCESS      SNODE=CD.OS390.NODE
STEP01   COPY FROM   (DSN=DUA0:[JSMITH.CDTEST]PDSTEST.TLB PNODE -
                    SELECT=(*)                      -
                    SYSOPTS="LIBRARY='TEXT' "        -
                    DISP=OLD                          -
                    DCB=(DSORG=PO))                  -
                    TO   (DSN=JSMITH.MEMBERS         -
                    DISP=RPL SNODE)
```

Copy a Data Set from a z/OS Node to an Executable File on an OpenVMS Node

This Process, submitted from the OpenVMS node, copies the z/OS data set RSMITH.ACCTJAN to file specification DUC4:[ACCT.COM]JAN.EXE at the OpenVMS node. For the appropriate file attribute information, the SYSOPTS parameter TYPE=IMAGE must be specified. Also, BINARY must be specified as part of the SYSOPTS parameter so that EBCDIC to ASCII translation will not occur.

```
PROCESS1  PROCESS      SNODE=CD.OS390
STEP01   COPY FROM   (SNODEID=(RSMITH,ROGER)
                    TO   (DSN=RSMITH.ACCTJAN SNODE )
                    (DSN=DUC4:[ACCT.COM]JAN.EXE PNODE
                    SYSOPTS="TYPE='IMAGE' BINARY"
                    DISP=RPL)
```

Copy an Executable File from an OpenVMS Node to a z/OS Node

This Process, submitted from the OpenVMS node, copies the OpenVMS file JAN.EXE in directory [ACCT.COM] on device DUC4 to data set RSMITH.ACCTJAN at the z/OS node. BINARY must be specified as part of the SYSOPTS parameter so that ASCII to EBCDIC translation will not occur.

```
PROCESS1  PROCESS      SNODE=CD.OS390
STEP01   COPY FROM   (SNODEID=(RSMITH,ROGER)
                    TO   (DSN=DUC4:[ACCT.COM]JAN.EXE PNODE
                    SYSOPTS="BINARY")
                    (DSN=RSMITH.ACCTJAN
                    DISP=RPL SNODE)
```

To submit this Process from the z/OS node (the PNODE is a z/OS node), the following syntax changes must be made:

- Enclose the OpenVMS file specification between single or double quotation marks to allow special characters to be passed to the OpenVMS node.
- Change the brackets ([]) to less than and greater than signs (< >).

The modified Process is as follows:

```
PROCESS1  PROCESS      SNODE=CD.VMS
STEP01   COPY FROM   (SNODEID=(RSMITH,ROGER)
                    TO   (DSN='DUC4:<ACCT.COM>JAN.EXE' SNODE
                    SYSOPTS="BINARY")
                    (DSN=RSMITH.ACCTJAN
                    DISP=RPL PNODE)
```

Copy a Text File from OpenVMS to Microsoft Windows and Back to OpenVMS

In this example, a text file (TIME_TEST.TXT) is being copied from the OpenVMS platform to Microsoft Windows and being renamed VMS_TEST.TXT. In the next COPY statement, the VMS_TEST.TXT file is copied from the Microsoft Windows platform back to OpenVMS and being renamed WIN_TIME_TEST.TXT.

```

WINTXT  PROCESS      SNODE=W2S.4200.CDWOPS7  snodeid=(userid,pwd)
VMS2WIN COPY FROM    (DSN=DISK$AXP:[NDM_3400.INPUT]TIME_TEST.TXT DISP=SHR pnode)  -
          TO          (DSN="C:\output\vms_test.txt" DISP=RPL snode)
WIN2VMS COPY FROM    (DSN="C:\output\vms_test.txt" DISP=SHR snode)  -
          TO          (DSN=DISK$AXP:[NDM_3400.OUTPUT]WIN_TIME_TEST.TXT DISP=RPL pnode)

```

Copy between the z/OS and OpenVMS Platforms

In this example, the PNODE resides on an OpenVMS system while the SNODE is on a z/OS system. This is a two-step Process where in step 1 a file (CSDQA1.TESTFILE.BENCH.M100) is being copied from the SNODE to a file on the PNODE and being renamed test100.dat. In step 2 the OpenVMS file, test100.dat, is copied to a new z/OS dataset called CSDQA2.TEST.MB100. The SUB1 statement submits a Process that resides on the z/OS system.

```

vms2mvsr process      snode=Q1B.ZOS.V4600 SNODEID=(JWARD1,APR1951)  -
                      pnodeid=(jward1,ward1949) pnode=q3a.alpha.v3400
*
* test copy, ckpt, comp and sub job
Step1  copy from      (file=CSDQA1.TESTFILE.BENCH.M100 DISP=(OLD) snode)  -
          ckpt = 10M compress
          to          ( pnode file=disk$data:[qaalpha.q3a]test100.dat DISP=(RPL)  -
                      sysopts="nobinary")
Step2  copy from      (file=disk$data:[qaalpha.q3a]test100.dat disp=(SHR))  -
          ckpt = 20M compress
          to          (file=CSDQA2.TEST.MB100 snode
                      disp=(RPL,CATLG,DELETE)
                      dcb=(RECFM=FB,LRECL=80,BLKSIZE=6160)
                      space=(cyl,(20,10),rlse))
SUB1   SUBMIT        DSN='CSDQA1.MANUAL.PROCESS(MVS2VMSR)'  -
                      SUBNODE=SNODE CASE=YES

```

Copy and Compare Files on OpenVMS

In this example Processes, two files are being copied and then compared. If the two files check out to be the same, the user is notified that the compare was good; if not, the user is notified that the compare failed.

```

BENM1SEC PROCESS      SNODE=SHEMPHILL-DT SNODEID=(SHEMPHILL,cayanne)
itP2S  COPY FROM      (DSN=DISK$DATA:[QADATA.INPUT]BENCHM5.dat pnode)  -
          compress
          ckpt=1M
          TO          (file=c:\BENCHM1SEC.TXT snode)
itS2P  COPY FROM      (file=c:\BENCHM1SEC.TXT snode)  -
          TO          (DSN=DISK$DATA:[QADATA.TEMP]BENCHM5.dat pnode)  -
          compress
          ckpt=1M
CMPR   RUN TASK      (pgm=vms) pnode  -
          sysopts = "output='DISK$DATA:[qadata.logs]diff.log'
          cmd = 'DIFF DISK$DATA:[qadata.input]BENCHM5.dat
          DISK$DATA:[qadata.temp]BENCHM5.dat'"
RCZERO IF (CMPR = 0) then
          run task (pgm=vms) pnode sysopts = "
          cmd = 'reply/user=SHEMP1 BENCHM1.dat-GOOD'"
          else
          run task (pgm=vms) pnode sysopts = "
          cmd = 'reply/user=SHEMP1 BENCHM1.dat-COMPARE_FAILED'"
EIF

```

Copy HFS and Text Files Back and Forth Between z/OS and OpenVMS

In this example, the PNODE is OpenVMS while the SNODE is z/OS. For each node, userids and passwords have been included. Step up1 copies a HFS text file from the z/OS USS to a text file on the OpenVMS system called hfsout.txt. The OpenVMS directory is disk\$data:[qaalalpha.q3a]. Step back1 is just the opposite but it names the HFS file unix2.txt. The sysopts="permission" gives read write authority to HFS files. The sysopts="binary" indicates the data is not to be converted from ASCII to EBCDIC.]

```
VMS2HFS  PROCESS      SNODE=Q1B.ZOS.V4600  SNODEID=(JWARD1,MAY1975)  -
up1      COPY          PNODE=Q3A.ALPHA.V3400  PNODEID=(jward1,ward1949)  -
          TO          ( DSN=DISK$DATA:[qaalalpha.q3a]hfsout.txt disp=rpl  -
          SYSOPTS="binary" )  -
          FROM        (file='/u/jward1/unix.txt' disp=shr SNODE  -
          SYSOPTS="PERMISS=777")  -
back1    COPY  TO      (file='/u/jward1/unix2.txt' disp=rpl SNODE  -
          SYSOPTS="PERMISS=777")  -
          FROM        ( DSN=DISK$DATA:[qaalalpha.q3a]hfsout.txt  -
          DISP=SHR  -
          SYSOPTS="binary")
```

Copy a Text File from OpenVMS to UNIX and Back to OpenVMS

In this example, a text file (TIME_TEST.TXT) is being copied from the OpenVMS platform to UNIX. In the next COPY statement, the TIME_TEST.TXT file is copied from the UNIX platform back to OpenVMS and being renamed UX_TIME_TEST.TXT.

```
UXTXT  PROCESS      SNODE=Qasol103700  Snodeid(qatest,qatest)
VMS2UX  COPY  FROM  ( DSN=DISK$AXP:[NDM_3400.INPUT]TIME_TEST.TXT DISP=SHR pnode )  -
          TO        ( DSN="/tmp/TIME_TEST.TXT" DISP=RPL snode )  -
UX2VMS  COPY  FROM  ( DSN="/tmp/TIME_TEST.TXT" DISP=SHR snode )  -
          TO        ( DSN=DISK$AXP:[NDM_3400.OUTPUT]UX_TIME_TEST.TXT DISP=RPL pnode )
```

Copy a FB Text File to a z/OS PDS File and Pull Back to OpenVMS

In this example, a text file with fixed-length records on an OpenVMS node is being copied to a z/OS PDS file and then copied from the z/OS secondary node and pulled back to the OpenVMS primary node as a fixed-block text file.

```
ZOSPDS1  PROCESS      SNODE=Q1B.ZOS.V4600  -
STEP01   COPY  FROM  ( DSN=DISK$AXP:[NDM_3400.INPUT]A80_80FB.dat DISP=SHR PNODE)  -
          TO        ( DSN=CSDQA1.0.TESTFILE.PDS(a8080) DISP=RPL SNODE)  -
STEP01   COPY  FROM  ( DSN=CSDQA1.0.TESTFILE.PDS(a8080) DISP=SHR SNODE)  -
          TO        ( DSN=DISK$AXP:[NDM_3400.OUTPUT]zos_80.dat  -
          DCB=(DSORG=PS,RECFM=FB)  -
          DISP=RPL  PNODE )
```

Copy a Binary File from OpenVMS to Microsoft Windows and Back

In this example, a binary file (TEST.EXE) is being copied from the OpenVMS platform to Microsoft Windows. In the next COPY statement, the TEST.TXT file is copied from the Microsoft Windows platform back to OpenVMS.

```
WINBIN   PROCESS      SNODE=W2S.4200.CDWOPS8  snodeid=(userid,pswd)
VMS2WIN  COPY  FROM  ( DSN=DISK$AXP:[NDM_3400.INPUT]TEST.EXE DISP=SHR PNODE )  -
          TO        ( DSN="C:\output  est.exe"  -
          SYSOPTS="datatype(binary)" DISP=RPL snode )  -
WIN2VMS  COPY  FROM  ( DSN="C:\output est.exe"  -
          SYSOPTS="datatype(binary)" DISP=SHR snode )  -
          TO        ( DSN=DISK$AXP:[NDM_3400.OUTPUT]test.exe  -
          DCB=(DSORG=PS,RECFM=FB,LRECL=512) DISP=RPL  PNODE )
```

Copy a Binary File from OpenVMS to UNIX and Back

In this example, a binary file (TEST.EXE) is being copied from the OpenVMS platform to UNIX. In the next COPY statement, the TEST.TXT file is copied from the UNIX platform back to OpenVMS (ux_test.exe).

```
UXBIN  PROCESS      SNODE=qasol103700 Snodeid(userid,pwd)
VMS2UX COPY FROM    (DSN=DISK$AXP:[NDM_3400.INPUT]TEST.EXE DISP=SHR PNODE) -
      TO            (DSN="/tmp/test.exe" SYSOPTS=":datatype=binary:" DISP=RPL SNODE)
UX2VMS COPY FROM    (DSN="/tmp/test.exe" SYSOPTS=":datatype=binary:" DISP=SHR SNODE) -
      TO            (DSN=DISK$AXP:[NDM_3400.OUTPUT]ux_test.exe
                   DCB=(DSORG=PS,RECFM=FB,LRECL=512) DISP=RPL PNODE) -
```

Copy an OpenVMS Key-Sequenced File to an HP NonStop Node

This Process, submitted from the OpenVMS node, will copy a key-sequenced file to the Connect:Direct for HP NonStop node. When copying files from Connect:Direct for OpenVMS to Connect:Direct for HP NonStop nodes, include SET XLATE ON in the SYSOPTS parameter of the TO clause of the COPY statement. Because the Connect:Direct for OpenVMS system translates ASCII characters to EBCDIC, the XLATE subparameter will turn on the text conversion utility and translate from EBCDIC to ASCII. The FASTLOAD option is used to reduce disk I/O overhead.

```
VAXSND PROCESS      PNODE=CD.VMS -
STEP1  COPY FROM    SNODE=CD.HPNONSTOP SNODEID=(GRP.USR,PASWRD) -
      TO            (DSN=[USER.DATA]KSDS1.DAT) -
                   (DSN=$B.DATA.KSDS -
                   DCB=(DSORG=K -
                   BLKSIZE=4096 -
                   LRECL=100 -
                   KEYLEN=10) -
                   DISP=RPL -
                   SYSOPTS="SET XLATE ON FAST.LOAD Y")
```

Copy an OpenVMS File from Disk to Tape

The following example copies an OpenVMS file from disk to tape. Specifying the /OVERRIDE qualifier causes the name of the tape volume to be ignored. The /OVERRIDE qualifier can be added either to the MOUNT command or to the tape label parameter. The example shows the qualifier added to the parameter.

```
T1     PROCESS      SNODE=SC.VMS.JOEUSER SNODEID=(JOEUSER,USER_PASSWORD)
STEP01 COPY FROM    (FILE=DUXX:[DIRECTORY]TESTFILE.DAT) -
      TO            (FILE=MUXX:TESTFILE.DAT -
                   SYSOPTS="MOUNT='MUXX: TAPE /OVERRIDE=ID' ")
```

Copy an OpenVMS File from Tape to Disk

The following example copies an OpenVMS file from tape to disk. Specifying the /OVERRIDE qualifier causes the name of the tape volume to be ignored.

```
T2     PROCESS      SNODE=SC.VMS.JOEUSER SNODEID=(JOEUSER,USER_PASSWORD)
STEP01 COPY FROM    (FILE=MUXX:TESTFILE.DAT -
                   SYSOPTS="MOUNT='MUXX: TAPE /OVERRIDE=ID' ") -
      TO            (FILE=DUXX:[DIRECTORY]TESTFILE.DAT)
```

Use Symbolics in an OpenVMS COPY Statement

This example shows the basic use of symbolics in a Process. Both the FROM and TO files, as well as the file disposition, are resolved at Process submission.

```
SYMBOL1 PROCESS      SNODE=REMOTE_NODE
STEP01  COPY FROM    (FILE=&FROM PNODE)  -
        TO           (FILE=&TO  DISP=&DISP)
```

The Process, SYMBOL1, can be submitted with the following command issued in DCL command format:

```
$ NDMUI SUBMIT SYMBOL1      -
/SYMBOLICS=("FROM=VMS_FILENAME.TYPE", -
           "TO=OS390.DATASET.NAME", -
           "DISP=RPL")
```

Copy an OpenVMS Sequential File to a Text Library

This example shows the format for copying a sequential file to a text library.

```
COPY01  PROCESS      SNODE=CD.VMS.DATAMOVER
DO_A    COPY FROM    (FILE=VMSFILE)      -
        TO           (FILE=VMSLIBRARY.TLB(VMSFILE) -
                     SYSOPTS="LIBRARY='TEXT' REPLACE" -
                     DISP=RPL           -
                     DCB=(DSORG=PO))
```

Copy a File from OpenVMS to z/OS and Back to OpenVMS

These example COPY statements (STEP01 and STEP02) copy an executable file from a Connect:Direct for OpenVMS node to a Connect:Direct for z/OS node and then back to the Connect:Direct for OpenVMS node. Because BINARY is specified as part of the SYSOPTS parameter, ASCII-to-EBCDIC translation does not occur. Enclose the SYSOPTS string 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)
```


Copy Files Between UNIX and z/OS

This Process copies a file from UNIX to z/OS. The Process was initiated from the UNIX node. The **ckpt** parameter specifies that no checkpoints will be taken. The **ckpt** parameter is generally coded between the FROM and TO clauses of the COPY statement.

```
OS390xx process      snode=OS390.node
step01  copy from    (file=file1
                    pnode)
                    ckpt=no
                    to  (file=file2
                    dcb=(dsorg=PS,
                    recfm=FB,
                    lrecl=80,
                    blksize=2400)
                    space=(TRK,(1,,))
                    snode
                    disp=rpl)
pend
```

This Process, submitted from the z/OS node, copies a text file from UNIX to z/OS. The DSN and SYSOPTS strings for the UNIX system must be in the proper case for UNIX and enclosed in double quotation marks.

```
PROC2   PROCESS      SNODE=UNIX.NODE
STEP01  COPY FROM    (DSN="/company/payroll/monthly/jan"
                    SYSOPTS=":datatype=text:"
                    SNODE)
                    TO  (DSN=OS390.PAYROLL.MONTHLY.JANUARY
                    DISP=(NEW,CATLG)
                    SPACE=(23040,(2,1))
                    DCB=(RECFM=U,LRECL=0,BLKSIZE=23040,DSORG=PS)
                    PNODE)
```

Copy Files Between UNIX and z/OS

This Process copies a file from UNIX to z/OS. The Process was initiated from the UNIX node. The **ckpt** parameter specifies that no checkpoints will be taken. The **ckpt** parameter is generally coded between the FROM and TO clauses of the COPY statement.

```
OS390xx process      snode=OS390.node
step01  copy from    (file=file1
                    pnode)
                    ckpt=no
                    to  (file=file2
                    dcb=(dsorg=PS,
                    recfm=FB,
                    lrecl=80,
                    blksize=2400)
                    space=(TRK,(1,,))
                    snode
                    disp=rpl)
pend
```

This Process, submitted from the z/OS node, copies a text file from UNIX to z/OS. The DSN and SYSOPTS strings for the UNIX system must be in the proper case for UNIX and enclosed in double quotation marks.

```
PROC2   PROCESS      SNODE=UNIX.NODE
STEP01  COPY FROM    (DSN="/company/payroll/monthly/jan"
                    SYSOPTS=":datatype=text:"
                    SNODE)
                    TO  (DSN=OS390.PAYROLL.MONTHLY.JANUARY
                    DISP=(NEW,CATLG)
                    SPACE=(23040,(2,1))
                    DCB=(RECFM=U,LRECL=0,BLKSIZE=23040,DSORG=PS)
                    PNODE)
```

Copy Files and Using sysopts (UNIX to UNIX)

This Process copies a file between two UNIX nodes. The **sysopts** parameter specifies to remove trailing blank characters from a line of text before writing it to a text file. The **sysopts** subparameters are a series of field names and values, each of which is delimited by a colon and enclosed in double quotation marks.

```
strip process      snode=unix.node
step01 copy from  (file=blank.dat
                  sysopts=":datatype=text:"
                  snode)
                to  (file=blank_no
                  sysopts=":datatype=text:strip.blanks=yes:"
                  pnode)
pend
```

Copy Files and Use the Checkpointing Feature (UNIX to UNIX)

This Process copies a file between two UNIX nodes. The **ckpt** parameter specifies that checkpoints will be taken at 128K intervals. If the COPY operation is interrupted, the IBM Connect:Direct system will restart that COPY step at the last checkpoint. Code the **ckpt** parameter between the FROM and TO clauses of the COPY statement.

```
ckpt01 process      snode=unix.node
step01 copy from  (file=file1
                  snode)
                  ckpt=128k
                to  (file=file2
                  disp=new
                  pnode)
pend
```

Copy Files and Use the Compression Feature (UNIX to UNIX)

This Process shows the syntax of the **compress** parameter. The **compress** parameter specifies that 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. Code the **compress** parameter between the FROM and TO clauses of the COPY statement.

Compression activities for each step are as follows:

- Step01 specifies use of hex **20**, the default, as a compression character.
- Step02 specifies use of character **1** as a compression character.
- Step03 specifies use of hex **11** as a compression character.
- Step04 specifies use of the extended compression method. CMP specifies the compression level. WIN specifies the window size. MEM specifies the memory level.

Use this Process when you copy files from UNIX to UNIX using the compress parameter.

```

comp    process    snode=unix.node
step01  copy    from  (file=text2 snode)
                    compress
                    to    (file=file3 pnode)
step02  copy    from  (file=text2 snode)
                    compress primechar=c'1'
                    to    (file=file3 pnode)
step03  copy    from  (file=text2 snode)
                    compress primechar=x'11'
                    to    (file=file3 pnode)
step04  copy    from  (file=text2 snode)
                    compress extended [(CMP=1
                    WIN=9
                    MEM=1
                    )
                    ]
pend    to    (file=file3 pnode)

```

Archive Files Using the Connect:Direct for UNIX Pipe I/O Function

This Process changes the **pnode** directory to the **se** subdirectory, archives all the *.c files in the **se** subdirectory using the **tar** command, and then transfers the archive to the **snode**. No checkpointing occurs when **pipe=yes** is specified.

```

pipe_ex1 process    snode=testcdu
step01   copy    from  (file="cd se; tar -cf - *.c"
                    pnode sysopts=":pipe=yes:")
                    to    (file=unix.se.tar snode disp = rpl)
pend

```

Restore Files Using the Connect:Direct for UNIX Pipe I/O Function

This Process copies a **tar** archive from the **snode** and extracts files from the archive. No checkpointing occurs when **pipe=yes** is specified.

```

pipe_ex2 process    snode=testcdu
step01   copy    from  (file=unix.se.tar snode)
                    to    (file="cd se; tar -xf -" pnode sysopts=":pipe=yes:")
pend

```

Archive and Restore Files in a Single Step Using the Connect:Direct for UNIX Pipe I/O Function

This Process changes the **pnode** directory to the **se** subdirectory, archives all the *.c files in the **se** subdirectory using the **tar** command, then transfers the archive to the **snode**. At the **snode**, this Process changes the directory to the **testdir** subdirectory and extracts the *.c files from the archive using the **tar -xf** command. No checkpointing occurs when **pipe=yes** is specified.

```

pipe_ex3 process    snode=testcdu
step01   copy    from  (file="cd se; tar -cf - *.c"
                    pnode sysopts=":pipe=yes:")
                    to    (file="cd testdir; tar -xf -"
                    snode sysopts=":pipe=yes:")
pend

```

Copy Files from UNIX to a Member on i5/OS

This Process copies an ASCII file from UNIX to a member on the i5/OS.

```
* COPY TO MEMBER *
copy01 process      snode=os400
                   snodeid=(userid,password)
step01 copy from   (file=/cd/file1
                   pnode
                   sysopts=":datatype=text:xlate=yes:")
                   to   (file="LIB/FILENAME(MBR_NAME)"
                   sysopts="TYPE( MBR )"
                   disp=rpl)
pend
```

Copy Files from UNIX to a Spool File on i5/OS

This Process copies an ASCII file from UNIX to a spool file on the i5/OS.

See the *IBM Connect:Direct for i5/OS User's Guide* for the spool file parameters.

```
* COPY TO SPOOL FILE *
copy01 process      snode=os400
                   snodeid=(userid,password)
step01 copy from   (file=/cd/file1
                   pnode
                   sysopts=":datatype=text:xlate=yes:")
                   to   (file=FILE2
                   snode
                   sysopts="TYPE( SPLF ) PRTQLTY( *NLQ )"
                   disp=rpl)
pend
```

Copy Save Files from i5/OS to UNIX

This Process copies a save file from i5/OS to UNIX.

```
* COPY SPECIFYING DCB INFORMATION *
copy01 process      snode=os400
                   snodeid=(userid,password)
step01A copy       from   (file="URGRSSSV1/SAVEFILE1"
                   snode
                   sysopts="TYPE(OBJ)")
                   compress
                   to   (file=/cd/usavefile1
                   sysopts=":datatype=binary:permiss=774:"
                   pnode
                   disp=new)
pend
```

Copy Save Files from UNIX to i5/OS

This Process copies a save file from UNIX to i5/OS. Set **datatype=binary** for save files. Specify DCB information to copy a save file to i5/OS.

```
* COPY SPECIFYING DCB INFORMATION *
copy02  process      snode=os400
                               snodeid=(userid,passwd)
step02B  copy
        from  (file=/cd/usavefile1
                sysopts=":datatype=binary:permiss=774:"
                pnode)
        to    (file="URGRSSSV1/SAVEFILE1"
                snode
                sysopts="TYPE(OBJ) MAXRCDS(*NOMAX)"
                disp=new)
pend
```

Copy Executables from UNIX to i5/OS

This Process copies an executable from UNIX to i5/OS. Specify FILETYPE(*DATA) in the **sysopts** parameter.

```
* COPY UNIX EXECUTABLE TO i5/OS *
copy01  process      snode=os400
                               snodeid=(userid,passwd)
step01  copy  from  (file=/cd/xdt3
                sysopts=":datatype=binary:permiss=777:"
                pnode)
        to    (snode
                file="CD/BINARY(UDESKTOP)"
                sysopts="TYPE(MBR) FILETYPE(*DATA)"
                disp=new)
pend
```

Copy a File from UNIX to Microsoft Windows

This Process copies a binary file from a UNIX node to a Microsoft Windows node. A TCP/IP address is specified instead of the IBM Connect:Direct node name for the SNODE.

```
ux2nt   process      snode=111.11.11.111
                               hold=no
                               retain=no
copy1   copy  from  (file=/usr/data/out/invoi01.dat
                pnode
                sysopts=":datatype=binary:")
        to    (file=d:\users\data\in\invoi01.dat
                snode
                sysopts="datatype(binary)" )
pend
```

The following Process is a variation on the previous example. In this example, the file names are defined as symbolic variables in the COPY statement (&file1 and &file2) and are resolved at the time the Process is submitted.

```
proc1   process      snode=111.11.11.111
                               &file1="/usr/data/out/invoi01.dat"
                               &file2="d:\users\data\in\invoi01.dat"
copy1   copy  from  (file=&file1
                pnode
                sysopts=":datatype=binary:")
        to    (file=&file2
                sysopts="datatype(binary)"
                snode)
pend
```

Copy a File from a UNIX Node to a z/OS Node

In this example, the Process copies a binary file from a local UNIX node to a z/OS 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
```

Wildcard Copies from UNIX to Microsoft Windows

In the following example, a Connect:Direct for UNIX PNODE directory /financial/accounts contains the files customer1, customer2, customer3, supplier1, and supplier2. A Connect:Direct for Microsoft Windows SNODE has the directory C:/financial/accounts. The following wildcard copy command copies the files called customer1, customer2, and customer3 from the PNODE to the C:/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=C:\financial\accounts\
            DISP=RPL)
```

You must include the ending backslash (\) for the destination directory.

The following wildcard copy step copies customer1, customer2, customer3, supplier1, and supplier2 into the C:\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)
```

Wildcard Copy from UNIX to UNIX

To copy send to a Connect:Direct for 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 from UNIX to a z/OS Node

To copy send to sequential files on a Connect:Direct for z/OS 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 for z/OS 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 from UNIX to a Node with Download Restrictions

To copy send to a IBM 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)
```

Copy a VSAM File from VM to an Entry-Sequenced HP NonStop File

The following Process copies a VM VSAM file to an entry-sequenced HP NonStop file. The entry-sequenced file with extents of 100 pages each is created as indicated by the SYSOPTS parameter. Note that the VM file name is not enclosed in single quotation marks. If the VM file name is not placed between quotation marks, the IBM Connect:Direct system assumes the file is a VSAM file.

```
HPNONSTP  PROCESS      PNODE=CD.VM.DALLAS HOLD=YES      -
          SNODE=BOSTON.01 NOTIFY=%USER    -
          SNODEID=(127.200,JONES)         -
SEND01    COPY FROM    (DSN=ABC.RPTFILE    -
          LINK=(IVVB6,RIVVB6,RR,200))    -
          TO      (DSN=$C.JONES.VSAME SNODE -
          SYSOPTS="\ 'SET EXT(100 100)'\ || -
          \ 'SET TYPE E'\                -
          DISP=(RPL))
```

Copy a VSAM File from VM to an Entry-Sequenced HP NonStop File

The following Process copies a VM VSAM file to an entry-sequenced HP NonStop file. The entry-sequenced file with extents of 100 pages each is created as indicated by the SYSOPTS parameter. Note

that the VM file name is not enclosed in single quotation marks. If the VM file name is not placed between quotation marks, the IBM Connect:Direct system assumes the file is a VSAM file.

```

HPNONSTP  PROCESS      PNODE=CD.VM.DALLAS HOLD=YES      -
                                SNODE=BOSTON.01 NOTIFY=%USER      -
                                SNODEID=(127.200,JONES)          -
SEND01    COPY FROM    (DSN=ABC.RPTFILE                               -
                                LINK=(IVVB6,RIVVB6,RR,200))      -
                                TO      (DSN=$C.JONES.VSAME SNODE   -
                                SYSOPTS=\ "'SET EXT(100 100)'\ \ || -
                                \ 'SET TYPE E'" \                 -
                                DISP=(RPL))                       -

```

Copy a VSAM File from VM to a Key-Sequenced HP NonStop File

The following Process copies a VM VSAM file to a key-sequenced HP NonStop file. The key-sequenced file with extents of 100 pages each is created as indicated by the SYSOPTS parameter. Because the VM file name is not placed between quotation marks, the IBM Connect:Direct system assumes the file is a VSAM file.

```

HPNNSTP6  PROCESS      PNODE=CD.VM.DALLAS                               -
                                SNODE=BOSTON.01 NOTIFY=%USER      -
                                SNODEID=(127.210,SMITH)           -
SEND01    COPY FROM    (DSN=ABC.TST                               -
                                LINK=(IVVB6,RIVVB6,RR,200))      -
                                TO      (DSN=$C.ABC.VSAMERR SNODE  -
                                SYSOPTS=\ "'SET EXT(100 100)'\ \ || -
                                \ 'SET KEYLEN 8'\ \ ||           -
                                \ 'SET REC 880'\ \ ||           -
                                \ 'SET TYPE K'" \                 -
                                DISP=(RPL))                       -

```

Copy a VM File to VM Spool

This Process copies a file to the VM reader of user RJONES. Because a copy to VM spool does not involve writing to disk, you do not need to specify link information.

```

VM2RDR2   PROCESS      SNODE=CD.VM.JSMITH NOTIFY=RJONES          -
STEP01    COPY FROM    (DSN='JIM SCRIPT'                          -
                                LINK=(RRT,READPW,RR,191))          -
                                TO      (DSN='!SPOOL RJONES TEST DATA' -
                                DCB=(DSORG=PS,RECFM=F,LRECL=80,BLKSIZE=80)) -

```

Copy an Entire VM Minidisk

This example shows the COPY statement of a Process that copies an entire minidisk to another minidisk (301).

```

STEP01    COPY FROM    (GROUP='* *'                               -
                                LINK=(MDSKI,RMSKI,RR,199)          -
                                DISP=SHR)                          -
                                TO      (GROUP='%1% %2%'           -
                                LINK=(N4100,WN14100,W,301)        -
                                DISP=RPL)                          -

```


Copy from VM Disk to Tape

This Process copies a VM disk file from the 191 disk of IVVB8 to tape.

TAPE	PROCESS	SNODE=CD.VM.NODE	-
		NOTIFY=CDA8	
STEP01	COPY FROM	(DSN='TEST FILE'	-
		LINK=(IVVB8,RIVVB7,RR,191)	-
		DISP=SHR)	-
	TO	(DSN=TEST.EXPDT.ONE	-
		UNIT=TAPE	-
		LABEL=(1,SL,EXPDT=900967)	-
		SNODE	-
		DISP=RPL)	

VM to VM Group File Copy

This Process illustrates a VM group file copy, which copies source modules from one minidisk to a minidisk at another site. Notice that a source file name, as well as a group name, is specified on the FROM clause of the COPY statement. This causes the Connect:Direct for VM system to send members of the group beginning with that source file name instead of beginning with the first member of the group; therefore, members of a group are excluded from the transfer.

GROUP9	PROCESS	SNODE=CD.VM.NODE	
STEP01	COPY FROM	(DSN='ACF2C A*'	-
		GROUP='* A*'	-
		LINK=(IVVB7,RIVVB7,RR,191)	-
		DISP=SHR)	-
	TO	(GROUP='%1% A%2%'	-
		LINK=(IVVB6,WIVVB6,W,301)	-
		DISP=RPL)	

The parameter GROUP on the TO clause contains the special symbols %1% and %2%, which are used to build the destination name. Each symbol is replaced by characters from the name determined to be in that source group.

The source disk, CDA7 191, contains the following:

ACF2A ASSEMBLE	ARMMOD ASSEMBLE
ACF2B ASSEMBLE	DMCXRJ ASSEMBLE
ACF2C ASSEMBLE	DMDPTR ASSEMBLE
ALOEXIT ASSEMBLE	DMFPTR ASSEMBLE
ASMSAMP ASSEMBLE	DMGFTR ASSEMBLE
ASMTASK ASSEMBLE	DMMF ASSEMBLE

After the transfer completes, the CDA6 300 disk contains:

CF2C ASSEMBLE	DMCXRJ ASSEMBLE
ALOEXIT ASSEMBLE	DMDPTR ASSEMBLE
ASMSAMP ASSEMBLE	DMFPTR ASSEMBLE
ASMTASK ASSEMBLE	DMGFTR ASSEMBLE
ARMMOD ASSEMBLE	DMMF ASSEMBLE

ACF2A and ACF2B are excluded from the COPY because the DSN parameter indicated that the group file copy should start with the ACF2C A* file.

Copy VM files to a Shared File System (SFS)

This multi-step Process copies several different types of VM files to a SFS. In each step, if the file exists, IBM Connect:Direct replaces it. If the file does not exist, the IBM Connect:Direct system creates it as indicated by the DISP=RPL parameter. All of the files (input and output) are fixed length 80 byte records. Each step performs the following task:

- STEP1 copies a CMS file in a SFS to another SFS.
- STEP2 copies a CMS file from a Minidisk to a SFS.
- STEP3 copies a VSAM RRDS to a sequential file in a SFS.
- STEP4 copies a VSAM KSDS to a sequential file in a SFS.
- STEP5 copies a VSAM ESDS to a sequential file in a SFS.

```

SFSPROC  PROCESS
        &PROCESS=SFSPROC
        &CKPT=0K
        &COMPRESS=COMPRESS
        &EXT=,
        &CUU1=0199
        &CUU2=0195
        &DIR1='MYSFS:USER01.MYSFS'
        &DIR2='COSFS:USER02.COSFS'
        &INUSER=USER01
        &INUSERP=RPASS
        SNODEID=(USERID,PASSWD)
        &SNODE=CD.VM.NODE1
        SNODE=&SNODE
STEP1    COPY FROM (PNODE
                  SFSDIR("&DIR1")
                  DSN='MYINPUT FILE'
                  DISP=SHR )
                  CKPT=&CKPT &COMPRESS &EXT
                TO (SNODE
                  SFSDIR("&DIR2")
                  DSN=\xd5 FILETEST\&PROCESS.1\xd5 \
                  DCB=(LRECL=80,RECFM=F)
                  DISP=RPL )
STEP2    COPY FROM (PNODE
                  LINK=(&INUSER,&INUSERP,RR,&CUU1)
                  DSN='MYINPUT FILE2'
                  DCB=(LRECL=80,RECFM=F,DSORG=PS)
                  DISP=SHR )
                  CKPT=&CKPT &COMPRESS &EXT
                TO (SNODE
                  SFSDIR("&DIR2")
                  DSN=\xd5 FILETEST\&PROCESS.2\xd5 \
                  DCB=(LRECL=80,RECFM=F)
                  DISP=RPL )
STEP3    COPY FROM (PNODE
                  LINK=(&INUSER,&INUSERP,RR,&CUU2)
                  DSN=MYHLQ.TESTFILE.VRRDS.FB80
                  DCB=(DSORG=VSAM)
                  DISP=SHR )
                  CKPT=&CKPT &COMPRESS &EXT
                TO (SNODE
                  SFSDIR("&DIR2")
                  DSN=\xd5 FILETEST\&PROCESS.3\xd5 \
                  DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
                  DISP=RPL )
STEP4    COPY FROM (PNODE
                  LINK=(&INUSER,&INUSERP,RR,&CUU2)
                  DSN=MYHLQ.TESTFILE.VKSDS.FB80
                  DCB=(DSORG=VSAM)
                  DISP=SHR )
                  CKPT=&CKPT &COMPRESS &EXT
                TO (SNODE
                  SFSDIR("&DIR2")
                  DSN=\xd5 FILETEST\&PROCESS.4\xd5 \
                  DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
                  DISP=RPL )
STEP5    COPY FROM (PNODE
                  LINK=(&INUSER,&INUSERP,RR,&CUU2)
                  DSN=MYHLQ.TESTFILE.VESDS.FB80
                  DCB=(DSORG=VSAM)
                  DISP=SHR )
                  CKPT=&CKPT &COMPRESS &EXT
                TO (SNODE
                  SFSDIR("&DIR2")
                  DSN=\xd5 FILETEST\&PROCESS.5\xd5 \
                  DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
                  DISP=RPL )

```

Extract an SFS File and Placing the File on the VM Reader Spool

In this example, one file is being extracted from the CDSFS filepool to be placed upon a VM reader spool. Note that the format of the SFSDIR statement must end with a period when only the main directory is referenced.

```
TESTSFS PROCESS      SNODE=CSD.VM.NODE NOTIFY=USER1
STEP01  COPY FROM    (DSN='PROFILE EXEC'          -
                      SFSDIR=('CDSFS:USER1.') -
                      TO      (DSN='!SPOOL USER1 TSET DATA')
```

Copy Files from VM to OpenVMS

This Process copies an existing VM file to an existing OpenVMS file.

```
VMSTEST PROCESS      SNODEID=(RGL,UNISEF)
COPY01  COPY FROM    (DSN='TEST FILE'          -
                      LINK=(SE9GWWT,TOM,RR,191) -
                      DISP=SHR                 -
                      PNODE)                   -
                      TO      (DSN='DISK:<RGL>VMTEST.DAT' -
                      DISP=OLD)
```

Copy a VM Sequential File to a CA-DYNAM/T Tape File (VSE)

This Process copies a sequential file to a DYNAM/T volume tape file. Note that DCB information and the disposition of RPL are specified. If the VSE file does not exist, RPL specifies the Process is to allocate the file NEW using the DCB information.

```
AMTAPE  PROCESS      SNODE=CD.VSE.NODE NOTIFY=%USER
STEP01  COPY FROM    (DSN='TESTA INPUT'        -
                      LINK=(IVB4100,WIVB4100,RR,202) -
                      DISP=SHR                 -
                      TO      (DSN='DYNAM.TAPE***'    -
                      DCB=(DSORG=PS BLKSIZE=3200 LRECL=80 RECFM=F) -
                      DISP=RPL                 -
                      UNIT=TNOASGN            -
                      SNODE)
```

Copy VM Sequential Files to CA-DYNAM/D Files (VSE)

This multi-step Process shows various ways of copying sequential files to DYNAM/D.

- STEP01 and STEP02 copy to a DYNAM/D file that has not been defined to DYNAM/D.
- STEP03 copies to a DYNAM/D file using a TYPEKEY containing DCB information.
- STEP04 copies to an existing DYNAM/D file that has not been defined to DYNAM/D.
- STEP05 copies a sequential file to a DYNAM/D file (not previously defined). The DYNAM/D file has different DCB information specified.

```

NODEFINE PROCESS SNODE=CD.VSE.NODE
STEP01 COPY FROM (DSN='TESTA INPUT' -
LINK=(IVB4100,RIVB4100,RR,202) -
PNODE DISP=SHR) -
TO (DSN='VSE.NODEF.STEP01' -
DCB=(DSORG=PS BLKSIZE=3200 LRECL=80 RECFM=F) -
UNIT=DNOASGN -
VOL=SER=POOLNAME -
SNODE DISP=RPL) -
STEP02 COPY FROM (DSN='CDFILE INPUT' -
LINK=(IVB4100,RIVB4100,RR,202) -
PNODE DISP=SHR) -
TO (DSN='VSE.NODEF.STEP02' -
DCB=(DSORG=PS BLKSIZE=3200 LRECL=80 RECFM=F) -
UNIT=DNOASGN -
VOL=SER=POOLNAME -
SNODE DISP=RPL) -
STEP03 COPY FROM (DSN='TESTA INPUT' -
LINK=(IVB4100,RIVB4100,RR,202) -
PNODE DISP=SHR) -
TO (DSN='VSE.NODEF.STEP03' -
TYPE=DYNAMD -
SNODE DISP=RPL) -
STEP04 COPY FROM (DSN='TESTA INPUT' -
LINK=(IVB4100,RIVB4100,RR,202) -
PNODE DISP=SHR) -
TO (DSN='VSE.NODEF.STEP03' -
DCB=(DSORG=PS BLKSIZE=3200 LRECL=80 RECFM=F) -
UNIT=DNOASGN -
SNODE DISP=RPL) -
STEP05 COPY FROM (DSN='TESTC INPUT' -
LINK=(SMI4100,RSMI4100,RR,202) -
PNODE DISP=SHR) -
TO (DSN='VSE.NODEF.STEP05' -
DCB=(DSORG=PS BLKSIZE=1000 LRECL=100 RECFM=F) -
UNIT=DNOASGN -
VOL=SER=POOLNAME -
SNODE DISP=RPL)

```

Copy Files from VM to i5/OS

This Process copies a sequential file from a Connect:Direct for VM node to an i5/OS node. The SYSOPTS parameter specifies that the data is to be copied to the Connect:Direct for i5/OS node as a member of a physical database file. CKPT=0K turns off the checkpointing feature.

```

TEST PROCESS SNODE=OS400 -
SNODEID=(USER1,PASSWD1) -
*****
* STEP 1 WILL COPY SEQUENTIAL TO SEQUENTIAL ( VM TO OS400)
*****
STEP1000 COPY FROM (PNODE -
LINK=(QACD,RQACD,RR,191) -
DSN='TESTFILE' -
DCB=(LRECL=80,RECFM=F,DSORG=PS,BLKSIZE=80) -
DISP=SHR -
CKPT=0K -
TO (SNODE -
DSN='TEST/PDS(STEP1000)' -
SYSOPTS="\ \
\ TYPE ( MBR ) \
\ MAXMBRS ( *NOMAX ) \
\ RCDLEN ( 92 ) \
\ TEXT ( 'ADDED BY PROCESS TEST \
\ IN STEP1000' ) \
\ \
DISP=RPL)

```

Copy VSAM Files from VM to i5/OS

This Process copies a VSAM file from a Connect:Direct for VM node to a sequential file on a Connect:Direct for i5/OS node. The DSN parameter on the COPY TO side specifies the destination object name based on

the i5/OS standard file naming conventions and must be in single quotation marks. CKPT=0K turns off the checkpointing feature.

```

TEST1    PROCESS    SNODE=OS400          -
                SNODEID=(USER1,PASSWD1)
*****
* STEP 2000 WILL COPY VSAM TO SEQUENTIAL (VM TO OS400)
*****
STEP2000 COPY FROM (PNODE
                LINK=(QACD,RQACD,RR,192)
                DSN=SCQA1.TESTFILE
                DCB=(DSORG=VSAM,LRECL=80)
                DISP=SHR)
                CKPT=0K
                TO (SNODE
                DSN='TEST1/PDS(STEP2000)'
                SYSOPTS="\
                \ TYPE ( FILE ) \
                \ MAXMBRS ( *NOMAX ) \
                \ RCDLEN ( 92 ) \
                \ TEXT ( 'ADDED BY PROCESS TEST1 \
                \ IN STEP2000' ) \
                \"\
                DISP=RPL)

```

VM-Initiated Copy from i5/OS to VM

This Process is initiated from the Connect:Direct for VM/ESA to copy a file from an i5/OS node to a VM node. The contents of the SYSOPTS parameter specifies that the object being copied is to be transferred in save object format.

```

TEST2    PROCESS    SNODE=OS400          -
                SNODEID=(USER1,PASSWD1)
*****
* COPY PROCESS FROM OS400 TO VM (VM INITIATED)
*****
STEP4000 COPY FROM (SNODE
                DSN=' TEST2 / SAVEFILE1 . FILE '
                SYSOPTS="\
                \ TYPE ( OBJ ) \
                \"\
                TO (DSN='IAOB001 REGRESS1'
                LINK=(QACD,WQACD,W,192)
                DCB=(RECFM=FB,LRECL=528,BLKSIZE=5280,DSORG=PS)
                DISP=NEW
                PNODE)

```

VM-Initiated Copy VM to i5/OS Spool

This Process is initiated from the Connect:Direct for VM/ESA to copy a file from a VM node to a spooled output file on an i5/OS node.

```

TEST3    PROCESS    SNODE=OS400          -
                SNODEID=(USER1,PASSWD1)
*****
* VM TO OS400 COPY OF A FILE TO OS400 SPOOL (VM INITIATED)
*****
STEP5000 COPY FROM (PNODE
                LINK=(QACD,RQACD,RR,191)
                DSN='OS400 REP1'
                DISP=SHR)
                TO (SNODE
                DSN=TEST
                DISP=RPL
                SYSOPTS="\
                \ TYPE ( SPLF ) \
                \ CTLCHAR ( *FCFC ) \
                \ PRTQLTY( *NLQ ) \
                \"\

```

VM-Initiated Copy from i5/OS to VM Spool

This Process is initiated from the Connect:Direct for VM/ESA to copy a physical database file member from a Connect:Direct for i5/OS node to a VM spool file.

```

TEST3    PROCESS      SNODE=CD.DALLAS          -
          COPY FROM   SNODEID=(USER1,PASSWD1)
          *****
          * OS400 TO VM COPY OF A OS400 FILE TO VM SPOOL (VM INITIATED)
          *****
STEP6000 COPY FROM   (SNODE
                     DSN=' TEST / CD01 (VMTEST) '          -
                     SYSOPTS="\ \
                               \ TYPE ( MBR ) \             -
                               \ " \                         -
                     DISP=SHR)                               -
          TO          (DSN='!SPOOL VMTEST TESTFILE'         -
                     DCB=(DSORG=PS,RECFM=F,LRECL=80,BLKSIZE=80)) -
                     PNODE)

```

Copy a VM VSAM file to z/OS

This Process copies a VSAM Disk from a VM/ESA node to a sequential disk file on a z/OS node. If the file exists, IBM Connect:Direct replaces it. If the file does not exist, IBM Connect:Direct creates it as indicated by the DISP parameter.

```

VSMtozOS PROCESS      SNODE=CD.OS390.NDOEY
          COPY FROM   (DSN=MYNAME.TESTFILE.VESDS.FB80S      -
                     LINK=(VSAMDSK,RPASWD,,RR,195)         -
                     DCB=(DSORG=VSAM,LRECL=80)             -
                     DISP=SHR                               -
                     PNODE)
          TO          (DSN=HLQ.VSAMTST.FILE                 -
                     UNIT=SYSDA                            -
                     DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120) -
                     DISP=RPL                               -
                     SNODE)

```

Copy a VM CMS Disk File to a z/OS Node

This Process copies a CMS Disk file from a VM/ESA node to a disk on a Connect:Direct for z/OS node. If the file exists, IBM Connect:Direct replaces it. If the file does not exist, IBM Connect:Direct creates it as indicated by the DISP parameter.

```

VMT0ZOS  PROCESS      SNODE=CD.OS390.NODEX          -
          NOTIFY=USERID
STEP01   COPY FROM   (DSN='VMCMS FILE'                -
                     LINK=(USER1,RPASWWD,RR,191)         -
                     DISP=SHR                             -
                     PNODE)
          TO          (DSN=HLQ.VMTEST.FILE             -
                     UNIT=SYSDA                          -
                     DISP=RPL                             -
                     SNODE)

```

Copy a VM CMS Sequential File to UNIX

This Process copies a VM CMS Sequential file to UNIX in binary format.

```

PROC01  PROCESS                                     -
          &USER=MYUSR1                             -
          &COMPRESS=COMPRESS                       -
          &EXT=PRIME=X'40'                         -
          &USERPW=ALL                              -
          &SNODE=cd.unix.node                      -
          SNODEID=(userx,passwd)                   -
          PNODEID=(myuser1,passwd)                 -
          SNODE=&SNODE                              -
STEP1    COPY FROM                                 -
          ( PNODE                                  -
            DSN='TESTFILE STRESS01'                -
            LINK=(&USER,&USERPW,RR,192)            -
          )                                         -
          &COMPRESS          &EXT                 -
TO                                              -
          ( SNODE                                  -
            DSN='/tmp_mnt/home/fremont/mfinc1/hello' -
            DISP=(RPL)                             -
            SYSOPTS=":STRIP.BLANKS=NO:DATATYPE=BINARY:" -
          )                                         -

```

Copy a CMS Sequential File from VM to Microsoft Windows

This Process copies a CMS Sequential file from VM to Microsoft Windows.

```

PROC01  PROCESS                                     -
          &CKPT=0K                                  -
          &COMPRESS=,                               -
          &EXT=,                                     -
          &CUU1=0192                                -
          &USER=MYUSR1                              -
          &USERPW=ALL                              -
          &SNODE=WINNT                              -
          SNODE=&SNODE                              -
          SNODEID=(MYUSER)                         -
STEP1    COPY FROM                                 -
          ( PNODE                                  -
            LINK=(&USER,&USERPW,RR,&CUU1)          -
            DSN='TESTFILE STRESS04'                -
            DISP=SHR                                -
          )                                         -
          CKPT=&CKPT &COMPRESS &EXT               -
TO                                              -
          ( SNODE                                  -
            DSN='C:\OUTPUT\VM\OUT04'                -
            DISP=RPL                                -
          )                                         -

```

Copy a VM CMS File to Microsoft Windows

This Process copies a VM CMS file to Microsoft Windows.

```

PROC01  PROCESS SNODE=CD.NT.V1300  SNODEID=(USERID,PASSWD)
STEP1    COPY FROM                                     -
          ( PNODE                                     -
            LINK=(MYUSR1,ALL,RR,192)                 -
            DSN='TESTFILE FB80S'                     -
            DISP=SHR                                 -
          )                                         -
TO      (SNODE DSN='C:\OUTPUT\MYDATA' DISP=RPL)

```

Copy a DBCS File from Microsoft Windows to VMESA Using the KSCXEBC Translation Table

The following PC-to-host DBCS translation uses the supplied translation table KSCXEBC. This sample COPY statement copies a data set from a PC to a host Connect:Direct for VM/ESA node.

```
/*
*****
/*          PC to HOST DBCS translation using table KSCXEBC          */
*****
/
PCTOHOST  PROCESS SNODE=HOSTNODE  -
          HOLD=CALL                -
STEP01    COPY                                     -
          TO      (PNODE             -
                  DSN='h1q.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        -
                  DISP=SHR          -
                  )                  -

```

- 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.

Copy a DBCS File from VMESA to Microsoft Windows Using the EBCXKSC Translation Table

The following host-to-PC DBCS translation uses the supplied translation table EBCXKSC. This example COPY statement copies a data set from a host Connect:Direct for VM/ESA to a PC node.

```
/*
*****
/*          HOST to PC DBCS translation using table EBCXKSC          */
*****
/
HOSTTOPC  PROCESS SNODE=PCNODE  -
          HOLD=CALL                -
STEP01    COPY                                     -
          FROM    (PNODE             -
                  DSN='h1q.HOSTFILE' -
                  SYSOPTS="DBCS=EBCXKSC" -
                  DISP=(SHR)         -
                  )                  -
          TO      (SNODE             -
                  DSN=PCFILE        -
                  DISP=RPL          -
                  )                  -

```

- 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.

Copy a DBCS File From UNIX to VMESA Using the EBCXKSC Translation Table

The following UNIX-to-host DBCS translation uses the default translation table EBCXKSC. This example COPY statement copies a data set from a UNIX to a host Connect:Direct for VM/ESA node.

```
/*
/* 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 copy step is named STEP01.
- The SYSOPTS attribute 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.

Copy a DBCS File From VMESA to UNIX Using the KSCXEBC Translation Table

The following host-to-UNIX DBCS translation uses the default translation table KSCXEBC. This COPY statement copies a data set from a host Connect:Direct for VM/ESA to a UNIX node.

```
/*
/* 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 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.

Copy a Non-VSAM File on VMESA

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))
```

Copy All Files In a Group to a Destination with Fn Ft Unchanged on VMESA

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 following transfers occur:

```
FROM          TO
ABC ASSEMBLE  ----> ABC   ASSEMBLE
AAA ASSEMBLE  ----> AAA   ASSEMBLE
ABCD ASSEMBLE ----> ABCD  ASSEMBLE
SOURCE1 FILE  ----> SOURCE1 FILE
SOURCE2 FILE  ----> SOURCE2 FILE
```

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%.

Copy Selected Files from a Group to a Destination with Fn Ft Unchanged on VMESA

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.

```
FOI FROM GROUP='a* *'
TO   GROUP='a%1% %2%'
```

The following transfers occur:

```

FROM          TO
ABC ASSEMBLE ----> ABC ASSEMBLE
AAA ASSEMBLE ----> AAA ASSEMBLE
ABCD ASSEMBLE ----> ABCD ASSEMBLE
SOURCE1 FILE
SOURCE2 FILE

```

Copy Selected Files from a Group to a Destination with Added Characters In Fn Ft on VMESA

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 following transfers occur:

```

FROM          TO
ABC ASSEMBLE ----> AABC ASSEMBLE
AAA ASSEMBLE ----> AAAA ASSEMBLE
ABCD ASSEMBLE ----> AABCD ASSEMBLE
SOURCE1 FILE
SOURCE2 FILE

```

Copy Selected Files from a Group to a Destination with Characters Stripped from Fn Ft on VMESA

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 following transfers occur:

```

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.

Copy Selected Files with Equal Length Names from a Group to a Destination on VMESA

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 following transfers occur:

```
FROM TO  
ABC ASSEMBLE ----> ABC ASSEMBLE  
AAA ASSEMBLE ----> AAA ASSEMBLE  
ABCD ASSEMBLE  
SOURCE1 FILE  
SOURCE2 FILE
```

Copy Selected Files from a Group to a Destination with Fn Ft Formatting on VMESA

This example illustrates copying selected files from a group to a destination with Fn Ft formatting.

```
For FROM GROUP='* *'  
TO GROUP='%1%.%2%'
```

The following transfers occur:

```
FROM TO  
ABC ASSEMBLE ----> ABC.ASSEMBLE  
AAA ASSEMBLE ----> AAA.ASSEMBLE  
ABCD ASSEMBLE ----> ABCD.ASSEMBLE  
SOURCE1 FILE ----> SOURCE1.FILE  
SOURCE2 FILE ----> SOURCE2.FILE
```

Copy Selected Files from a Group to a Destination with Fn Ft Reversed and Formatted on VMESA

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 following transfers occur:

```
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: Do not make group file copies from disks linked R/W. The results are unpredictable.

Use SYSOPTS for DBCS in VMESA

The SYSOPTS statement declares that a Process is transferring a DBCS file. Include this statement on the host node COPY statement.

Support for multiple transfers with multiple translation tables is possible. All Processes support compression and checkpointing.

The following example uses the table name EBCXKSC and the default values x'0E' for so, and x'0F' for si.

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

The following example uses the table name KSCXEBC and the default values x'0E' for so, and x'0F' for si.

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

The following example uses the table name EBCXKSC and the NOSO value x'00' for so and si.

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

The following example uses the table name EBCXKSC and takes the defaults for so and si.

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

The following example uses the table name USERTAB and takes the defaults for so and si. USERTAB is a user-defined, customized translation table.

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

Copy a File from z/OS to VSE (DYNAM/T Tape Files)

This multi-step Process copies various sequential files from a z/OS node (the SNODE) to VSE DYNAM/T tape files at the PNODE. Different record formats are specified in the steps. Note that a TYPE file record is specified in STEP03. This record contains the DCB and UNIT information for this file. The TYPE record must be in the TYPE file at the destination (VSE node).

```

DYMTVSE  PROCESS      SNODE=CD.VSE.DALLAS  NOTIFY=%USER
STEP01   COPY  FROM   (DSN=$ABC.FDATA
                      DISP=SHR)
                      TO   (DSN=FDATA.DYMT.STEP01
                      DCB=(DSORG=PS BLKSIZE=80 LRECL=80 RECFM=FB)
                      DISP=RPL
                      LABEL=(RETPD=0007)
                      UNIT=TNOASGN
                      SNODE)
STEP02   COPY  FROM   (DSN=$ABC.SOURCE
                      DISP=SHR)
                      TO   (DSN=SOURCE.DYMT.STEP02
                      DCB=(DSORG=PS BLKSIZE=3120 LRECL=80 RECFM=FB)
                      UNIT=TNOASGN
                      LABEL=(RETPD=0007)
                      DISP=RPL
                      SNODE)
STEP03   COPY  FROM   (DSN=$ABC.REPORT
                      DISP=SHR)
                      TO   (DSN=REPORT.DYMT.STEP03
                      TYPE=OS390DYMT
                      LABEL=(RETPD=0007)
                      DISP=RPL
                      SNODE)
STEP04   COPY  FROM   (DSN=$ABC.UDATA
                      DISP=SHR)
                      TO   (DSN=UDATA.DYMT.STEP04
                      DCB=(DSORG=PS BLKSIZE=80 LRECL=0 RECFM=U)
                      UNIT=TNOASGN
                      DISP=RPL
                      LABEL=(RETPD=0007)
                      SNODE)
STEP05   COPY  FROM   (DSN=$ABC.VDATA
                      DISP=SHR)
                      TO   (DSN=VDATA.DYMT.STEP05
                      DCB=(DSORG=PS BLKSIZE=88 LRECL=84 RECFM=V)
                      UNIT=TNOASGN
                      LABEL=(RETPD=0007)
                      DISP=RPL  SNODE)

```

Copy a File from z/OS to VSE (DYNAM/T Tape Files)

This multi-step Process copies various sequential files from a z/OS node (the SNODE) to VSE DYNAM/T tape files at the PNODE. Different record formats are specified in the steps. Note that a TYPE file record is specified in STEP03. This record contains the DCB and UNIT information for this file. The TYPE record must be in the TYPE file at the destination (VSE node).

```

DYMTVSE  PROCESS      SNODE=CD.VSE.DALLAS  NOTIFY=%USER
STEP01   COPY FROM   (DSN=$ABC.FDATA
                      DISP=SHR)
                      TO   (DSN=FDATA.DYMT.STEP01
                      DCB=(DSORG=PS BLKSIZE=80 LRECL=80 RECFM=FB)
                      DISP=RPL
                      LABEL=(RETPD=0007)
                      UNIT=TNOASGN
                      SNODE)
STEP02   COPY FROM   (DSN=$ABC.SOURCE
                      DISP=SHR)
                      TO   (DSN=SOURCE.DYMT.STEP02
                      DCB=(DSORG=PS BLKSIZE=3120 LRECL=80 RECFM=FB)
                      UNIT=TNOASGN
                      LABEL=(RETPD=0007)
                      DISP=RPL
                      SNODE)
STEP03   COPY FROM   (DSN=$ABC.REPORT
                      DISP=SHR)
                      TO   (DSN=REPORT.DYMT.STEP03
                      TYPE=OS390DYMT
                      LABEL=(RETPD=0007)
                      DISP=RPL
                      SNODE)
STEP04   COPY FROM   (DSN=$ABC.UDATA
                      DISP=SHR)
                      TO   (DSN=UDATA.DYMT.STEP04
                      DCB=(DSORG=PS BLKSIZE=80 LRECL=0 RECFM=U)
                      UNIT=TNOASGN
                      DISP=RPL
                      LABEL=(RETPD=0007)
                      SNODE)
STEP05   COPY FROM   (DSN=$ABC.VDATA
                      DISP=SHR)
                      TO   (DSN=VDATA.DYMT.STEP05
                      DCB=(DSORG=PS BLKSIZE=88 LRECL=84 RECFM=V)
                      UNIT=TNOASGN
                      LABEL=(RETPD=0007)
                      DISP=RPL SNODE)

```

Copy a z/OS PDS Member to a New VSE File in a DYNAM Pool

This Process copies a PDS member on z/OS to a DYNAM-controlled file on VSE. The file on VSE will be dynamically allocated by DYNAM in a virtual disk pool defined to DYNAM as POOL01.

```

VSE0S390  PROCESS      PNODE=SC.VSE.NODE1 SNODE=SC.OS390.NODE1
STEP01    COPY FROM   (DSN=0S390.PDS.DATASET(TESTDATA)
                      DISP=(SHR,KEEP)
                      SNODE)
                      TO   (DSN=VSE.DYNAM.FILE
                      DISP=NEW
                      UNIT=DNOASGN
                      VOL=SER=POOL01
                      SPACE=(1,(300))
                      PNODE)

```

Copy a z/OS BSAM File to a VSE-Controlled Disk Data Set

Use this Process to transfer a z/OS/ESA BSAM file into a VSE CA-DYNAM/D or CA-EPIC controlled disk data set. The disk data set has already been defined to the appropriate system catalog. You do not need to specify DCB attributes for the output data set, these will be taken from the input data set.

This Process was written with symbolics for substitution.

```

ZOS2DYD1  PROC      SNODE=SC.OS390.NODE          -
                  PNODE=SC.VSE.NODE              -
                  &VSEDSN=USER01.TEST.COPYFILE    -
STEP0001  COPY      FROM ( SNODE                  -
                        DSN=USER01.TEST.OS390PRT01 -
                        DISP=SHR                   -
                        )                           -
                  TO  ( PNODE                      -
                        DSN=&VSEDSN                 -
                        UNIT=DLBONLY                -
                        DCB=(DSORG=PS,RECFM=FBA,LRECL=121,BLKSIZE=27951) -
                        )                           -
                  COMPRESS                          -
STEP0002  IF        (STEP0001 EQ 0) THEN          -
                  RUN TASK (PGM=DMNOTIFY,         -
                          PARM=('GOOD',&VSEDSN)) -
                  PNODE                           -
ELSE                                             -
                  RUN TASK (PGM=DMNOTIFY,         -
                          PARM=('FAIL',&VSEDSN)) -
                  PNODE                           -
EIF

```

Copy a z/OS Sequential Data Set or PDS to a VSE-Controlled Tape Data Set

This Process copies either a z/OS sequential data set or a z/OS partitioned data set into a CA-DYNAM/T or CA-EPIC non-controlled tape data set. The input resides on z/OS and the output file is written to tape (or cartridge) on VSE.

```

ZOS2DYT1  PROC      SNODE=SC.OS390.NODE0         -
                  PNODE=SC.VSE21.USER01         -
STEP0001  COPY      FROM ( SNODE                  -
                        DSN=RPITT1.LARGE.OS390.FILE -
                        DISP=SHR                   -
                        )                           -
                  TO  ( PNODE                      -
                        DSN=USER01.TEST.NONDYNAM     -
                        UNIT=TNOASGN                -
                        LABEL=(1,SL)                -
                        DISP=(NEW,CATLG)            -
                        DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=1600) -
                        )                           -

```

Copy a z/OS PDS Member to a VSE BSAM Sublibrary Member

This Process copies a z/OS PDS library member into a VSE BSAM sublibrary member. This sample Process is coded with symbolic parameters to allow you to use a generic Process to move any type of members (Dump, OBJ, Phase, Source, Processes, JCL) Use this Process when you are moving files from z/OS to VSE with the Process running on VSE.

The disk data set has already been defined to the appropriate system catalog. This Process was written with symbolics for substitution. When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

When you copy data into a VSE BSAM library, you must add either RECFM=F or RECFM=V to your DCB parameter. This specification depends on the type of input file. If you do not include the RECFM, the Process fails with the message SVSJ122I.


```

ZOS2LIB1 PROC  SNODE=SC.OS390.NODE -
                PNODE=SC.VSE.NODE -
                &MEMBER=, -
                &OS390PDS=USER01.PROCESS.LIB -
                &VSELIB=CONN.DIRECT.LIBRARIES -
                &VSESUB=USER01 -
                &VSETYP=N -
                &VSEVOL=USER03 -
STEP0001 COPY FROM ( SNODE -
                  DSN=&OS390PDS -
                  DISP=SHR -
                  SELECT=(&MEMBER) -
                  ) -
                TO ( PNODE -
                  DSN=&VSELIB -
                  DISP=SHR -
                  UNIT=DISK -
                  DCB=(DSORG=PS,RECFM=F) -
                  VOL=SER=&VSEVOL -
                  LIBR=(REPLACE=YES -
                        SLIBDISP=SHR -
                        SUBLIB=&VSESUB -
                        TYPE=&VSETYP) -
                  ) -
STEP0002 IF (STEP0001 EQ 0) THEN -
              RUN TASK (PGM=DMNOTIFY, -
                        PARM=('GOOD',&VSELIB)) -
              PNODE -
ELSE -
              RUN TASK (PGM=DMNOTIFY, -
                        PARM=('FAIL',&VSELIB)) -
              PNODE -
EIF

```

Copy a z/OS PDS Member to a VSE VSAM Sublibrary Member

This Process copies a z/OS PDS library member into a VSE VSAM sublibrary member. This sample Process is coded with symbolic parameters to allow you to use a generic Process to move any type of members (Dump, OBJ, Phase, Source, Processes, JCL). Use this Process when you are moving files from z/OS to VSE with the Process running on VSE.

For VSAM libraries you must specify the DSN and DSORG parameters on the FROM statement. You can optionally specify the catalog parameter if needed.

```

ZOS2LIB2 PROC  SNODE=SC.OS390.NODE -
                PNODE=SC.VSE.NODE -
                &MEMBER=, -
                &OS390PDS=USER01.TEST.JCLLIB -
                %VSELIB=CONN.DIRECT.LIB1 -
                &VSESUB=RXUSER01 -
                &VSETYP=JCL -
STEP0001 COPY FROM ( SNODE -
                  DSN=&OS390PDS -
                  DISP=SHR -
                  SELECT=(&MEMBER) -
                  ) -
                TO ( PNODE -
                  DSN=&VSELIB -
                  DISP=SHR -
                  DCB=(DSORG=VSAM) -
                  LIBR=(REPLACE=YES -
                        SLIBDISP=SHR -
                        SUBLIB=&VSESUB -
                        TYPE=&VSETYP) -
                  ) -

```

Copy a z/OS Sequential Data Set or z/OS PDS to a VSE-Controlled Tape Data Set

This Process copies either a z/OS sequential data set or a z/OS partitioned data set into a CA-DYNAM/T or CA-EPIC controlled tape data set. The input is from z/OS with the Process running on VSE. The disk data

set has already been defined to the appropriate system catalog. This Process was written with symbolics for substitution.

```

ZOS2TAP1 PROC  SNODE=SC.OS390.NODE -
                PNODE=SC.VSE.NODE -
                &VSECUU=CART -
                &VSEDSN=TEST.TAPE.FILE -
STEP0001 COPY FROM ( SNODE -
                    DSN=RPITT1.LARGE.OS390.FILE -
                    DISP=SHR -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=800) -
                    ) -
                TO ( PNODE -
                    DSN=&VSEDSN -
                    UNIT=&VSECUU -
                    LABEL=(1,SL) -
                    DISP=(NEW,KEEP) -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                    ) -
STEP0002 IF (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                PARM=('GOOD',&VSEDSN)) -
                PNODE -
                ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                PARM=('FAIL',&VSEDSN)) -
                PNODE -
EIF

```

Copy an HP NonStop File to a VSE VSAM File

This Process copies a file from an HP NonStop node to a VSE VSAM file. The transfer is initiated by the VSE node. The SYSOPTS SET XLATE ON parameter enables ASCII to EBCDIC translation.

```

VSE2TAN PROCESS  PNODE=SC.VSE.NODE1 SNODE=HPNONSTOP.NODE -
                SNODEID=(HPNONSTOP.NODE) -
                SNODEID=(123.456,TANUSR) -
STEP01 COPY FROM (DSN='\CLX.$SUP1.XFILES.SENDTEST' -
                DISP=SHR -
                SYSOPTS="SET XLATE ON" -
                SNODE) -
                TO (DSN=VSE.TEST.VSAM -
                DISP=RPL -
                DCB=(DSORG=VSAM) -
                PNODE)

```

Copy a VSE VSAM File to an HP NonStop Node

This Process copies a VSE VSAM file to an HP NonStop ESDS file. The transfer is initiated by the VSE node. The SYSOPTS SET XLATE ON parameter enables EBCDIC to ASCII translation.

```

VSE2TAN PROCESS  PNODE=SC.VSE.NODE1 SNODE=HPNONSTOP.NODE -
                SNODEID=(123.456,TANUSR) -
STEP01 COPY FROM (DSN=VSE.TEST.VSAM -
                DISP=RPL -
                DCB=(DSORG=VSAM) -
                PNODE) -
                TO (DSN='\$SUP1.TSTPROC.VSETEST' -
                DISP=NEW -
                SYSOPTS='\ " 'SET XLATE ON\ -
                \ , TYPE=E\ -
                \ , REC=100\ -
                \ , EXT=(100,100)' "\ -
                SNODE)

```

Copy a VSE DYNAM-Controlled File to a VM Node

This Process copies a DYNAM-controlled file from a VSE node to a VM node. The file is copied to the CDUSR CMS ID.

VSE2VM	PROCESS	PNODE=SC.VSE.NODE1	SNODE=SC.VM.NODE1	
STEP01	COPY FROM	(DSN=VSE.DYNAM.FILE		-
		DISP=SHR		-
		UNIT=DLBLONLY		-
		DCB=(DSORG=PS,LRECL=80,BLKSIZE=8000,RECFM=FB)		-
		PNODE)		-
	TO	(DSN='VMDYND TEMP02'		-
		DISP=RPL		-
		LINK=(CDUSR,XCDUSR,RR,301)		-
		SNODE)		-

Use a Typekey to Copy a VSE DYNAM-Controlled File to a VM Node

This Process copies a DYNAM-controlled file from a VSE node to a VM node. DCB and UNIT information is supplied through a TYPEKEY named DYNAMD.

VSE2VM	PROCESS	PNODE=SC.VSE.NODE1	SNODE=SC.VM.NODE1	
STEP01	COPY FROM	(DSN=VSE.DYNAM.FILE		-
		TYPE=DYNAMD		-
		PNODE)		-
	TO	(DSN='VMDYND TEMP02'		-
		DISP=RPL		-
		LINK=(CDUSR,XCDUSR,RR,301)		-
		SNODE)		-

Copy a VSE Sequential File to an OpenVMS Node

This Process copies a VSE sequential file to an OpenVMS node.

PROC01	PROCESS	PNODE=SC.VSE.NODE1	SNODE=SC.VMS.NODE2	
STEP01	COPY FROM	SNODEID=(VMSUSR,PASSWD)		-
		(DSN=VSE.TEST.DATA		-
		DCB=(BLKSIZE=2400,DSORG=PS,LRECL=80,RECFM=FB)		-
		UNIT=241		-
		SPACE=(10620,(45))		-
		DISP=SHR)		-
		CKPT=0K		-
	TO	(DSN='\$SUP:<VMSUSR>DATA.TST'		-
		SNODE)		-

Copy a VSE Sequential File to Another VSE Sequential File

This Process copies a sequential file from one VSE node to another VSE node, with checkpointing at 128K intervals.

PROC01	PROCESS	PNODE=SC.VSE.NODE1	SNODE=SC.VSE.NODE2	
STEP01	COPY FROM	PNODEID=(SUPERUSR,SUPERUSR)		-
		(DSN=VSE.TEST.DATA		-
		DCB=(BLKSIZE=2400,DSORG=PS,LRECL=80,RECFM=FB)		-
		UNIT=DISK		-
		VOL=SER=123456		-
		SPACE=(10620,(15))		-
		DISP=SHR)		-
		CKPT=128K		-
	TO	(DSN=VSE.CKPT.TEST		-
		DCB=(BLKSIZE=24000,DSORG=PS,LRECL=80,RECFM=FB)		-
		UNIT=243		-
		SPACE=(10530,(45))		-
		DISP=(NEW,KEEP))		-

Copy a VSE Non-Labelled Tape to a VSE Sequential File

This Process copies the second data set on a non-labeled tape from one VSE node to a sequential file on another VSE node.

```

VSETAPE  PROCESS      PNODE=SC.VSE.NODE1  SNODE=SC.VSE.NODE2
STEP01   COPY  FROM   (DSN=VSE.NLTAPE                -
                      UNIT=CART                    -
                      DCB=(DSORG=PS,LRECL=80,BLKSIZE=18000,RECFM=FB) -
                      LABEL=(2,NL)                 -
                      VOL=SER=123456               -
                      PNODE)                       -
                      TO   (DSN=VSE.SEQFILE         -
                      UNIT=TAPE                    -
                      SPACE=(3645,(60))           -
                      DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=14400) -
                      DISP=NEW                     -
                      SNODE)                       -

```

Copy the IBM Connect:Direct Message File to SL Tape on VSE

This Process copies the Connect:Direct for z/OS VSAM message file to a standard label tape device on another VSE node.

```

VSETAPE  PROCESS      PNODE=SC.VSE.NODE1  SNODE=SC.VSE.NODE2
COPY01   COPY  FROM   (DSN=ABC.MSG                -
                      DISP=SHR                    -
                      DCB=(DSORG=VSAM)           -
                      PNODE)                       -
                      TO   (DSN=ABC.MSG.TAPE      -
                      DISP=NEW                     -
                      UNIT=TAPE                    -
                      DCB=(DSORG=PS,LRECL=80,BLKSIZE=12000,RECFM=VB) -
                      LABEL=(1,SL)               -
                      DISP=NEW                     -
                      SNODE)                       -

```

Copy a Non-managed Disk Data Set into Another Non-managed CKD Disk Data Set (VSE)

Use this Process to copy a non-managed disk data set into another non-managed disk data set residing on a CKD device. This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values.

```

DSK2DSK1 PROC  PNODE=SC.VSE.NODE                -
                SNODE=SC.VSE.NODE                -
                &VSEDSN=GGREG1.TEST.NODYNAM1
STEP0001 COPY  FROM  ( PNODE                      -
                      DSN=LRR.LREC480.ADDX        -
                      DISP=(SHR)                  -
                      DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=480) -
                      VOL=SER=USER01              -
                      )                            -
                      TO  ( SNODE                  -
                      DSN=&VSEDSN                  -
                      DISP=(RPL)                   -
                      VOL=SER=USER02              -
                      SPACE=(6055,(25))           -
                      DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=13600) -
                      )                            -
STEP0002 IF    (STEP0001 EQ 0) THEN                -
                RUN TASK (PGM=DMNOTIFY,           -
                PARM=('GOOD',&VSEDSN))           -
                PNODE                               -
                ELSE                                -
                RUN TASK (PGM=DMNOTIFY,           -
                PARM=('FAIL',&VSEDSN))           -
                PNODE                               -
EIF

```

Copy a Non-controlled Disk Data Set to a Managed CKD Disk Data Set (VSE)

This Process copies a non-DYNAM/D or non-EPIC controlled disk data set into a DYNAM/D or EPIC managed CKD disk data set. The disk data set has already been defined to the appropriate system catalog. This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values.

```

DSK2DYD1 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.VSE.NODE -
                &VSEDSN=USER01.TEST.GDGCOPY1
STEP0001 COPY  FROM ( PNODE -
                    DSN=LRR.LREC480.ADDX -
                    DISP=SHR -
                    VOL=SER=USER01 -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=800) -
                    ) -
                TO ( SNODE -
                    DSN=&VSEDSN -
                    UNIT=DLBLONLY -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920) -
                    )
STEP0002 IF    (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&VSEDSN)) -
                    PNODE
                ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&VSEDSN)) -
                    PNODE
                EIF

```

Copy a Non-managed Disk File into a Start Track 1 FBA Non-controlled Data Set (VSE)

Use this Process to copy a Non-managed disk file into a DYNAM/D or EPIC start-track 1 FBA non-controlled data set.

This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values. CA-DYNAM/D or CA-EPIC will perform the disk allocation for IBM Connect:Direct but since the data set is allocated as a start-track 1 data set with a vol=ser it will not be a managed data set.

```

DSK2DYD2 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.VSE.NODE -
                &VSEDSN=USER01.TEST.FILENAME
STEP0001 COPY  FROM ( PNODE -
                    DSN=LRR.LREC480.ADDX -
                    DISP=SHR -
                    VOL=SER=USER01 -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=800) -
                    ) -
                TO ( SNODE -
                    DSN=&VSEDSN -
                    VOL=SERUSER04 -
                    UNIT=DNOASGN -
                    LABEL=(,,EXPDT=99365) -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920) -
                    )
STEP0002 IF    (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&VSEDSN)) -
                    PNODE
                ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&VSEDSN)) -
                    PNODE
                EIF

```

Copy to Non-TMS Controlled Tapes (VSE)

This Process copies two non-TMS controlled tapes. The input tape is 3480/3490 cartridge and the output tape is reel (3420). This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing. This Process uses symbolic values.

```
TAP2TAP1 PROC  PNODE=SC.VSE.USER01          -
                SNODE=SC.VSE.USER01        -
                &FCUU=CART                   -
                &TCUU=TAPE                   -
STEP001  COPY  FROM ( PNODE                  -
                    DSN=TEST.TAPE.FILE      -
                    UNIT=&FCUU              -
                    LABEL=(1,SL)            -
                    VOL=SER=(807012)        -
                    DCB=(RECFM=FB,LRECL=80, -
                          BLKSIZE=800)      -
                    )                        -
                TO  ( SNODE                  -
                    DSN=NON.DYNAM.TAPE      -
                    UNIT=&TCUU              -
                    LABEL=(1,SL,,EXPDT=99365) -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80, -
                          BLKSIZE=16000)    -
                    )                        -
                COMPRESS                     -
```

Copy a Non-managed Disk File to a CA-DYNAM/D or CA-EPIC Start Track 1 FBA Non-controlled Data Set (VSE)

Use this Process to copy a non-managed disk file into a DYNAM/D or EPIC start-track 1 FBA non-controlled data set. This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing.

This Process uses symbolic values. CA-DYNAM/D or CA-EPIC will perform the disk allocation for IBM Connect:Direct. The data set is allocated as a start-track **1** data set with a VOL=SER it will not be a managed data set.

```
DSK2DYD3 PROC  PNODE=SC.VSE.NODE           -
                SNODE=SC.VSE.NODE           -
                &VSEDSN=USER01.TEST.NONDYD  -
                &VOLSER=FBA001              -
STEP0001 COPY  FROM ( PNODE                  -
                    DSN=LRR.LREC480.ADDX    -
                    DISP=SHR                -
                    VOL=SER=USER01          -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80, -
                          BLKSIZE=800)      -
                    )                        -
                TO  ( SNODE                  -
                    DSN=&VSEDSN             -
                    UNIT=DNOASGN           -
                    VOL=SER=&VOLSER         -
                    SPACE=(1,(5),RLSE)     -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80, -
                          BLKSIZE=27920)    -
                    )                        -
STEP0002 IF    (STEP0001 EQ 0) THEN         -
                RUN TASK (PGM=DMNOTIFY,    -
                          PARM=('GOOD',&VSEDSN)) -
                PNODE                       -
                ELSE                          -
                RUN TASK (PGM=DMNOTIFY,    -
                          PARM=('FAIL',&VSEDSN)) -
                PNODE                       -
EIF
```

Print a Managed Disk Data Set (VSE)

Use this Process to print the contents of a CA-DYNAM/D or CA-EPIC managed disk data set. The output will become a LST queue member under the name of &JBNAME. The disk data set has already been defined to the appropriate system catalog.

```

DYD2LST1 PROC  SNODE=SC.VSE.NODE -
                PNODE=SC.VSE.NODE -
                &JBNAME=GGGDYD00 -
                &JBNUMB=0000 -
                &VSEDSN=USER01.TEST.GDGPOWR1
STEP0001 COPY  FROM ( PNODE -
                    DSN=&VSEDSN -
                    DISP=SHR -
                    UNIT=DLBLONLY -
                    DCB=(RECFM=FBM,LRECL=133,BLKSIZE=1330) -
                    ) -
                TO ( SNODE -
                    DSN=&JBNAME. .&JBNUMB -
                    LST=( -
                    CC=M -
                    CLASS=Q -
                    COPIES=2 -
                    DISP=L) -
                    )

```

The previous Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values. You must specify the input DCB parameter (RECFM, LRECL); this information will be copied to the output data set.

Copy a Non-controlled Sequential File to a MSAM File (VSE)

This Process copies a non-controlled BSAM (sequential) file into a MSAM (VSAM Managed SAM) file. The disk data set has already been defined to the appropriate system catalog (the default ESDS model). This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values.

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

```

DSK2MSM1 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.VSE.NODE -
                &VSEDSN=USER01.TEST.MSAMFIL1
STEP0001 COPY  FROM ( PNODE -
                    DSN=LRR.LREC480.ADDX -
                    DISP=SHR -
                    VOL=SER=USER01 -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=800) -
                    ) -
                TO ( SNODE -
                    DSN=&VSEDSN -
                    DISP=RPL -
                    UNIT=DISK -
                    VOL=SER=USER06 -
                    SPACE=(80,(500,300)) -
                    VSAMCAT=(VSE.COMMON.CATALOG,X,X,,123) -
                    DCB=(DSORG=MSAM,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                    )
STEP0002 IF    (STEP0001 EQ 0) THEN
                RUN TASK (PGM=DMNOTIFY,
                        PARM=('GOOD',&VSEDSN))
                        PNODE
                ELSE
                RUN TASK (PGM=DMNOTIFY,
                        PARM=('FAIL',&VSEDSN))
                        PNODE
EIF

```

Copy a Non-controlled Tape Data Set to a Controlled Disk File (VSE)

This Process copies a non-CA-DYNAM/T or CA-EPIC controlled tape data set into a controlled disk file. The disk data set has already been defined to the appropriate system catalog. This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing.

```

TAP2DYD1 PROC      PNODE=SC.VSE.NODE          -
                   SNODE=SC.VSE.NODE          -
STEP001  COPY FROM ( PNODE                    -
                   (DSN=TEST.TAPE.FILE        -
                     UNIT=5A0                 -
                     LABEL=(1,NL)             -
                     VOL=SER=(777777)        -
                     DCB=(RECFM=FB,LRECL=1500, -
                           BLKSIZE=22500)     -
                   )
                   TO ( SNODE                  -
                       DSN=USER01.TEST.GDGCOPY1 -
                       UNIT=DLBLONLY          -
                       LABEL=(EXPDT=99365)    -
                       DCB=(RECFM=FB,LRECL=1500, -
                             BLKSIZE=22500)   -
                       )

```

Copy Non-managed Disk Data Set to a Non-managed Tape Data Set (VSE)

This Process copies a non-managed disk data set into a non-managed tape data set and runs on the same IBM Connect:Direct node using PNODE=SNODE processing. This Process uses symbolic values.

```

DSK2TAP1 PROC      PNODE=SC.VSE.NODE          -
                   SNODE=SC.VSE.NODE          -
                   &VSECUU=CART              -
                   &VSEDSN=TEST.TAPE.FILE    -
STEP0001  COPY FROM ( PNODE                    -
                   DSN=USER01.TEST.NODYNAM1   -
                   DISP=(SHR)                 -
                   UNIT=DISK                   -
                   VOL=SER=DOSRES             -
                   DCB=(DSORG=PS,RECFM=FB,LRECL=80, -
                           BLKSIZE=8000)      -
                   )
                   TO ( SNODE                  -
                       DSN=&VSEDSN            -
                       UNIT=&VSECUU          -
                       LABEL=(1,SL)          -
                       DISP=(NEW,KEEP)       -
                       DCB=(DSORG=PS,RECFM=FB,LRECL=80, -
                             BLKSIZE=800)     -
                       )
STEP0002  IF (STEP0001 EQ 0) THEN              -
           RUN TASK (PGM=DMNOTIFY,           -
                     PARM=('GOOD',&VSEDSN)) -
           PNODE                              -
           ELSE                                -
           RUN TASK (PGM=DMNOTIFY,           -
                     PARM=('FAIL',&VSEDSN)) -
           PNODE                              -
EIF

```


Copy a Managed Disk Data Set to Another Managed Data Set (VSE)

Use this Process to copy either a CA-DYNAM/D or CA-EPIC managed disk data set into another DYNAM/D or EPIC managed data set and reblock the output data set. The disk data set has already been defined to the appropriate system catalog.

```
DYD2DYD1 PROC PNODE=SC.VSE.NODE -
              SNODE=SC.VSE.NODE -
              &VSEDSN=USER01.TEST.GDGCOPY2 -
STEP0001 COPY FROM ( PNODE -
                   DSN=USER01.TEST.GDGCOPY1 -
                   DISP=(SHR) -
                   UNIT=DLBLOONLY -
                   DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                   ) -
              TO ( SNODE -
                  DSN=&VSEDSN -
                  UNIT=DLBLOONLY -
                  DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920) -
                  ) -
STEP0002 IF (STEP0001 EQ 0) THEN -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&VSEDSN)) -
                    PNODE -
            ELSE -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&VSEDSN)) -
                    PNODE -
            EIF
```

Copy a Managed Generation Disk Data Set to Another Managed Data Set (VSE)

Use this Process to copy either a CA-DYNAM/D or CA-EPIC managed disk data set into another DYNAM/D or EPIC managed data set. The input data set is CKD and the output data set is FBA. The disk data set has already been defined to the appropriate system catalog.

This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing. This Process uses symbolic values.

```
DYD2DYD2 PROC PNODE=SC.VSE.NODE -
              SNODE=SC.VSE.NODE -
              &CKDDSN=USER01.TEST.GDGCOPY2 -
              &FBADSN=USER01.TEST.FBACOPY1 -
STEP0001 COPY FROM ( PNODE -
                   DSN=&CKDDSN -
                   UNIT=DLBLOONLY -
                   DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                   ) -
              TO ( SNODE -
                  DSN=&FBADSN -
                  UNIT=DLBLOONLY -
                  DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920) -
                  ) -
STEP0002 IF (STEP0001 EQ 0) THEN -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&FBADSN)) -
                    PNODE -
            ELSE -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&FBADSN)) -
                    PNODE -
            EIF
```

Copy a Controlled Disk Data Set to a Controlled Tape Output File (VSE)

Use this Process to copy a CA-DYNAM/D or CA-EPIC controlled disk data set to a CA-DYNAM/T or CA-EPIC controlled tape output file on the same IBM Connect:Direct node by using PNODE=SNODE. All of the disk and tape data set names have been predefined to the appropriate system catalog.

```

DYD2DYT1 PROC PNODE=SC.VSE.NODE -
              SNODE=SC.VSE.NODE -
              &VSEDSN=USER01.TEST.TAPE1
STEP0001 COPY FROM ( PNODE -
                   DSN=USER01.TEST.COPYFILE -
                   UNIT=DLBLONLY -
                   DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                   ) -
              TO ( SNODE -
                  DSN=&VSEDSN -
                  UNIT=TNOASGN -
                  LABEL=(1,SL) -
                  DISP=(NEW,CATLG) -
                  DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=32000) -
                  )
STEP0002 IF (STEP0001 EQ 0) THEN
              RUN TASK (PGM=DMNOTIFY, -
                       PARM=('GOOD',&VSEDSN)) -
                       PNODE
              ELSE
              RUN TASK (PGM=DMNOTIFY, -
                       PARM=('FAIL',&VSEDSN)) -
                       PNODE
              EIF

```

Copy a Controlled BSAM Data Set to a MSAM Output Data Set (VSE)

This Process copies from a CA-DYNAM/D or CA-EPIC controlled BSAM data set into a MSAM (VSAM Managed SAM) output data set. The disk data set has already been defined to the appropriate system catalog (the default ESDS model).

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

You can ignore the IBM Connect:Direct information message: SVSG501I VSAM OPEN ERROR='A0'. ASSUMING ESDS. RETRYING OPEN.

```

DYD2MSM1 PROC PNODE=SC.VSE.NODE -
              SNODE=SC.VSE.NODE -
              &VSEDSN=USER01.TEST.MSAMFIL1
STEP0001 COPY FROM ( PNODE -
                   DSN=USER01.TEST.GDGCOPY1 -
                   DISP=(SHR) -
                   UNIT=DLBLONLY -
                   DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                   ) -
              TO ( SNODE -
                  DSN=&VSEDSN -
                  DISP=RPL -
                  UNIT=DISK -
                  VOL=SER=USER06 -
                  SPACE=(80,(500,300)) -
                  VSAMCAT=(VSE.COMMON.CATALOG,X,X,,123) -
                  DCB=(DSORG=MSAM,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                  )
STEP0002 IF (STEP0001 EQ 0) THEN
              RUN TASK (PGM=DMNOTIFY, -
                       PARM=('GOOD',&VSEDSN)) -
                       PNODE
              ELSE
              RUN TASK (PGM=DMNOTIFY, -
                       PARM=('FAIL',&VSEDSN)) -
                       PNODE
              EIF

```

This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing. This Process uses symbolic values.

Copy a Controlled Tape Data Set to a Controlled FBA Disk Output Data Set (VSE)

Use this Process to copy a CA-DYNAM/D or CA-EPIC controlled tape data set to CA-DYNAM/D or CA-EPIC controlled FBA disk output data set. The disk data set has already been defined to the appropriate system catalog.

This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values.

DYT2DYD1	PROC	PNODE=SC.VSE.NODE	-
		SNODE=SC.VSE.NODE	-
		&VSEDSN=USER01.TEST.FBAFILE1	-
STEP0001	COPY	FROM (PNODE	-
		DSN=USER01.TEST.TAPE1	-
		UNIT=TNOASGN	-
		DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=32000)	-
)	-
		TO (SNODE	-
		DSN=&VSEDSN	-
		DISP=(SHR)	-
		UNIT=DLBLONLY	-
)	-
STEP0002	IF	(STEP0001 EQ 0) THEN	-
		RUN TASK (PGM=DMNOTIFY,	-
		PARM=('GOOD',&VSEDSN))	-
		PNODE	-
	ELSE		-
		RUN TASK (PGM=DMNOTIFY,	-
		PARM=('FAIL',&VSEDSN))	-
		PNODE	-
	ENDIF		-

Copy a Controlled CKD Disk Data Set to a non-controlled Tape Data Set (VSE)

This Process copies a CA-DYNAM/D or CA-EPIC controlled CKD disk data set to a non-CA-DYNAM/T or CA-EPIC controlled tape data set. The disk data set has already been defined to the appropriate system catalog. The output data set DCB attributes will be propagated from the input file attributes. This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values.

DYD2TAP1	PROC	PNODE=SC.VSE.NODE	-
		SNODE=SC.VSE.NODE	-
		&VSECUU=CART	-
		&VSEDSN=TEST.TAPE.FILE	-
STEP0001	COPY	FROM (PNODE	-
		DSN=USER01.TEST.GDGCOPY1	-
		UNIT=DLBLONLY	-
		DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=8000)	-
)	-
		TO (SNODE	-
		DSN=&VSEDSN	-
		UNIT=&VSECUU	-
		LABEL=(1,SL)	-
		DISP=(NEW,KEEP)	-
)	-
STEP0002	IF	(STEP0001 EQ 0) THEN	-
		RUN TASK (PGM=DMNOTIFY,	-
		PARM=('GOOD',&VSEDSN))	-
		PNODE	-
	ELSE		-
		RUN TASK (PGM=DMNOTIFY,	-
		PARM=('FAIL',&VSEDSN))	-
		PNODE	-
	ENDIF		-

Copy a VSE Sublibrary Member from a BSAM Sublibrary to a Controlled Disk Data Set

This Process copies a VSE sublibrary member from a BSAM sublibrary to a CA-DYNAM/D or CA-EPIC controlled disk data set on the same IBM Connect:Direct node using PNODE=SNODE processing. All of the disk and tape data set names have been predefined to the appropriate system catalog. This Process was written with symbolic parameters to allow for a generic Process, so you must modify to your standards.

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

LIB2DYD1	PROC	SNODE=SC.VSE.NODE	-
		PNODE=SC.VSE.NODE	-
		&MEMBER=LIB2DYT1	-
		&VSEDSN=USER01.TEST.LIBCOPY1	-
		&VSELIB=CONN.DIRECT.LIBRARIES	-
		&VSESUB=USER01	-
		&VSETYP=N	-
		&VSEVOL=USER03	-
STEP0001	COPY	FROM (SNODE	-
		DSN=&VSELIB	-
		DISP=SHR	-
		VOL=SER=&VSEVOL	-
		DCB=(DSORG=PS)	-
		LIBR=(-
		SELMEM=&MEMBER	-
		SELSLIB=&VSESUB	-
		SELTYPE=&VSETYP)	-
)	-
		TO (PNODE	-
		DSN=&VSEDSN	-
		UNIT=DLBLONLY	-
		DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)	-
		DISP=(NEW,CATLG)	-
)	-
STEP0002	IF	(STEP0001 EQ 0) THEN	-
		RUN TASK (PGM=DMNOTIFY,	-
		PARM=('GOOD',&VSEDSN))	-
		PNODE	-
	ELSE		-
		RUN TASK (PGM=DMNOTIFY,	-
		PARM=('FAIL',&VSEDSN))	-
		PNODE	-
	ENDIF		-

Copy a VSE Sublibrary Member from a BSAM Sublibrary to a Controlled Tape Data Set

This member copies a VSE sublibrary member from a BSAM sublibrary to a CA-DYNAM/D or CA-EPIC controlled tape data set on the same IBM Connect:Direct node using PNODE=SNODE processing.

All of the disk and tape data set names have been predefined to the appropriate system catalog. You do not have to specify output DCB parameters, these will be copied from the input library DCB parameters.

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

This Process was written with symbolic parameters to allow for a generic Process. You must modify to your standards.

```

LIB2DYT1 PROC  SNODE=SC.VSE.NODE -
                PNODE=SC.VSE.NODE -
                &MEMBER=LIB2DYT1 -
                &VSEDSN=USER01.TEST.TAPE1 -
                &VSELIB=CONN.DIRECT.LIBRARIES -
                &VSESUB=USER01 -
                &VSETYP=JCL -
                &VSEVOL=USER03 -
STEP0001 COPY  FROM ( SNODE -
                    DSN=&VSELIB -
                    DISP=SHR -
                    VOL=SER=&VSEVOL -
                    DCB=(DSORG=PS) -
                    LIBR=( -
                        SELMEM=&MEMBER -
                        SELSLIB=&VSESUB -
                        SELTYPE=&VSETYP) -
                    ) -
                TO ( PNODE -
                    DSN=&VSEDSN -
                    UNIT=TNOASGN -
                    LABEL=(1,SL) -
                    DISP=(NEW,CATLG) -
                ) -
STEP0002 IF    (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&VSEDSN)) -
                    PNODE -
ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&VSEDSN)) -
                    PNODE -
EIF

```

Copy a VSE/POWER LST Queue Member to a Controlled Disk Data Set

This Process extracts a VSE/POWER LST queue member (where: DSN=power.jobname) and places the data into a CA-DYNAM/D or CA-EPIC controlled disk data set. The disk data set has already been defined to the appropriate system catalog.

```

LST2DYD1 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.VSE.NODE -
                CLASS=5 -
                &JBNAME=, -
                &JBDISP=D -
                &JBCLASS=A -
                &VSEDSN=USER01.TEST.GDGPOWR1 -
STEP0001 COPY  FROM ( PNODE -
                    DSN=&JBNAME -
                    LST=(CLASS=&JBCLASS DISP=&JBDISP) -
                ) -
                TO ( SNODE -
                    DSN=&VSEDSN -
                    UNIT=DLBLOONLY -
                    * DCB=(DSORG=PS,RECFM=VM,LRECL=133,BLKSIZE=137) -
                ) -
STEP0002 IF    (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&VSEDSN)) -
                    PNODE -
ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&VSEDSN)) -
                    PNODE -
EIF

```

You do not need to specify an output DCB parameter. This information will be obtained from the LST queue entry. The Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing and uses symbolic values.

Copy a BSAM VSE Sublibrary to a New VSE BSAM Library

This Process sends an entire BSAM VSE sublibrary into a new VSE BSAM library to be allocated on the SNODE. This Process uses symbolic values. You must specify all of the shown below, on the FROM side to Process BSAM libraries.

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

When you copy data into a VSE BSAM library, you must add either RECFM=F or RECFM=V to your DCB parameter. This specification depends on the type of input file. If you do not include the RECFM, the Process fails with the message SVSJ122I.

```
LIB2LIB1 PROC  SNODE=SC.VSE.NODE -
                PNODE=SC.VSE.NODE -
                &NEWLIB=USER01.TEST.FBALIB01 -
                &VSELIB=CONN.DIRECT.LIBRARIES -
                &VSESUB=USER01 -
                &VSETYP=* -
STEP0001 COPY  FROM ( PNODE -
                    DSN=&VSELIB -
                    DISP=SHR -
                    LIBR=(*) -
                    VOL=SER=USER03 -
                    DCB=(DSORG=PS,RECFM=FB,LRECL=80) -
                    ) -
                TO ( SNODE -
                    DSN=&NEWLIB -
                    UNIT=FBA -
                    DISP=NEW LIBR=(*) -
                    VOL=SER=FBA001 -
                    DCB=(DSORG=PS,RECFM=F) -
                    SPACE=(100,(4000),RLSE) -
                    ) -
STEP0002 IF    (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                PARM=('GOOD',&NEWLIB)) -
                PNODE -
                ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                PARM=('FAIL',&NEWLIB)) -
                PNODE -
EIF
```

Copy a BSAM VSE Sublibrary to a New z/OS PDS

This Process sends an entire BSAM VSE sublibrary into a new z/OS partitioned data set. The Process runs on the PNODE (VSE) and sends the data to the SNODE (z/OS). This Process uses symbolic values.

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

You must specify all of the parameters in this sample on the FROM side to Process BSAM libraries.

```

LIB2Z0S4 PROC  SNODE=SC.OS390.NODE -
                PNODE=SC.VSE.NODE -
                &OS390LIB=USER01.TEST.VSELIB -
                &VSELIB=CONN.DIRECT.LIBRARIES -
                &VSESUB=PROCESS -
                &VSETYP=N -
STEP0001 COPY  FROM ( PNODE -
                    DSN=&VSELIB -
                    DISP=SHR -
                    VOL=SER=USER03 -
                    DCB=(DSORG=PS) -
                    LIBR=(SELSLIB=&VSESUB -
                        SELTYPE=&VSETYP -
                        REPLACE=YES) -
                    ) -
                TO ( SNODE -
                    DSN=&OS390LIB -
                    DISP=RPL -
                    UNIT=SYSDA -
                    SPACE=(CYL,(5,1,100),RLSE) -
                    DCB=(DSORG=P0,RECFM=FB,LRECL=80,BLKSIZE=27920) -
                    ) -
                COMPRESS -
STEP0002 IF    (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&OS390LIB)) -
                    PNODE -
                ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&OS390LIB)) -
                    PNODE -
                EIF

```

Copy a MSAM Data Set to a Controlled BSAM Data Set (VSE)

Use this Process to copy a MSAM (VSAM Managed SAM) data set into a CA-DYNAM/D or CA-EPIC controlled BSAM data set.

```

MSM2DYD1 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.VSE.NODE -
                &VSEDSN=USER01.TEST.GDGCOPY1 -
STEP0001 COPY  FROM ( PNODE -
                    DSN=USER01.TEST.MSAMFIL1 -
                    DISP=OLD -
                    UNIT=DISK -
                    VOL=SER=USER06 -
                    VSAMCAT=(VSE.COMMON.CATALOG,X,X,,123) -
                    DCB=(DSORG=MSAM,RECFM=FB,LRECL=80,BLKSIZE=16000) -
                    ) -
                TO ( SNODE -
                    DSN=&VSEDSN -
                    DISP=NEW -
                    UNIT=DLBLONLY -
                    DCB=(DSORG=PS,RECFM=F,LRECL=80,BLKSIZE=16000) -
                    ) -
                COMPRESS -
STEP0002 IF    (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&VSEDSN)) -
                    PNODE -
                ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&VSEDSN)) -
                    PNODE -
                EIF

```

In the previous example, the disk data set has already been defined to the appropriate system catalog. This Process runs on the same IBM Connect:Direct node using PNODE=SNODE processing. This Process uses symbolic values.

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.

When you copy data into a VSE BSAM library, you must add either RECFM=F or RECFM=V to your DCB parameter. This specification depends on the type of input file. If you do not include the RECFM, the Process fails with the message SVSJ122I.

Copy a VSE VSAM to an i5/OS PDS Member

This Process copies a VSAM file from a VSE node to an i5/OS PDS member. The VSAM file resides on a catalog other than IJSYSUC, so a VSAMCAT parameter is coded.

```

PROC01  PROCESS      PNODE=SC.VSE.NODE1  SNODE=OS400.NODE2  -
                                SNODEID=(OS400ND,PWD400)
STEP01  COPY  FROM  (DSN=VSE.TEST.VSAM) -
                                DCB=(DSORG=VSAM) -
                                VSAMCAT=(VSESP.USER.CATALOG,1,1,,111) -
                                DISP=SHR) -
                                TO  (DSN='OS400X/PDSLIB(TSTDATA)') -
                                SYSOPTS=(TYPE(MBR)\ -
                                        \RCDLEN(100)\ -
                                        \FILETYPE(*DATA)\ -
                                DISP=RPL -
                                SNODE)

```

Copy a VSE VSAM File to an i5/OS Spooled File

This Process copies a VSAM file from a VSE node to an i5/OS spooled file. Page size is optional and dependent on the printer device.

```

PROC01  PROCESS      PNODE=SC.VSE.NODE1  SNODE=OS400.NODE2  -
                                SNODEID=(OS400ND,PWD400)
STEP01  COPY  FROM  (DSN=VSE.TEST.VSAM) -
                                DCB=(DSORG=VSAM) -
                                DISP=SHR) -
                                TO  (DSN='DATAFF') -
                                SYSOPTS=(TYPE(SPLF)\ -
                                        \CTLCHAR(*NONE)\ -
                                        \PAGE SIZE(66 378)\ -
                                        \SPOOL(*YES)\ -
                                SNODE)

```

Copy a VSE Librarian BSAM Member to a Preallocated z/OS PDS Member

This Process copies a VSE Librarian BSAM member into a preallocated z/OS partitioned data set (PDS) member. The disk data set has already been defined to the appropriate system catalog. This Process was written with symbolics for substitution.

When you reference BSAM libraries in a IBM Connect:Direct Process, you must specify: DSORG, DSN, UNIT, and VOL=SER= parameters.


```

LIB2Z0S1 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.OS390.NODE -
                &OS390LIB=USER01.TEST.VSEPROC -
                &VSELIB=CONN.DIRECT.LIBRARIES -
                &VSEMEM=, -
                &VSESUB=USER01 -
                &VSETYP=N -
                &VSEVOL=USER03 -
STEP0001 COPY FROM ( PNODE -
                  DSN=&VSELIB -
                  DISP=SHR -
                  UNIT=DISK -
                  DCB=(DSORG=PS) -
                  VOL=SER=&VSEVOL -
                  LIBR=(SELMEM=&VSEMEM -
                        SELSLIB=&VSESUB -
                        SELTYPE=&VSETYP) -
                  ) -
                TO ( SNODE -
                  DSN=&OS390LIB -
                  DISP=SHR -
                  ) -
STEP0002 IF (STEP0001 EQ 0) THEN -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&OS390LIB)) -
            PNODE -
            ELSE -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&OS390LIB)) -
            PNODE -
EIF

```

Copy a VSAM VSE Library Member to a Preallocated z/OS PDS Member

Use the Process to copy a VSAM VSE Library member into a preallocated z/OS partitioned data set (PDS) member.

The disk data set has already been defined to the appropriate system catalog. This Process was written with symbolics for substitution. When referencing VSAM libraries in a IBM Connect:Direct Process you must specify the DSORG and DISP parameters.

```

LIB2Z0S2 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.OS390.NODE -
                &MEMBER=DITTODVT -
                &OS390LIB=USER01.PROCESS.LIB -
                &VSELIB=CONN.DIRECT.LIB1 -
                &VSESUB=RXUSER01 -
                &VSETYP=JCL -
STEP0001 COPY FROM ( PNODE -
                  DSN=&VSELIB -
                  DISP=SHR -
                  DCB=(DSORG=VSAM) -
                  LIBR=(SELMEM=&MEMBER -
                        SELTYPE=&VSETYP -
                        SELSLIB=&VSESUB -
                        REPLACE=YES) -
                  ) -
                TO ( SNODE -
                  DSN=&OS390LIB -
                  DISP=SHR -
                  ) -
                COMPRESS EXT -
STEP0002 IF (STEP0001 EQ 0) THEN -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&OS390LIB)) -
            PNODE -
            ELSE -
            RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&OS390LIB)) -
            PNODE -
EIF

```

Copy a VSE/POWER LST Queue Member to a Preallocated z/OS PDS

Use this Process to copy a VSE/POWER LST queue member into a preallocated z/OS partitioned data set (PDS). The member name is submitted when the Process is submitted by overriding the symbolic &PINUMB.

The disk data set has already been defined to the appropriate system catalog. Verify that your job class in the LST queue matches the Process job class (&JBCLASS); otherwise the Process will end and not copy the data set. This Process uses symbolic values.

```
LST2ZOS1 PROC  PNODE=SC.VSE.NODE -
                SNODE=SC.OS390.NODE -
                CLASS=8 -
                &JBNAME=GGG3200 -
                &JBNUMB=0000 -
                &JBDISP=L -
                &JBCLASS=Q -
                &PINUMB=
STEP0001 COPY FROM ( PNODE -
                    DSN=&JBNAME -
                    LST=(CLASS=&JBCLASS,DISP=&JBDISP) -
                    ) -
                TO ( SNODE -
                    DSN=USER01.TEST.VSEDUMPS(&PINUMB) -
                    DISP=RPL -
                    ) -
                COMPRESS
STEP0002 IF (STEP0001 EQ 0) THEN
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&PINUMB)) -
                PNODE
                ELSE
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&PINUMB)) -
                PNODE
                EIF
```

Copy a File from UNIX HP to a Controlled Disk Data Set on VSE Using LU6.2

Use this Process to copy a file from UNIX HP into a CA-DYNAM/D or CA-EPIC controlled disk data set using the LU 6.2 protocol. The disk data set has already been defined to the appropriate system catalog. DCB information will be provided by the SNODE. This Process uses symbolic values.

```
UNX2DYD1 PROCESS PNODE=SC.VSE.USER01 -
                 SNODE=SC.UNIX.NODE -
                 &VSEDSN=USER01.TEST.UNIXFILE
STEP0001 COPY TO ( PNODE -
                 DSN=&VSEDSN -
                 UNIT=DLBLONLY -
                 ) -
                FROM ( SNODE -
                 DSN='/home/fremont/ddunc1/750070/airx02.dat' -
                 SYSOPTS=":xlate=no:strip.blanks=no:" -
                 ) -
                CKPT=1M -
                COMPRESS
STEP0002 IF (STEP0001 EQ 0) THEN
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('GOOD',&VSEDSN)) -
                PNODE
                ELSE
                RUN TASK (PGM=DMNOTIFY, -
                    PARM=('FAIL',&VSEDSN)) -
                PNODE
                EIF
```

Copy a File from HP UNIX to a VSE Controlled Disk Data Set Using TCP/IP

This Process copies a file from HP UNIX into a CA-DYNAM/D or CA-EPIC controlled disk data set using the TCP/IP protocol.

The disk data set has already been defined to the appropriate system catalog. Verify that you have updated your network map with the SNODE TCP/IP address and port number. DCB information will be provided by the SNODE. This Process was written with symbolics for substitution.

```

UNX2DYD2 PROCESS PNODE=SC.VSE.USER01 -
                SNODE=199.1.4.87 -
                &VSEDSN=USER01.TEST.UNIXFILE -
STEP0001 COPY TO ( PNODE -
                DSN=&VSEDSN -
                UNIT=DLBLONLY -
                ) -
                FROM ( SNODE -
                DSN='/home/fremont/ddunc1/750070/airx02.dat' -
                SYSOPTS=":xlate=no:strip.blanks=no:" -
                ) -
                CKPT=1M -
                COMPRESS -
STEP0002 IF (STEP0001 EQ 0) THEN -
                RUN TASK (PGM=DMNOTIFY, -
                PARM=('GOOD',&VSEDSN)) -
                PNODE -
                ELSE -
                RUN TASK (PGM=DMNOTIFY, -
                PARM=('FAIL',&VSEDSN)) -
                PNODE -
EIF

```

Copy a DBCS Data Set from VSE to UNIX Using the KSCXEBC Translation Table

This COPY statement copies a data set from a VSE node to a UNIX node using the 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 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.

Copy a DBCS Data Set from Microsoft Windows to VSE Using the KSCXEBC Translation Table

The following PC-to-host DBCS translation uses the supplied translation table KSCXEBC when copying data set from a PC to a host Connect:Direct for VSE node. 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='h1q.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       -
                          DISP=SHR        -
                          )                -

```

- 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.

Copy a DBCS Data Set from VSE to Microsoft Windows Using the EBCXKSC Translation Table

This COPY statement copies a data set from a host Connect:Direct for VSE to a PC node using the 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='h1q.HOSTFILE' -
                          SYSOPTS="DBCS=EBCXKSC" -
                          DISP=(SHR)       -
                          )                -
              TO     (SNODE                -
                          DSN=PCFILE       -
                          DISP=RPL        -
                          )                -

```

- 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.

Copy a DBCS Data Set from UNIX to VSE Using the EBCXKSC Translation Table

This COPY statement copies a data set from a UNIX to a host Connect:Direct for VSE node using the 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 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.

Copy a Data Set from a VSE Node to Another VSE Node

This example COPY statement copies a data set from one Connect:Direct for VSE node to another.

```

STEP1 COPY FROM (DSN=SRCDATA.SET           -
                 DISP=(SHR)               -
                 PNODE                     -
                 DCB=(DSORG=PS,LRECL=80,   -
                 RECFM=FB,BLKSIZE=3120)   -
                 UNIT=3380                 -
                 VOL=SER=VOL003            -
                 )                         -
      TO   (COMPRESS                       -
            (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.

Use SYSOPTS for DBCS in VSE

The SYSOPTS statement declares that a Process is transferring a DBCS file. Include this statement on the host node COPY statement.

Support for multiple transfers with multiple translation tables is possible. 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 uses the table name EBCXKSC and the default values x'0E' for **so**, and x'0F' for **si**.

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

(Transfers attempted in block mode can produce unpredictable results in the destination file. These results can compromise data integrity. You do not receive an error message in these cases.)

The following example uses the table name KSCXEBC and the default values x'0E' for **so**, and x'0F' for **si**.

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

The following example uses the table name EBCXKSC and the NOSO value x'00' for **so** and **si**.

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

The following example uses the table name EBCXKSC and takes the defaults for **so** and **si**.

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

The following example uses the table name USERTAB and takes the defaults for **so** and **si**. USERTAB is a user-defined, customized translation table.

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

Use Defaults to Allocate a File (z/OS to z/OS)

This Process designates that the destination file is placed on a specified unit and volume. The SPACE information is also supplied. Because DCB information is not specified for the destination data set, DCB information from the source data set is used.

Both checkpointing and compression are requested. If the Process receives an X37-type abend, the Process is requeued, because REQUEUE=YES is specified in the PROCESS statement. Corrective action can then be taken.

```
COPYSEQ  PROCESS      PNODE=CD.DALLAS      -
STEP01   COPY        FROM  (PNODE DSN=$DAL.PSDATA)  -
                                CKPT=10M          -
                                COMPRESS           -
                                TO    (SNODE DSN=$CHI.PSDATA -
                                DISP=(NEW,CATLG)      -
                                UNIT=3380           -
                                VOL=(SER=SYS009)     -
                                SPACE=(4096,(200,40),,ROUND))
```

Use Defaults to Allocate a File (z/OS to z/OS)

This Process designates that the destination file is placed on a specified unit and volume. The SPACE information is also supplied. Because DCB information is not specified for the destination data set, DCB information from the source data set is used.

Both checkpointing and compression are requested. If the Process receives an X37-type abend, the Process is requeued, because REQUEUE=YES is specified in the PROCESS statement. Corrective action can then be taken.

```

COPYSEQ  PROCESS          PNODE=CD.DALLAS          -
                                SNODE=CD.CHICAGO REQUEUE=YES      -
STEP01   COPY    FROM    (PNODE DSN=$DAL.PSDATA)                -
                                CKPT=10M                          -
                                COMPRESS                           -
                                TO    (SNODE DSN=$CHI.PSDATA      -
                                DISP=(NEW,CATLG)                   -
                                UNIT=3380                           -
                                VOL=(SER=SYS009)                   -
                                SPACE=(4096,(200,40),,ROUND))      -

```

Use %SUBDATE and %SUBTIME for a File Name (z/OS to z/OS)

This Process shows how to use the %SUBDATE and %SUBTIME variables to construct a unique dataset name based on the date and time of a file transfer submission.

%JDATE, %SUBDATE, and %SUBTIME are exclusive to Connect:Direct for z/OS.

```

CD1      PROCESS  SNODE=CD.VM.NODE          -
                                & DATE=%SUBDATE                    -
                                & TIME=%SUBTIME                    -
STEP01   COPY    FROM    (                                           -
                                DSN=ACCTNG.DAILY.DATA                -
                                SELECT=(PL*)                        -
                                )                                     -
                                TO    (                                 -
                                DSN=ACCTNG.UPDATES.D || & DATE || .T || & TIME -
                                DISP=(NEW,CATLG)                     -
                                DCB=(DSORG=PO,LRECL=80,RECFM=FB,BLKSIZE=8000) -
                                )                                     -

```

File Allocation Using a TYPE File (z/OS to z/OS)

The following Process uses a TYPE file to allocate a data set; therefore, UNIT, VOLUME, and SPACE parameters are not specified within the COPY statement.

The destination data set is allocated using definitions specified in the TYPE record, PSFILE.

```

COPYSEQ  PROCESS          PNODE=CD.DALLAS          -
                                SNODE=CD.CHICAGO REQUEUE=YES      -
STEP01   COPY    FROM    (DSN=DAL.PSDATA)                -
                                CKPT=10M                          -
                                COMPRESS                           -
                                TO    (DSN=CHI.PSDATA            -
                                TYPE=PSFILE)                   -

```

The following parameters make up the TYPE record, PSFILE. This TYPE record must be present at the destination node (SNODE).

```

DISP=(NEW,CATLG)          -
DCB=(BLKSIZE=3120,LRECL=80,DSORG=PS) -
UNIT=3380                  -
VOL=(SER=SYS009)          -
SPACE=(4096,(200,40),,ROUND) -

```

You can also place the IOEXIT parameter in a TYPE entry. Any parameters specified on the COPY statement take precedence over those coded in the TYPE file. See the appropriate IBM Connect:Direct platform's documentation for details on setting up entries in the TYPE file.

Copy a KSDS that Must be Extended

This COPY statement copies and sends a basic key-sequenced data set (KSDS) to the secondary node as a new KSDS that is required to be extended. Because the LIKE parameter is not included, the KEYLEN, KEYOFF, AND LRECL parameters had to be specified.

```
COPY FROM (PNODE DSN=EPETE1.KSDS   DISP=(SHR)   ) -
         TO (SNODE DSN=EPETE2.KSDSE DISP=(NEW,CATLG) -
            DSNTYPE=EXTREQ KEYLEN=8 KEYOFF=6 LRECL=80)
```

Copy a SAM File (z/OS to z/OS)

The following Process copies a SAM file from CD.DALLAS to a new file located at another site, CD.CHICAGO. The receiving node allocates the file using the same file attributes as the source data set because no file information is supplied. Both checkpointing and compression are requested within the Process. The submitting user is notified when the Process completes.

```
COPYSEQ PROCESS      PNODE=CD.DALLAS NOTIFY=%USER -
                SNODE=CD.CHICAGO
STEP01 COPY FROM (DSN=DAL.PSDATA) -
                CKPT=10M -
                DISP=SHR -
                COMPRESS -
                TO (DSN=CHI.PSDATA) -
                DISP=RPL
```

Copy a DFDSS Volume Dump (z/OS to z/OS)

This Process transmits a z/OS Data Facility Data Set Services (DFDSS) volume dump to tape. Note that the RECFM must be U.

```
DFDSS  PROCESS      SNODE=CD.OS390.ALTO
COPY1  /* STEP NAME */ -
        COPY FROM (DSN=DAT.P34ECH.DEC91 -
                   DISP=OLD DCB=(RECFM=U, BLKSIZE=32760)) -
                   COMPRESS -
        TO (DSN=DAT.P44ECH.DEC91 -
            DISP=(NEW,CATLG) LABEL=(,SL,,,RETPD=15) -
            DCB=(RECFM=U, BLKSIZE=32760) -
            UNIT=CART SNODE)
```

Copy an Entire PDS (z/OS to z/OS)

This COPY statement transmits a PDS to a new file located at another site. Because allocation data is not included in the Process, the receiving node allocates the file using the same file attributes as the source data set. Both checkpointing and compression are requested within the Process. Checkpointing only takes place at the member level when copying a PDS.

```
COPY FROM (DSN=PDS.SOURCE) -
         CKPT=1M -
         DISP=SHR -
         COMPRESS -
         TO (DSN=PDS.DEST) -
         DISP=RPL
```


Specify a Range and the NR Subparameter to Copy Selected PDS Members (z/OS to z/OS)

The following COPY statement shows how the NR (NOREPLACE) subparameter can be used when a range is specified. The member ABC is copied to the PDS, PDS.DEST. All members from FAA through GBB are copied if they do not already exist. The member PQR is also copied.

```
/* USE OF NR WHEN A RANGE IS SPECIFIED */
COPY FROM (DSN=PDS.SOURCE -
          SELECT=(ABC, -
                (FAA/GBB,,NR), -
                PQR)) -
        TO (DSN=PDS.DEST -
          DISP=OLD)
```

Copy One Member of a PDS (z/OS to z/OS)

This example illustrates three ways to copy a single member of a PDS.

- In STEP01, the FROM and TO member names are specified in the data set names.
- In STEP02, the FROM and TO member names are specified in the data set names, with the output member name changed.
- STEP03 uses the SELECT parameter as part of the FROM clause of the COPY statement, with a new member name specified on the data set name for the destination data set.

```
STEP01 COPY FROM (DSN=PDS.SOURCE(MEMBER)) -
        TO (DSN=PDS.DEST(MEMBER)) -
          DISP=RPL)
STEP02 COPY FROM (DSN=PDS.SOURCE(OLDMEM)) -
        TO (DSN=PDS.DEST(NEWNAME)) -
          DISP=RPL)
STEP03 COPY FROM (DSN=PDS.SOURCE -
          SELECT=OLDMEM) -
        TO (DSN=PDS.DEST(NEWNAME)) -
          DISP=RPL)
```

Copy a PDS and Excluding an Individual Member (z/OS to z/OS)

This COPY statement copies an entire PDS, with the exception of the member, MEM3, named in the EXCLUDE parameter.

```
COPY FROM (DSN=PDS.SOURCE -
          EXCLUDE=MEM3) -
        TO (DSN=PDS.DEST)
```

Copy a PDS and Excluding Members Generically (z/OS to z/OS)

This COPY statement shows how to exclude members generically. In this example, all members are copied, except members with names beginning with ABC.

```
COPY FROM (DSN=PDS.SOURCE -
          EXCLUDE=ABC*) -
        TO (DSN=PDS.DEST)
```

Copy a PDS and Using a Range to Exclude PDS Members (z/OS to z/OS)

The following COPY statement copies an entire PDS but uses a range to exclude any members from AAB through BBC.

```
COPY FROM (DSN=PDS.SOURCE -  
          EXCLUDE=((AAB/BBC))) -  
      TO (DSN=PDS.DEST)
```

The source file, PDS.SOURCE, contains the following members:

```
AAA, AAB, ABB, ABC, BBA, BBB, BBCA, BGG, XXX
```

Members AAA, BBCA, BGG, and XXX are copied. Members AAB, ABB, ABC, BBA, and BBB are not copied because they are within the range of members excluded in AAB/BBC. There is no member BBC to exclude; BBB was the last member in PDS.SOURCE within the specified range. Because BBCA is not part of the range to be excluded, it is copied. A generic range, like AAB*/BBC*, is not valid.

All range entries must be specified as subparameters and enclosed in parentheses.

Copy a PDS and Generically Selecting Members (z/OS to z/OS)

This COPY statement shows how to select generically all member names starting with PAID (the first four characters of the data set names). The remaining members of PDS.SOURCE are not copied.

```
COPY FROM (DSN=PDS.SOURCE -  
          SELECT=(PAID*)) -  
      TO (DSN=PDS.DEST)
```

Copy PDS Members Using the EXCLUDE and SELECT Parameters (z/OS to z/OS)

In this example, the data set, PDS.SOURCE, contains the following members:

```
A, AB, ABA, ABB, ABC, ABD, ABE, ACB, ACC, ACD, BAA, BAB, BAC, CDE, CDF
```

This COPY statement shows the use of the SELECT and EXCLUDE parameters:

```
COPY FROM (DSN=PDS.SOURCE -  
          SELECT=((BA/BBB),A*,ABC,(CDF,ZZZ)) -  
          EXCLUDE=(AC*,BAA,(ABC/AZ))) -  
      TO (DSN=PDS.DEST)
```

Results are as follows:

- A, AB, ABA, ABB are copied because they are generically selected by A*.
- ABC is copied because a specific ABC selection overrides the EXCLUDE range (ABC/AZ).
- ABD and ABE are not copied because they are excluded by the range ABC/AZ.
- ACB, ACC, and ACD are not copied because they are excluded generically by AC*.
- BAA is not copied because the specific BAA EXCLUDE overrides the SELECT range (BA/BBB).
- BAB and BAC are copied because they are selected by the range (BA/BBB).
- CDE is not copied because it is not selected.
- CDF is copied because it is specifically selected by (CDF,ZZZ); ZZZ is the new name.

Copy a PDS Using the ALIAS Parameter with SELECT and EXCLUDE (z/OS to z/OS)

This COPY statement shows how the ALIAS parameter works when other optional parameters are specified.

```
COPY      FROM (DSN=PDS.SOURCE -
              ALIAS=Y           -
              EXCLUDE=C3        -
              SELECT=(A,C1))    -
          TO  (DSN=PDS.DEST)
```

In this example, the data set PDS.SOURCE has the following members and associated aliases:

Members	Aliases
A	A1 A2
B	
C	C1 C2 C3

The data set PDS.DEST contained no members and no aliases before the COPY operation. After the COPY operation, PDS.DEST contains the following members and aliases:

Members	Aliases
A	A1 A2
C	C1 C2

The explanation follows:

- A is copied because it was selected by member name.
- A1 is copied because ALIAS=Y and A1 is an alias for member A.
- A2 is copied because ALIAS=Y and A2 is an alias for member A.
- B is not copied because it was not selected.
- C is copied because ALIAS=Y and C is the true member name for C1.
- C1 is copied because it was selected by member name.
- C2 is copied because ALIAS=Y and C1 (another alias for member C) was selected by member name.
- C3 is not copied because it was specifically excluded.

Copy a PDS Member to Tape (z/OS to z/OS)

The following Process copies a PDS member to a tape device, specifying the PDS member as part of the DSN. Because you can only copy a single PDS member to tape in one copy step, DSORG=PS must be coded on the TO clause of the Connect:Direct for z/OS COPY statement.

```
COPYSEQ  PROCESS  PNODE=CHICAGO           -
                SNODE=MINNEAPOLIS        -
STEP01   COPY    FROM (DSN=MY.PDS.FILE(MEMBER) -
                DISP=SHR PNODE)          -
                TO  (DSN=TAPE.FILE        -
                DISP=(NEW,CATLG)         -
                UNIT=TAPE                 -
                LABEL=(,SL)               -
                DCB=(DSORG=PS))
```

You can copy multiple PDS members by coding multiple COPY steps in a Process, and either append the additional PDS members to the first file or create separate tape files for each member. In the following example, MEMB is appended to TAPE.FILE.

```

COPYSEQ  PROCESS  PNODE=CHICAGO          -
          SNODE=MINNEAPOLIS          -
STEP01   COPY    FROM  (DSN=MY.PDS.FILE(MEMA) -
                      DISP=SHR PNODE)     -
          TO      (DSN=TAPE.FILE        -
                      DISP=(NEW,CATLG)   -
                      UNIT=TAPE         -
                      DCB=(DSORG=PS))   -
STEP02   COPY    FROM  (DSN=MY.PDS.FILE(MEMB) -
                      DISP=SHR PNODE)     -
          TO      (DSN=TAPE.FILE        -
                      DISP=(MOD,CATLG)   -
                      UNIT=TAPE         -
                      LABEL=(,SL)       -
                      DCB=(DSORG=PS))   -

```

Use the IOEXIT Parameter (z/OS to z/OS)

You can specify use of an I/O exit by the inclusion of the IOEXIT keyword on the COPY statement. The IOEXIT keyword is valid in either the FROM or TO clause of the COPY statement. You can specify a different user-written IOEXIT on each side as shown in the following example. The exit must, however, reside on the node where it is referenced.

The exit referenced in this COPY (OUEXT03) must reside in an authorized loadlib at the destination site.

In this example, INEXT01, an IOEXIT program, on the source IBM Connect:Direct node will be invoked and passed two parameters—a character string ('DB0A05') and a hexadecimal value (X'0E'). It will pass records using IBM Connect:Direct to OUEXT03, an IOEXIT on the destination IBM Connect:Direct node.

```

COPY  FROM  (PNODE
            IOEXIT=(INEXT01,C'DB0A05',X'0E')) -
      TO    (SNODE
            IOEXIT=OUEXT03)                  -

```

Copy to a New SMS-Controlled Data Set (z/OS to z/OS)

This COPY statement transmits a sequential file to a new SMS-controlled file at another site. The new data set will be allocated using the DCB parameters that are specified within the DATACLAS definition on the receiving node.

```

COPY  FROM  (DSN=SEQ.SOURCE) -
      TO    (DSN=SEQ.DEST    -
            DATACLAS=DCLASS1 -
            STORCLAS=SCLASS1  -
            MGMTCLAS=MCLASS1) -

```

Copy to a New SMS Data Set Using LIKE (z/OS to z/OS)

This COPY statement transmits a sequential file to a new SMS-controlled file at another site. The new data set will be allocated using the DCB parameters from the data set specified on the LIKE parameter.

```

COPY  FROM  (DSN=SEQ.SOURCE) -
      TO    (DSN=SEQ.DEST    -
            LIKE=MODEL.DATASET.DEST -
            STORCLAS=SCLASS1  -
            MGMTCLAS=MCLASS1) -

```

Create and Copy a PDSE Data Set (z/OS to z/OS)

This COPY statement transmits a PDS file to a new PDSE file at another site. The DSNTYPE parameter is used to specify that the receiving data set is a LIBRARY (another name for a PDSE file). The data set is also allocated with the STORCLAS parameter to ensure that the SMS controls the data set.

```
COPY FROM (DSN=PDS.SOURCE) -  
      TO (DSN=PDSE.DEST -  
         DSNTYPE=LIBRARY -  
         DATACLAS=DCLASS1 -  
         STORCLAS=SCLASS1)
```

Create and Copy a VSAM KSDS Data Set (z/OS to z/OS)

This COPY statement transmits a VSAM KSDS file to a new VSAM KSDS at another site. The KEYLEN parameter is used to specify the length of the key for the data set. The RECO RG parameter specifies that the output VSAM data set is a KSDS. The LRECL parameter specifies the maximum length of a record in the VSAM data set. The KEYOFF parameter specifies the offset to the key within each record. Note that the offset is relative to 0.

```
COPY FROM (DSN=VSAM.KSDS.SOURCE) -  
      TO (DSN=VSAM.KSDS.DEST -  
         DISP=(NEW,CATLG) -  
         RECO RG=KS -  
         KEYLEN=16 -  
         KEYOFF=20 -  
         LRECL=256 -  
         STORCLAS=SCLASS1 -  
         MGMTCLAS=MCLASS1)
```

Create and Copy a VSAM ESDS Data Set (z/OS to z/OS)

This COPY statement transmits a VSAM ESDS file to a new VSAM ESDS at another site. The RECO RG parameter specifies that the target VSAM data set is an ESDS. The LRECL parameter specifies the maximum length of a record in the VSAM data set.

```
COPY FROM (DSN=VSAM.ESDS.SOURCE) -  
      TO (DSN=VSAM.ESDS.DEST -  
         DISP=(NEW,CATLG) -  
         RECO RG=ES -  
         LRECL=128 -  
         STORCLAS=SCLASS1 -  
         MGMTCLAS=MCLASS1)
```

Create and Copy a VSAM RRDS Data Set (z/OS to z/OS)

This COPY statement transmits a VSAM RRDS file to a new VSAM RRDS at another site. The RECO RG parameter specifies that the target VSAM data set is an RRDS. The LRECL parameter specifies the maximum length of a record in the VSAM data set.

```
COPY FROM (DSN=VSAM.RRDS.SOURCE) -  
      TO (DSN=VSAM.RRDS.DEST -  
         DISP=(NEW,CATLG) -  
         RECO RG=RR -  
         LRECL=300 -  
         STORCLAS=SCLASS1 -  
         MGMTCLAS=MCLASS1)
```

Create and Copy a VSAM Linear Data Set (z/OS to z/OS)

This COPY statement transmits a VSAM LINEAR file to a new VSAM LINEAR file at another site. The RECOG parameter specifies that the target VSAM data set is a LINEAR file.

```
COPY FROM (DSN=VSAM.LINEAR.SOURCE) -
      TO (DSN=VSAM.LINEAR.DEST -
          DISP=(NEW,CATLG) -
          RECOG=LS -
          STORCLAS=SCLASS1 -
          MGMTCLAS=MCLASS1)
```

Copy a Data Set with a Security Profile (z/OS to z/OS)

This COPY statement transmits a sequential file to a new sequential file at another site. The security profile of the model data set is used as a model to create a generic security profile for the new data set.

```
COPY FROM (DSN=SEQ.SOURCE) -
      TO (DSN=SEQ.DEST -
          DISP=(NEW,CATLG) -
          SECMODEL=(SEQ.DATASET.DEST,GENERIC) -
          STORCLAS=SCLASS1 -
          MGMTCLAS=MCLASS1)
```

Copy to z/OS Nodes with Unique Member Name Allocation (AXUNIQ Exit)

The following examples demonstrate how IBM Connect:Direct resolves member names when UNIQUE=YES is specified on the SYSOPTS parameter of the COPY TO statement.

The member name is made unique as follows:

- The member name specified in the TO DSN (or defaulted from the FROM DSN) is used as a seed for comparison. If the seed name is not unique on the receiving node, IBM Connect:Direct will modify the specified member name to create a unique name.
- If the seed name is less than eight characters long, IBM Connect:Direct appends a unique numeric character to the member name, starting with 1. If the member name formed by the seed and the suffix exists already, the suffix is incremented until a unique name is created.
- The suffix appended can be as long as seven digits.
- If the seed name is eight characters long, IBM Connect:Direct will truncate the name from the right to limit the name to eight characters. This truncation will also take place if the seed name and its appended suffix are more than eight characters long.

Resolution of a Unique Member Name by Appending a Digit

This example assumes that HLQ.PDS already contains the members DATA, DATA1, DATA2, and DATA11. Because the member name in the example, DATA, already exists on the z/OS TO node, a different member name will have to be created. The AXUNIQ exit, invoked by SYSOPTS="UNIQUE=YES" does this for you by adding numeric digits to the specified member name until a unique member name is achieved.

```
COPY FROM (DSN=\app101\data) -
      TO (DSN=HLQ.PDS(DATA) SYSOPTS="UNIQUE=YES")
```

IBM Connect:Direct resolves the member name to DATA3.

Resolution of a Unique Member Name by Truncating and Appending a Digit

This example assumes that HLQ.PDS already contains the member DATEABLE. Because the member name in the example, DATAFILE, already exists on the z/OS TO node, a different member name will have to be specified. The AXUNIQ exit, invoked by SYSOPTS="UNIQUE=YES", does this for you by dropping the

rightmost byte and adding a numeric digit to the specified member name until a unique member name is achieved.

```
COPY FROM (DSN=\app101\data)
      TO (DSN=HLQ.PDS(DATAFILE) SYSOPTS="UNIQUE=YES")
```

IBM Connect:Direct resolves the member name to DATAFIL1.

Copy a Sequential File from z/OS to a Member of a Physical Data Base File on an i5/OS Node

This Process copies a sequential file from z/OS to a member of a physical data base file on the i5/OS node. The RUN TASK then sends a message notifying an i5/OS user that the Process completed.

All SYSOPTS keyword values must be enclosed in parentheses, and the entire SYSOPTS string must be enclosed in double quotation marks. IBM Connect:Direct syntax requires backslashes to continue the SYSOPTS over multiple lines when the Process is submitted from a z/OS node. Bracketing backslashes allow for continuation of quotation marks when they begin and end on different lines.

Specifying COMPRESS without a subparameter indicates that blanks are compressed during transmission and converted back to the original string during decompression. The default for the COMPRESS parameter is PRIMEchar=X'40'.

```
PROC#01  PROCESS  SNODE=OS400.CDI1          -
                PNODE=SC.OS390.CD5A      -
                PRTY=8                    -
                NOTIFY=%USER              -
                CLASS=4                    -
                SNODEID=(USERID,PSWRD)     -
STEP001  COPY    FROM  (PNODE              -
                      DSN=CD.OS400.SEQFILE1 -
                      DISP=SHR)           -
                TO    (SNODE              -
                      DSN='CD/OS400(TEST01) ' -
                      SYSOPTS=\"TYPE(MBR)\ -
                      \TEXT('CREATED BY PROC#001')\ -
                      \RCDLEN(133)\" \ -
                      DISP=RPL)           -
STEP002  RUN TASK (PGM = OS400) SNODE      -
                SYSOPTS=\"\"              -
                \CMD(\                     -
                \SNDBRKMSG\                -
                \MSG('PROCESS PROC#01 HAS COMPLETED')\ -
                \TOMSGQ(DSP07)\           -
                \ )\                       -
                \"\
```

Copy a Member of a Physical Data Base File from i5/OS to a Sequential File on z/OS

This Process copies a member of a physical data base file from the i5/OS node to a sequential file on z/OS. DCB information is specified for file allocation on z/OS. The SYSOPTS keyword value is enclosed in parentheses, and the SYSOPTS string is enclosed in double quotation marks. Specifying COMPRESS EXT indicates that repetitive strings in the data will be compressed and converted to the original string during decompression.

```

PROC#001  PROCESS      SNODE=OS400.CDI1          -
                                PNODE=SC.OS390.CD5A          -
                                PRTY=8                    -
                                NOTIFY=%USER              -
                                CLASS=4                   -
                                HOLD=NO                   -
                                SNODEID=(RSMITH,ROGER)     -
STEP001  COPY FROM (SNODE          -
                                DSN='ABC/OS400(TEST01) '   -
                                SYSOPTS="TYPE(MBR)"        -
                                DISP=SHR)                 -
                                COMPRESS EXT               -
                                TO (PNODE                   -
                                DSN=ABC.OS400.SEQFILE9     -
                                DCB=(RECFM=FB,LRECL=133,  -
                                BLKSIZE=23408)            -
                                DISP=RPL)

```

Copy a Data Set from z/OS to a Spooled File on i5/OS

This example copies a data set from z/OS to a spooled file on an i5/OS node. The specified SYSOPTS parameters override the defaults defined for the print device at installation. The parameter CTLCHAR(*FCFC) is always used when the source z/OS file has a record format (RECFM) of xxA, which indicates that it contains ANSI carriage control.

```

TEST01A  PROCESS      SNODE=OS400.CDI1          -
                                PNODE=SC.OS390.CD5A          -
                                PRTY=8                    -
                                NOTIFY=%USER              -
                                CLASS=4                   -
                                HOLD=NO                   -
                                SNODEID=(RSMITH,ROGER)     -
STEP001  COPY FROM (PNODE          -
                                DSN=CD.OS400.REPORTS       -
                                DISP=SHR)                 -
                                TO (SNODE                   -
                                DSN=REPORTS401              -
                                SYSOPTS=\ "TYPE(SPLF)\     -
                                \DEV(*JOB)\                -
                                \DEVTYPE(*IPDS)\           -
                                \CTLCHAR(*FCFC)\           -
                                \SPOOL(*YES)\              -
                                \PRTQLTY(*NLQ)\            -
                                \PRTTXT('*** END OF PAGE  -
                                ***')\                    -
                                \JUSTIFY(100)\             -
                                \OUTQ(*JOB)\               -
                                \COPIES(2)\                -
                                \MAXRCDS(999)\             -
                                \FILESEP(2)\               -
                                \HOLD(*YES)\               -
                                \SAVE(*YES)\               -
                                \OUTPTY(*JOB)\             -
                                \USRDTA(RSMITH)"\         -
                                DISP=RPL)

```


Copy a Member of a PDS from z/OS to a Spooled File on i5/OS

This Process copies a member of a PDS from z/OS to a spooled file on an i5/OS node. The SYSOPTS parameter PRTQLTY specifies near-letter-quality print.

```
PROC#001 PROCESS      SNODE=OS400.CDI1      -
                    PRTY=8                          -
                    NOTIFY=RSMITH                   -
                    CLASS=4                          -
                    SNODEID=(RSMITH,ROGER)           -
STEP001  COPY FROM    (PNODE                             -
                    DSN=CD.OS400.CNTL(LRECL80)      -
                    DISP=SHR)                       -
                    TO      (SNODE                     -
                    DSN=REPORTS402                   -
                    SYSOPTS=\\                        -
                               \TYPE(SPLF)\           -
                               \PRTQLTY(*NLQ)\        -
                               \"\)                  -
```

Copy to an Entry-Sequenced File (z/OS to HP NonStop)

This Process, submitted from Connect:Direct for HP NonStop, contains two steps:

- STEP1, which executes a FUP RUN TASK to purge a file. (Note the ! character in the PURGE command—it is required when issuing some FUP commands in BATCH mode.) The output from the FUP process will be written to spool location \$s.#FUP.
- STEP2, which executes a COPY to pull a file from z/OS to HP NonStop. The HP NonStop file is created NEW, so the desired file attributes are specified in SYSOPTS. The space attributes are explicitly declared rather than allowing them to default to the space information provided by the z/OS node. Checkpointing is disabled (CKPT=0K).

```
A110257 PROCESS      SNODE=CD.OS390
STEP1  RUN TASK      PGM=FUP PNODE
                    PARM=("/OUT $S.#FUP/"
                    "PURGE $B.FILERESO.A110257 !")
STEP2  COPY FROM    (SNODE DISP=SHR
                    DSN=DATA1.SEQ.A110257)
                    TO      (PNODE DISP=NEW
                    DSN=$B.FILERESO.A110257
                    SYSOPTS=("SET TYPE E"
                    "SET CODE 0"
                    "SET REC 4000"
                    "SET BLOCK 4096"
                    "SET EXT(32,32)"
                    "SET MAXEXTENTS 128"
                    "SET XLATE ON")
                    CKPT=0K
```

Create a Code 101 File (z/OS to HP NonStop)

This Process, submitted from the z/OS node, copies a file from z/OS and creates a type 101 EDIT file. Space attributes are allowed to default. Checkpointing is disabled.

```
TEST  PROCESS      SNODE=CD.HPNONSTOP
STEP1 COPY FROM    (PNODE DISP=SHR
                    DSN=DATA1.VSAME001.DATA)
                    TO      (SNODE DISP=NEW
                    DSN=$WORK02.HPDATA.UNST1
                    SYSOPTS=\\'SET TYPE U' \\|
                               \\SET CODE 101' \\|
                               \\SET XLATE ON' \" \\)
                    CKPT=0K
```

Create a Code 0 Oddunstructured File (z/OS to HP NonStop)

This Process, submitted from the z/OS node, copies a file from z/OS and creates a type 0 oddunstructured file. Space attributes are allowed to default. Checkpointing is disabled, as is translation.

```
TEST  PROCESS      SNODE= CD.HPNONSTOP
STEP1 COPY FROM   (PNODE DISP=SHR           -
                  DSN= DATA1.VSAME001.DATA) -
                  TO      (SNODE DISP=NEW           -
                  DSN=$WORK02.HPDATA.UNST1      -
                  SYSOPTS=\"'SET TYPE U' \ \ \ -
                  \ 'SET ODDUNSTR' \ \ \ -
                  \ 'SET XLATE OFF' \" \) -
                  CKPT=0K
```

Copy a Sequential File from a z/OS Node to an HP NonStop Node

The following Process transfers a sequential file from z/OS to HP NonStop. The Process is submitted on the z/OS node. Because the parameter SET EXT (50 50) is specified, the HP NonStop file system allocates 50 pages (one page=2048 bytes) to the primary extent and 50 pages to all secondary extents. Using the default for MAXEXTENTS results in a file with a capacity of 16 x 50 x 2048, or 1638400 bytes.

```
HPNONSTP PROCESS      PNODE=CD.OS390.DALL           -
                   SNODE=CD.BILL             -
                   SNODEID=(127.202,WILLIE) -
STEP01  COPY FROM   (PNODE DSN=SMITH.FDATA   -
                   DISP=SHR)               -
                   TO      (DSN=' $C.ROGER.TESTSND' -
                   SYSOPTS=\"'SET EXT(50 50)' \" \ -
                   DISP=NEW)
```

Copy a File Submitted from z/OS to HP NonStop

The following Process, submitted on a Connect:Direct for z/OS node, copies a data set from z/OS to a remote node on an HP NonStop EXPAND network. The SNODEID field passes the userid 147.200 to the HP NonStop, along with the password MONEY.

The userid and password are validated before the Connect:Direct for z/OS system copies the JSMITH.DATA file into the TESTOUT file in the \$B.JOHN volume on the \TSCIEXT system.

The following Process illustrates copying a file submitted from z/OS to HP NonStop.

```
GENSEND1 PROCESS      SNODE=ACCT.JOHN           -
                   SNODEID=(147.200,MONEY) -
STEP01  COPY FROM   (PNODE                     -
                   DSN=JSMITH.DATFILE        -
                   DISP=SHR)               -
                   TO      (SNODE             -
                   DSN='\TSCIEXT.$B.JOHN.TESTOUT' -
                   DISP=NEW                 -
                   SYSOPTS=\"'SET TYPE E' \ \ \ -
                   \xd5 SET XLATE ON' \" \)
```

Enclose the data set name in single quotation marks.

Copy a File from HP NonStop on an EXPAND Network to z/OS

This Process is submitted on the z/OS node to copy a file from an HP NonStop node on an EXPAND network to a z/OS node. The Connect:Direct for HP NonStop data transformation facility (DTF) (or server) is on system \SYSTEM, but the file is being copied from system \SYSEXT.

Enclose the data set name in single quotation marks, because the Process is submitted from a Connect:Direct for z/OS node.

```

GENSEND2  PROCESS      SNODE=CD.SMITH          -
          SNOEID=(149.205,WILLIE)
STEP01    COPY FROM    (DSN=' \SYSEXT.$A.ABC.TESTOUT' SNODE -
          DISP=SHR)
          TO            (DSN=JSMITH.TESTDATA PNODE -
          DISP=RPL)

```

Use FUP in a Process Submitted on z/OS to Delete a File on HP NonStop

In this Process, the FUP utility is used to delete an HP NonStop file. The Process is submitted on z/OS.

```

PART0003  PROCESS      SNODE=HPNONSTOP.CD
STEP1     RUN TASK     (PGM=FUP
          SYSOPTS="/OUT $$.#SFUP01/PURGE $USER.FILE01.* !" -
          SNODE )

```

Use IBM Connect:Direct to Allocate a Partitioned File on a Single System (z/OS to HP NonStop)

In this Process, the IBM Connect:Direct system is used to allocate an HP NonStop partitioned file. All partitions reside on the same system. The Process is submitted on z/OS.

```

PART0001  PROCESS      SNODE=HPNONSTOP.CD
STEP1     COPY FROM    (PNODE
          DSN=DATA1.TESTFILE.SEQ.FB80L
          DISP=SHR
          )
          TO            (SNODE
          DSN=$A.TESTFILE.PCODE0
          DISP=NEW
          SYSOPTS="\ 'SET CODE 0'
          \ 'SET TYPE U'
          \ 'SET PART (1,$B,10,5)'
          \ 'SET PART (2,$C,10,5)'
          \ 'SET MAXEXTENTS 16'
          \ 'SET XLATE ON'"
          )

```

SYSOPTS Syntax Conventions (z/OS to HP NonStop)

This Process will copy a file from the z/OS node to the HP NonStop node. Because the Process is submitted from the z/OS node, syntax conventions must follow those established for z/OS. In this example, bracketing backslashes are used to continue a string containing special characters across multiple lines.

```

SYSOPTS   PROCESS      PNODE=SC.OS390.NODE1
STEP01    COPY FROM    (SNODE=TSCI.HPNONSTOP
          SNOEID=(157.214 ABLE1)
          DSN=DATA1.SMALLER
          UNIT=SYSDA)
          TO            (DSN=$C.ACCTPROC.HPNONSTOP
          SYSOPTS="\ 'SET EXT(10,10)\
          \ ,XLATE ON\
          \ ,TYPE E\
          \ ,REC 100'"
          DISP=RPL)

```

Multiple SET commands can also be specified as follows:

```

SYSOPTS  PROCESS      PNODE=SC.OS390.NODE1      -
          SNODE=TSCI.HPNONSTOP      -
          SNODEID=(157.214 ABLE1)
STEP01   COPY FROM   (DSN=DATA1.SMALLER      -
                    UNIT=SYSDA)           -
          TO         (DSN=$C.ACCTPROC.HPNONSTOP -
                    SYSOPTS="'SET EXT(10,10)\ \ | | -
                    \ SET XLATE ON\ \ | | -
                    \ SET TYPE E\ \ | | -
                    \ SET REC 100'\ \ -
                    DISP=RPL)

```

Copy Files Between z/OS and UNIX

This Process copies a sequential file from a z/OS node to a UNIX node (STEP1) and back to the z/OS node (STEP2). It then submits itself to run again (STEP3).

```

CPY01   PROCESS      SNODE=den34      -
          SNODEID=(test1,propty2)
STEP1   COPY        FROM (PNODE DSN=TSTFL1.IN.FILE01 DISP=SHR) -
          TO         (SNODE DSN='out01' DISP=SHR)
STEP2   COPY        TO (PNODE DSN=TSTFL1.IN.FILE01 DISP=SHR) -
          FROM (SNODE DSN='out01' DISP=SHR)
          IF (STEP1 EQ 0) THEN
          RUN TASK (PGM=DMNOTFY2,      -
                  PARM=(FAIL,TSTFL1.IN.FILE01,%USER)) -
                  PNODE
                  EXIT
          EIF
STEP3   SUBMIT      DSN=TSTFL1.PROCESS.MVS.TO.UNIX(CPY01) -
                  SUBNODE=PNODE CASE=YES

```

This example shows a file copy from a z/OS node to a UNIX node using SYSOPTS. The UNIX DSN and SYSOPTS strings are in lower case and are enclosed in double quotation marks.

```

BENCHM1 PROCESS      SNODE=ky1a      -
          SNODEID=(barry1,322red)
STEP1   COPY        FROM (PNODE DSN=TEST.TESTFILE.BENCH.M1 DISP=SHR -
                    CKPT=1M -
                    COMPRESS EXTENDED -
          TO         (SNODE DSN='benchm1' DISP=MOD -
                    SYSOPTS=":datatype=text:strip.blanks=no:")

```

Copy a File from z/OS to VM

This Process copies a z/OS data set to a file on the 191 disk of VM user IVVB. If the file exists, the IBM Connect:Direct system replaces it. If the file does not exist, the IBM Connect:Direct system creates it as indicated by the DISP=RPL parameter.

The following Process illustrates copying a file from z/OS to VM.

```

ZOSTOVM PROCESS      SNODE=CD.VM.NODE1 -
          CLASS=10
STEP01   COPY FROM   (DSN=SMITH.FDATA      -
                    DISP=SHR -
                    UNIT=SYSDA)           -
          TO         (DSN='TEMP1 OUTPUT'   -
                    LINK=(IVVB,WIVVB,W,191) -
                    DISP=(RPL))

```

Copy a z/OS PDS to a Set of Files on VM

The COPY statement in this example copies a PDS, excluding all members with names beginning with the characters DM, to files on VM with file names corresponding to the member names on z/OS. The file type is ACCTDATA.

```
STEP01 COPY FROM (DSN=OS390.PDS.ACCT -
                  DISP=SHR -
                  EXCLUDE=(DM*) -
                  UNIT=SYSDA) -
            TO (DSN='* ACCTDATA' -
               LINK=(PEI,WPEI,W,300) -
               DISP=(RPL))
```

Copy a z/OS File to Tape on VM

This Process copies a disk file from a Connect:Direct for z/OS node to tape at the Connect:Direct for VM node. If the file exists, the IBM Connect:Direct system replaces it. If the file does not exist, the IBM Connect:Direct system creates it as indicated by the DISP=RPL parameter.

```
TAPE PROCESS SNODE=CD.VM.NODE1 -
STEP01 COPY FROM (DSN=CHELSEN.DATA.FILE -
                  DISP=SHR) -
            TO (DSN=TAPE.DATA.FILE -
               UNIT=TAPE -
               LABEL=(1,SL,EXPDT=92067) -
               SNODE -
               DISP=RPL)
```

Copy a z/OS File to Spool on VM

This Process copies a disk file from a Connect:Direct for z/OS node to spool at a Connect:Direct for VM node.

```
READER PROCESS SNODE=CD.VM.NODE1 -
STEP01 COPY FROM (DSN=RRT.SALES.DATA -
                  DISP=OLD) -
            TO (DSN='!SPOOL VKK SALES DATA' -
               DCB=(DSORG=PS,RECFM=F,LRECL=80,BLKSIZE=80))
```

Use the SYSOPTS Parameter (z/OS to OpenVMS)

This Process shows how to specify the SYSOPTS parameter with the MOUNT and PROTECTION keywords.

When one or more SYSOPTS parameters are specified and they continue across several lines, bracketing backslashes (\) and the double bar (||) concatenation characters are required.

The entire SYSOPTS string must be enclosed in double quotation marks. IBM Connect:Direct syntax requires using backslashes to continue the SYSOPTS over multiple lines when the Process is submitted from a z/OS node.

```
SYSOPTS3 PROCESS SNODE=CD.VMS.NODE NOTIFY=CDA1 -
STEP01 COPY FROM (DSN=SMITH.CNTL(IEBGENER) -
                  PNODE) -
            TO (DSN='MSA0:TAPE.DAT' -
               DISP=RPL SNODE -
               SYSOPTS=\"MOUNT='MSA0:/NOLABEL' \ || -
                       \ NODISMOUNT \ || -
                       \ PROTECTION='S:E,O:E,G:E,W:E' \"\)
```

Copy a File from z/OS to Microsoft Windows

This Process copies a file from a z/OS node to a Microsoft Windows node. The DSN (data set name) for the Microsoft Windows file uses the Universal Naming Convention (UNC) to specify the computer name \WIN-NODE and the share name \ROOT_D. This can be used as an alternative to specifying a drive letter such as C:\, and must be used when copying a file to a remote computer where the IBM Connect:Direct server is not running. The Connect:Direct for Microsoft Windows node, WIN.1200.TCP, must have access to the share, \ROOT_D, for the copy operation to succeed. The SYSOPTS parameter DATATYPE(TEXT) indicates that the data contains text (rather than binary data) and EBCDIC to ASCII translation will be performed.

```

TONT1  PROCESS          SNODE=WIN.1200.TCP          -
                                HOLD=NO                          -
                                RETAIN=NO                         -
STEP1  COPY  FROM      (DSN=TEST.DEL.DATA99)                      -
                                TO  (DSN=' \WIN-NODE\ROOT_D\USERS\TEST1\DATA1.TXT' -
                                SYSOPTS="DATATYPE(TEXT)"          -
                                DISP=(RPL))                      -

```

The following Process is a variation on the previous example. In this example, the data set names (DSN) are defined as symbolic variables in the COPY statement (&DSN1 and &DSN2), and are resolved at the time the process is submitted. This process also uses the STARTT parameter to specify a day of the week and time when the process will execute, and the RETAIN=YES parameter to indicate that the Process should stay in the TCQ and execute again at the next scheduled start time (STARTT). This Process will execute automatically each Monday at 3:00 a.m.

```

TONT1  PROCESS          SNODE=WIN.1200.TCP          -
                                HOLD=NO                          -
                                RETAIN=YES                       -
                                STARTT=(MONDAY,03:00)           -
                                &DSN1='TEST.DEL.TEST99'         -
                                &DSN2=' \WIN-NODE\ROOT_D\USERS\TEST1\DATA1.TXT' -
STEP1  COPY  FROM      (DSN=&DSN1)                          -
                                TO  (DSN=&DSN2)                  -
                                SYSOPTS="DATATYPE(TEXT)"        -
                                DISP=(RPL))                      -

```

Copy a File from Connect:Direct for z/OS to Connect:Direct for i5/OS

In this example COPY statement, the member FILEA in PDS USER.TESTLIB on the z/OS system is copied to the member TEST in the TESTFILES/PROCLIB file on the IBM Connect:Direct i5/OS node. If the member TEST already exists in the file TESTFILES/PROCLIB on the IBM Connect:Direct i5/OS node, it is replaced by this version.

```

/* COPY FROM: SOURCE DATA SET ATTRIBUTES (z/OS) */
STEP01  COPY  FROM      (DSN=USER.TESTLIB(FILEA)                -
                                PNODE                             -
                                DISP=SHR                         -
                                )                                -
/* COPY TO: DESTINATION DATA SET ATTRIBUTES (i5/OS) */
                                TO  (SNODE                         -
                                DSN='TESTFILES/PROCLIB(TEST)'    -
                                DISP=RPL                         -
                                SYSOPTS="\\"                     -
                                        \TYPE(MBR)\              -
                                        \TEXT('COPIED FROM FILEA')\ -
                                        "\\")                     -
                                )

```

Copy a File From z/OS to i5/OS

In this example, the member FILEA in PDS USER.TESTLIB on a z/OS system is copied to the member TEST in the TESTFILES/PROCLIB file on a Connect:Direct for i5/OS node. If the member TEST already exists in the file TESTFILES/PROCLIB on the Connect:Direct for i5/OS node, it is replaced by this version.

```
/* COPY FROM: SOURCE DATA SET ATTRIBUTES (z/OS) */
STEP01 COPY FROM (DSN=USER.TESTLIB(FILEA) -
                  PNODE -
                  DISP=SHR -
                  ) -
/* COPY TO: DESTINATION DATA SET ATTRIBUTES (i5/OS) */
TO (SNODE -
    DSN='TESTFILES/PROCLIB(TEST)' -
    DISP=RPL -
    SYSOPTS="\\" -
            \TYPE(MBR)\ -
            \TEXT('COPIED FROM FILEA')\ -
            "\" -
    )
```

Copy DBCS Data Sets Using Translation Tables in z/OS

In this example consisting of six COPY steps, six different translation tables are being used to convert from one format to another while each file is being transferred from the PNODE to the SNODE. Each translation table is specified using the SYSOPTS DBCS parameter in the TO clause.

```
DBCSTST1 PROCESS SNODE=CD.MAINFRAME -
SNODEID=(USERID,PASSWRD) -
*****
* DBCS CHINESE NEW HOST CODE TO CHINESE BIG 5 (NHCXBG5)
*****
STEP01 -
COPY TO ( SNODE -
          DSN=CD.O.DBCS.NHCXBG5 -
          DISP=(RPL,CATLG) -
          UNIT=SYSDA -
          DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS) -
          SPACE=(254,(1000,100)) -
          SYSOPTS="DBCS=NHCXBG5" -
        ) -
FROM ( PNODE -
       DSN=CD.DBCS.NHC -
       DISP=SHR -
     ) -
*****
* DBCS TABLE BG5XNHC
*****
STEP02 -
COPY TO ( SNODE -
          DSN=CD.O.DBCS.BG5XNHC -
          DISP=(RPL,CATLG) -
          UNIT=SYSDA -
          DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS) -
          SPACE=(254,(1000,100)) -
          SYSOPTS="DBCS=BG5XNHC" -
        ) -
FROM ( PNODE -
       DSN=CD.DBCS.BG5XNHC -
       DISP=SHR -
     ) -
*****
* DBCS TABLE EBCXKSC
*****
STEP03 -
COPY TO ( SNODE -
          DSN=CD.O.DBCS.EBCXKSC.SOSI -
          DISP=(RPL,CATLG) -
          UNIT=SYSDA -
          DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS) -
          SPACE=(254,(1000,100)) -
          SYSOPTS="DBCS=(EBCXKSC,0E,0F)" -
        ) -
FROM ( PNODE -
```

```

          DSN=CD.DBCS.EBCXKSC.FILE.SOSI -
          DISP=SHR -
        )
*****
* DBCS TABLE KSCXEBC
*****
STEP04
COPY TO ( SNODE -
          DSN=CD.0.DBCS.KSCXEBC -
          DISP=(RPL,CATLG) -
          UNIT=SYSDA -
          DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS) -
          SPACE=(254,(1000,100)) -
          SYSOPTS="DBCS=KSCXEBC" -
        )
FROM ( PNODE -
        DSN=CD.0.DBCS.EBCXKSC.SOSI -
        DISP=SHR -
      )
*****
* DBCS TABLE EBCXKPC
*****
STEP05
COPY TO ( SNODE -
          DSN=CD.0.DBCS.EBCXKPC.NOSO -
          DISP=(RPL,CATLG) -
          UNIT=SYSDA -
          DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS) -
          SPACE=(254,(1000,100)) -
          SYSOPTS="DBCS=(EBCXKPC,00,00)" -
        )
FROM ( PNODE -
        DSN=CD.DBCS.EBCXKPC.NOSO -
        DISP=SHR -
      )
*****
* DBCS TABLE KPCXEBC
*****
STEP06
COPY TO ( SNODE -
          DSN=CD.0.DBCS.KPCXEBC -
          DISP=(RPL,CATLG) -
          UNIT=SYSDA -
          DCB=(RECFM=VB,LRECL=254,BLKSIZE=4096,DSORG=PS) -
          SPACE=(254,(1000,100)) -
          SYSOPTS="DBCS=KPCXEBC" -
        )
FROM ( PNODE -
        DSN=CD.0.DBCS.EBCXKPC.NOSO -
        DISP=SHR -
      )

```

MBCS Conversion During z/OS to UNIX Copy

The following is an example for copying a z/OS file to a UNIX system and converting the IBM-943 code set on the z/OS system to the SJIS code set on the UNIX system.

```

COPY FROM (PNODE DISP=SHR -
          DSN=TEST1.MBCS.IBM943 -
          SYSOPTS="CODEPAGE=(IBM-943,UTF-8)" ) -
TO (SNODE DISP=RPL -
    DSN='/home/unix3500/cab1\ || -
    \ /CD/TestData/Source/codepage/Sjis390.txt' \ -
    sysopts=":codepage=(UTF-8,SJIS):datatype=binary" )

```

First, the IBM-943 code set data is converted to UTF-8 on the z/OS node. Then, as specified in the TO clause, the UTF-8 encoded data is transferred to the UNIX node, which converts the data to the SJIS code set and writes a UNIX file.

MBCS Conversion During Microsoft Windows to z/OS Copy

The following is an example for copying a Microsoft Windows file to a z/OS system and converting the 932 code set on Microsoft Windows to the IBM-943 code set on the z/OS system.

```
FROMNT COPY TO (PNODE DISP=(,CATLG) -
                UNIT=SYSDA -
                VOL=SER=USER01 -
                SPACE=(TRK,(1,1)) -
                DCB=(DSORG=PS,RECFM=U,BLKSIZE=24000) -
                DSN=TEST2.MBCS.LMORL1.SJIS001 -
                SYSOPTS="CODEPAGE=(UTF-8,IBM-943)" ) -
FROM (SNODE DISP=SHR -
      file='I:\Lcm\CD\TestData\Source\SjisCodepage.txt' -
      sysopts="codepage(932,65001) datatype=binary" )
```

As specified in the FROM clause, the 932 code set data in the SjisCodepage.txt file is converted to the 65001 code set (the UTF-8 equivalent on Microsoft Windows) on the Microsoft Windows node. The 65001 encoded data is then transferred to the z/OS node, which converts the data to the IBM-943 code set and writes a z/OS sequential file.

Note that the z/OS output file is defined as having an undefined record format (RECFM=U). When an MBCS conversion is done for a file that is created on a z/OS receiving node, the RECFM for the file must be specified as either V (Variable), VB (Variable Block), or U. The results of an MBCS conversion can result in a change to the physical length of the data, and the amount of the change can vary, depending on the data. RECFM=F (Fixed) or FB (Fixed Block) cannot be used, because the final data length is not fixed.

MBCS Conversion During z/OS to z/OS Copy

The following is an example for copying from one z/OS node to another and converting the IBM-930 code set to IBM-1047 via UTF-8.

```
STEP01 COPY FROM (PNODE DISP=SHR -
                  DSN=TEST3.MBCS0001.IBM930 -
                  SYSOPTS="CODEPAGE=(IBM-930,UTF-8)" ) -
TO (SNODE DISP=(,CATLG) -
   UNIT=SYSDA SPACE=(CYL,(3,3)) -
   VOL=SER=USER01 -
   DCB=(RECFM=VB,LRECL=90,BLKSIZE=24000) -
   DSN=CHICAGO.MBCS0001.IBM1047 -
   SYSOPTS="CODEPAGE=(UTF-8,IBM-1047)" )
```

First, the IBM-930 code set data is converted to the UTF-8 encoding on the sending node. The UTF-8 encoded data is then transferred to the receiving node, which converts the UTF-8 data to the IBM-1047 code set and writes a z/OS sequential file.

The RECFM of the file is specified as Variable Format (RECFM=VB) to allow for a flexible output record length. Also, to allow for a possible increase in data length due to conversion, the LRECL of the receiving file must be larger than the LRECL of the sending file. In the example shown, the LRECL of the sending file is 80. For this particular MBCS conversion, the receiving file was successfully created by specifying LRECL as 90. Other conversions may require a larger value to avoid an SVSJ032I error during the Copy. If RECFM=VB, BLKSIZE for the output file must be at least as large as LRECL+4.

Copying a File from zOS to Microsoft Windows using Substitution in a Destination Path:

This example shows how to use symbolic substitution to specify Microsoft Windows path names in a COPY statement.

In this example, the z/OS data set TEST.DATASET is copied to the Microsoft Windows file STERLING\CD\CDWIN\TEMP\TEST.TXT

1. Create a batch command that signs on to IBM Connect:Direct, submits a Process that creates the variables, and signs off:

```
SIGNON USERID=(userid,)
  SUBMIT PROC=EXNTDIR      -
  &FROMDSN=TEST.DATASET  -
  &DIR1=CDWIN             -
  &DIR2=TEMP              -
  &FILENAME=TEST.TXT
SIGNOFF
```

This Process creates the following variables:

Variable	Value	Description
&FROMDSN	TEST.DATASET	The name of the source data set.
&DIR1	CDWIN	Third level of the destination path.
&DIR2	TEMP	Fourth level of the destination path.
&FILENAME	TEST.TXT	The destination file name.

2. Create the following Process to copy the file:

```
NTDIRPTH  PROCESS                -
          SNODE=STERLING.WINDOWS -
          &DIR1=,                 -
          &DIR2=,                 -
          &FILENAME=,            -
          SYMBOL &S1 = STERLING
          SYMBOL &S2 = CD
          SYMBOL &TODSN = '\\\\ || &S1 || \\ || &S2 || \\ ||
          &DIR1 || \\ || &DIR2 || \\ || &FILENAME || \\
STEP01    COPY FROM (PNODE
          DSN=&FROMDSN
          DISP=SHR)
          TO (DSN=&TODSN
          SYSOPTS="DATATYPE(TEXT)"
          DISP=(RPL))
```

This Process defines the following symbolic values:

Variable	Value	Description
&S1	STERLING	First level of the destination path.
&S2	CD	Second level of the destination path.
&TODSN	\\ \\ . . . &FILENAME \\	The full destination path.

The following table shows how the &TODSN variable resolves (two vertical bars [||] indicate concatenation).

Value	Resolves to . . .
\\ \\	\\

Value	Resolves to . . .
&S1	STERLING
\\	\
&S2	CD
\\	\
&DIR1	CDWIN
\\	\
&DIR2	TEMP
\\	\
&FILENAME	TEST.TXT
\\	'

As a result, the &TODSN variable resolves to

```
'\STERLING\CD\CDWIN\TEMP\TEST.TXT'
```

CODEPAGE Conversion During a File Copy (z/OS to Microsoft Windows)

The following example shows how to specify the CODEPAGE SYSOPTS parameter when copying a file from a U.S. or Canadian Connect:Direct for z/OS system (IBM-1047 codepage) to an ANSI Latin Connect:Direct for Microsoft Windows system (65001 codepage, which is the UTF-8 equivalent on Microsoft Windows).

```
CODEPGNT  PROCESS      SNODE=SHARE1.WINDOWS      -
          HOLD=NO
CODEPGCP  COPY FROM    (PNODE                                -
          DSN=NQORR1.DATAFILE.MIXED                          -
          DISP=SHR                                           -
          SYSOPTS="CODEPAGE=(IBM-1047,UTF-8)")                -
          CKPT=5M                                             -
          TO          (SNODE                                -
          DSN='C:\SRN_TESTDATA\MIXED.TXT'                    -
          DISP=RPL                                           -
          SYSOPTS="CODEPAGE(65001,1252)")
```

Create and Copy a LARGE Data Set (z/OS)

This COPY statement transmits a z/OS BASIC data set to a new LARGE data set on another z/OS system. The DSNTYPE parameter specifies that the receiving data set is created as a LARGE format sequential data set.

```
COPY FROM (DSN=BASIC.SOURCE) -
        TO (DSN=LARGE.DEST DSNTYPE=LARGE)
```

Pass JES Job Name, Number, and User as a Process Name

In this example, the COPYPROC Process takes an existing file on the primary node (TEST.INPUT.FILE) and copies it to the secondary node naming the new file CD.jobID.jobname.submitjobuserID.FILE. The resultant file name contains the job number, job name, and the ID of the user who submitted the job for easy identification.

```
SAMPLE PROCESS COPYPROC
COPYPROC PROCESS -
          SNODE=CD.SNODE -
          &VAR1=%JOBID -
          &JOBID=%JOBID -
          &JOBNM=%JOBNM -
          &JUSER=%JUSER -
*
VAR1 COPY FROM (PNODE DSN=TEST.INPUT.FILE -
              DISP=SHR ) -
              TO (SNODE -
                 DSN=CD.&VAR1..FILE -
                 DISP=(,CATLG) -
                 SPACE=(CYL,(5,5),RLSE) -
                 UNIT=SYSDA)
JOBID COPY FROM (PNODE DSN=TEST.INPUT.FILE -
               DISP=SHR ) -
               TO (SNODE -
                  DSN=CD.&JOBID..FILE -
                  DISP=(,CATLG) -
                  SPACE=(CYL,(5,5),RLSE) -
                  UNIT=SYSDA)
JOBNM COPY FROM (PNODE DSN=TEST.INPUT.FILE -
               DISP=SHR ) -
               TO (SNODE -
                  DSN=CD.&JOBNM..FILE -
                  DISP=(,CATLG) -
                  SPACE=(CYL,(5,5),RLSE) -
                  UNIT=SYSDA)
JUSER COPY FROM (PNODE DSN=TEST.INPUT.FILE -
               DISP=SHR ) -
               TO (SNODE -
                  DSN=CD.&JUSER..FILE -
                  DISP=(,CATLG) -
                  SPACE=(CYL,(5,5),RLSE) -
                  UNIT=SYSDA)
```

To initiate the file transfer, you can use any of the following SUBMIT statements:

```
SUBMIT PROC=COPYPROC,NEWNAME=%JOBNM
```

```
SUBMIT PROC=COPYPROC,NEWNAME=%JOBID,&VAR1=%JOBNM
```

```
SUBMIT PROC=COPYPROC,&JOBNM=%JOBNM,&JUSER=%JUSER
```

Use SUBMIT with Symbolic Substitution (z/OS to z/OS)

In the following examples, a source file at the CD.LA node is to be copied to the CD.NEWYORK node, using CD.DALLAS as a store-and-forward node. To do this, two Processes need to be defined.

PROCESS1 is submitted from CD.LA. PROCESS1 performs the following functions:

- STEP01 copies the source file from CD.LA to CD.DALLAS.
- STEP02 within the same Process submits PROCESS2.

PROCESS2 executes at CD.DALLAS and contains one step which copies the file received from CD.LA and forwards it to CD.NEWYORK.

The user submits a Process and does not pass symbolics to the Process. Values for symbolics to be resolved on the SNODE are contained within the Process. The Process on the PNODE passes symbolics to a Process submitted on the SNODE.

Symbolics for PROCESS2 are supplied from values coded in PROCESS1.

The operator at CD.LA issues the following IBM Connect:Direct command to initiate the file transfer.

```
SUB PROC = PROCESS1
```

PROCESS1 executes:

```
PROCESS1  PROCESS      PNODE=CD.LA          -
          PROCESS      SNODE=CD.DALLAS       -
          PROCESS      &DSN1=USER1.TEXT        -
          PROCESS      &DSN2=USER1.NEW.TEXT    -
          PROCESS      &PRTY=14                -
          PROCESS      NOTIFY=USER1            -
STEP01    COPY FROM    (DSN=&DSN1 DISP=SHR PNODE) -
          TO          (DSN=&DSN2 DISP=(NEW,CATLG) -
          SNODE)
STEP02    SUBMIT      DSN=USER1.PROCESS.LIB(PROCESS2) -
          PRTY=&PRTY -
          SUBNODE=SNODE -
          &DSN=&DSN2
```

PROCESS1 submits PROCESS2:

```
PROCESS2  PROCESS      PNODE=CD.DALLAS       -
          PROCESS      SNODE=CD.NEWYORK        -
STEP01    COPY FROM    (DSN=&DSN PNODE)        -
          TO          (DSN=USER1.NEW.TEXT1 DISP=SHR)
```

- PROCESS1 copies the file USER1.TEXT in LA to a file called USER1.NEW.TEXT at CD.DALLAS. Then it submits PROCESS2, which executes on the CD.DALLAS node because the SUBNODE parameter designates the SNODE, or CD.DALLAS, as the submitting node for PROCESS2.
- PROCESS2 copies the file ABC.NEW.TEXT at CD.DALLAS to file ABC.NEW.TEXT1 in CD.NEWYORK. The default DSN symbolic name for the PNODE is taken from the previous PROCESS1.

Use SUBMIT with the DSN Parameter (z/OS to z/OS)

In this example, symbolics for PROCESS2 are supplied by the operator submitting PROCESS1.

The operator at CD.LA issues the following IBM Connect:Direct 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 -
          PROCESS      &DSN1=&DSN1    -
          PROCESS      &DSN2=&DSN2    -
          PROCESS      &PRTY=14      -
          PROCESS      NOTIFY=A345    -
STEP01    COPY FROM    (DSN=&DSN1 DISP=SHR PNODE) -
          TO          (DSN=&DSN2 DISP=SHR SNODE)
STEP02    SUBMIT      DSN=A345.PROCESS.LIB(PROCESS2) -
          PRTY=&PRTY -
          SUBNODE=SNODE -
          &DSN=&DSN2
```

PROCESS1 submits PROCESS2:

```

PROCESS2  PROCESS      PNODE=CD.DALLAS      -
          SNODE=CD.NEWYORK
STEP01    COPY    FROM    (DSN=&DSN PNODE)    -
          TO      (DSN=A345.NEW.DATA1 DISP=SHR)

```

- PROCESS1 copies the file A345.DATA at CD.LA to a file called A345.NEW.DATA at the CD.DALLAS node, and then it submits PROCESS2, which executes on the CD.DALLAS node.
- PROCESS2 copies the file A345.NEW.DATA at the CD.DALLAS to the file A345.NEW.DATA1 at CD.NEWYORK.

Submit a Microsoft Windows Process from a UNIX Node

This Process copies a binary file from a Microsoft Windows node to a UNIX node. If the copy is successful, a **submit** of another process is performed on the CD.WIN node by setting the **subnode** parameter to SNODE.

```

proc1  process      snode=CD.WIN
          &file1="d:\estdat\frunix.cfg"
          &file2="/tdat/frnt.cfg"
copy1  copy    from  (file=&file1
                    sysopts="datatype(binary)"
                    snode)
          to      (file=&file2
                    pnode)
          if (copy1 eq 0) then
subpr1 submit  file="d:\estdat\nt2nwcp.cdp"
                    Subnode=snode
          eif
pend

```

Use SUBMIT at the Connect:Direct for HP NonStop Node

In this Process, STEP1 will execute FUP to purge \$B.FILERESO.FILEA. STEP2 and STEP3 will submit IBM Connect:Direct Processes at the HP NonStop node (PNODE).

```

SUBMIT  PROCESS      PNODE=CD.HP NONSTOP      -
          SNODE=CD.OS390.NODE
STEP1   RUN    TASK   (PGM=FUP          -
                    SYSOPTS=( '/OUT $$.#FUP/' , -
                    'VOLUME $B.FILERESO' , -
                    'PURGE FILEA ! '))
STEP2   SUBMIT  FILE=$B.DATA1.FILEB    -
          SUBNODE=PNODE
STEP3   SUBMIT  FILE=$B.DATA1.FILEC    -
          SUBNODE=PNODE

```

Use submit with the hold Parameter (UNIX to UNIX)

This Process shows how to submit another IBM Connect:Direct Process, named copy.cd, which resides on the **snode**. The Process will be held in the Hold queue until it is explicitly released by a **change process** command.

```

submit1  process      snode=unix.node
step01   submit      file=copy.cd
          subnode=snode
          hold=yes
pend

```

Use submit with the startt Parameter (UNIX to UNIX)

This Process shows how to submit another IBM Connect:Direct Process, named copy.cd, which resides on the **pnode**. The Process will be held in the Timer queue until it is automatically released at 11:30 p.m. on December 14, 2005.

```
submit1 process snode=unix.node
step01 submit file=copy.cd
        startt=(12/14/1999,23:30)
pend
```

Submit a Process to Run Upon Successful Completion of a Copy (HP NonStop to HP NonStop)

This Process copies USER1T.TEXT to \$B. HP NONSTOP.FILE at the SNODE (CD.TAN.NODE). If \$B. HP NONSTOP.FILE already exists, the file will be replaced. Upon completion of STEP01, \$B. HP NONSTOP.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=$B. HP NONSTOP.FILE DISP=RPL SNODE)
STEP02 SUBMIT DSN=$B. HP NONSTOP.FILE
          PRTY=5
          SUBNODE=SNODE
```

Submit a Process on OpenVMS

In this Process, STEP1 copies ACCT.VMSPROC to ACCTPROC.Sterling 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
```

Submit a Process That Copies a File from One UNIX Node to Another, Then to a Third UNIX Node

This example shows how to copy a file from your local node to a node in Dallas, which then copies 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)
          copy  to    (dsn=yourfile)
          submit  disp=(new,keep))
          submit  file=process2
          submit  subnode=snode
          submit  &dsn=yourfile
          pend

```

The copy statement in **process2** copies **yourfile** to the file **newfile** in Tampa.

```

process2  process  snode=tampa
copystep  copy  from  (dsn=&dsn)
          copy  to    (dsn=newfile)
          submit  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.

Submit a Process with Symbolics on VSE

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 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.

Submit a Process Using Symbolics on VMESA

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 IBM 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 for 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 for VM/ESA node and CD.NEWYORK is a Connect:Direct for z/OS 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.

Submit a Process from UNIX to z/OS to Connect:Enterprise Using BATCHID

The following examples submit a Process from Connect:Direct for UNIX to Connect:Direct for z/OS, which then executes an MB#ADD01 function on Connect:Enterprise.

The following is the Connect:Direct for UNIX Process:

```
set -v
ndmcli -x << E0J
submit
proc1 process snode=ip addr;portnum snodeid=(ROGER,ROGERpassword)
step1 submit file="$hlq.cd.PROCESS(MB#ADD01)"
      &FROMDSN=TEST.DATAFILE
      &RMTID=rmtid
      &BATCHID=TESTBATCH
      subnode=snode
pend;
```

The following is the z/OS Process called by the UNIX Process. This Process performs an MB#ADD01 on Connect:Enterprise:

```

MB#ADD01 PROCESS SNODE=CD.OS390 -
&NOTIFY=' ' /* Notify nobody */ -
&FROMDSN=' ' FROM.DSN.NAME' /* DATA SOURCE FILE NAME */ -
&FILETYP=TEXT /* DATA SOURCE FILE TYPE */ -
&PROFDSN='$hlq.CD.PROFILE' /* PROFILE DSN */ -
&PROFMEM=SUBADD01 /* EXTRACT PROFILE MEMBER */ -
&USER=%USER /* ORIGINATOR ID */ -
&SEQ=%NUM1 /* UNIQUE NUMBER */ -
&RMTID=RMTID /* RDX BATCH RMTID */ -
&MBNAME=E100 /* RDX NAME */ -
&LOCAPPL=MBICO /* RDX LOCAL APPLID */ -
&MBAPPL=SBLDUB74 /* RDX APPLID */ -
&LOGMODE=TESTLU62 /* RDX LOG MODE TABLE */ -
&USERID=USERID /* RDX USERID */ -
&PASSWRD=PASS /* RDX PASSWORD */ -
&BATCHID=BATCH.ID /* RDX BATCH ID */ -
&BATCHNO=0 /* RDX BATCH NUMBER */ -
&BUFSIZE=0 /* VTAM BUFFER SIZE */ -
*
SYMBOL &HLQ1=\CD.\ || &RMTID
SYMBOL &SEQ1=T\ || &SEQ
SYMBOL &ADSN=\&HLQ1\ || \.STOUT0.INFILE.\ || &SEQ1
SYMBOL &IDSN=\&HLQ1\ || \.STOUT0.SYSIN.\ || &SEQ1
SYMBOL &PDSN=\&HLQ1\ || \.STOUT0.SYSPRINT.\ || &SEQ1
SYMBOL &LDSN=\&HLQ1\ || \.STOUT0.LOGFILE.\ || &SEQ1
SYMBOL &SDSN=\&HLQ1\ || \.STOUT0.SNAPOUT.\ || &SEQ1
*
SYMBOL &BATCHID2=''\ || &BATCHID || '\'\
*
* COPY FILE FROM SOURCE NODE
*
MBA$COPY COPY FROM(SNODE DSN=&FROMDSN DISP=SHR) -
TO ( DSN=&ADSN DISP=(RPL,CATLG) -
TYPE=&FILETYP)
*
IF (MBA$COPY > 0) THEN
RUN TASK (PGM=DMIONOTE, -
PARM=(ADD, FAIL, "&FROMDSN", &RMTID, "&BATCHID2", -
MBA$COPY, &NOTIFY)) PNODE
EXIT
EIF
*
* CREATE THE TEMPORARY FILE FOR USE BY ADD
*
MBA$CREI RUN TASK (PGM=DMRTDYN -
PARM=(C'ALLOC', -
C" DSN=&IDSN" -
C' DISP=(NEW,CATLG)', -
C' SPACE=(TRK,(1,1))', -
C' DCB=(RECFM=FB,DSORG=PS,LRECL=80,BLKSIZE=800)', -
C'UNIT=SYSDA', -
F'-1', -
C'UNALLOC', -
C" DSN=&IDSN")) PNODE
*
IF (MBA$CREI > 0) THEN
RUN TASK (PGM=DMIONOTE, -
PARM=(ADD, FAIL, "&FROMDSN", &RMTID, "&BATCHID2", -
MBA$CREI, &NOTIFY)) PNODE
EXIT
EIF
*
MBA$CREP RUN TASK (PGM=DMRTDYN -
PARM=(C'ALLOC', -
C" DSN=&PDSN" -
C' DISP=(NEW,CATLG)', -
C' SPACE=(TRK,(1,1))', -
C' DCB=(RECFM=FBA,DSORG=PS,LRECL=133,BLKSIZE=2660)', -
C'UNIT=SYSDA', -
F'-1', -
C'UNALLOC', -
C" DSN=&PDSN")) PNODE
*
IF (MBA$CREP > 0) THEN
RUN TASK (PGM=DMIONOTE, -
PARM=(ADD, FAIL, "&FROMDSN", &RMTID, "&BATCHID2", -
MBA$CREP, &NOTIFY)) PNODE

```

```

/* GOTO MBA$DELI */
EXIT
EIF
*
MBA$CREL RUN TASK (PGM=DMRTDYN -
                   PARM=(C'ALLOC', -
                          C" DSN=&LDSN" -
                          C' DISP=(NEW,CATLG)', -
                          C' SPACE=(TRK,(1,1))', -
                          C' DCB=(RECFM=FB,DSORG=PS,LRECL=196,BLKSIZE=1960)', -
                          C' UNIT=SYSDA', -
                          F'-1', -
                          C'UNALLOC', -
                          C" DSN=&LDSN")) PNODE
*
IF (MBA$CREL > 0) THEN
  RUN TASK (PGM=DMIONOTE, -
           PARM=(ADD, FAIL, "&FROMDSN", &RMTID, "&BATCHID2", -
                 MBA$CREL, &NOTIFY)) PNODE
/* GOTO MBA$DELP */
EXIT
EIF
*
MBA$CRES RUN TASK (PGM=DMRTDYN -
                  PARM=(C'ALLOC', -
                         C" DSN=&SDSN" -
                         C' DISP=(NEW,CATLG)', -
                         C' SPACE=(TRK,(1,1))', -
                         C' DCB=(RECFM=FB,DSORG=PS,LRECL=196,BLKSIZE=1960)', -
                         C' UNIT=SYSDA', -
                         F'-1', -
                         C'UNALLOC', -
                         C" DSN=&SDSN")) PNODE
*
IF (MBA$CRES > 0) THEN
  RUN TASK (PGM=DMIONOTE, -
           PARM=(ADD, FAIL, "&FROMDSN", &RMTID, "&BATCHID2", -
                 MBA$CRES, &NOTIFY)) PNODE
/* GOTO MBA$DELP */
EXIT
EIF
*
* EXECUTE THE ADD TO CONNECT:Enterprise RDX
*
MBA$ST00 RUN TASK (PGM=DMSTOUT0, -
                  PARM=(C'ADD' -
                        C"&PROFDSN" -
                        C"&PROFMEM" -
                        C"&IDSN" -
                        C"&PDSN" -
                        C"&ADSN" -
                        C"&LDSN" -
                        C"&SDSN" -
                        C'&RMTID' -
                        C'&BATCHID2' -
                        C'&BATCHNO' -
                        C'&MBNAME' -
                        C'&LOCAPPL' -
                        C'&MBAPPL' -
                        C'&LOGMODE' -
                        C'&USERID' -
                        C'&PASSWRD' -
                        C'&BUFSIZE')) PNODE
*
IF (MBA$ST00 > 4) THEN
  RUN TASK (PGM=DMIONOTE, -
           PARM=(ADD, FAIL, "&FROMDSN", &RMTID, "&BATCHID2", -
                 MBA$ST00, &NOTIFY)) PNODE
ELSE
*
*
* NOTIFY ORIGINATOR OF SUCCESS
  RUN TASK (PGM=DMIONOTE, -
           PARM=(ADD, GOOD, "&FROMDSN", &RMTID, "&BATCHID2", -
                 MBA$ST00, &NOTIFY)) PNODE
EIF
*
* DELETE TEMPORARY FILES
* ...

```

Run a Job on the z/OS Node from a Process Submitted on the HP NonStop Node

In this Process submitted from the HP NonStop node, STEP1 will execute FUP to purge \$B.FILERESO.STEST. STEP2 will copy DATA1.SMALLER from the z/OS node to \$B.FILERESO.STEST at the HP NonStop node. Conditional logic (STEP3) is then used to check the completion code of STEP1. If the completion code is greater than 4, no further processing will occur. Otherwise STEP4 will execute DATA1.CNTL(IEFBR14A) at the z/OS node.

The Connect:Direct for HP NonStop system cannot execute the RUN JOB statement; however, the Connect:Direct for HP NonStop node as the PNODE can submit a Process to an z/OS or VSE node, and the SNODE can execute the RUN JOB.

RUN	PROCESS	PNODE=CD.HP NONSTOP	-
		SNODE=SS.CD.0S390	
STEP1	RUN TASK	(PGM=FUP	-
		PARM= ('/OUT \$S.#STEST/',	-
		'VOLUME \$B.FILERESO',	-
		'PURGE STEST')	
STEP2	COPY FROM	(DSN=DATA1.SMALLER SNODE	-
		DISP=SHR)	-
	TO	(DSN=\$B.FILERESO.STEST PNODE	-
		DISP=NEW)	
STEP	IF (STEP1 GT 4) THEN		
	EXIT		
	EIF		
STEP4	RUN JOB	(DSN=DATA1.CNTL(IEFBR14A)) SNODE	

Run a Job on the z/OS Node from a Process Submitted on the HP NonStop Node

In this Process submitted from the HP NonStop node, STEP1 will execute FUP to purge \$B.FILERESO.STEST. STEP2 will copy DATA1.SMALLER from the z/OS node to \$B.FILERESO.STEST at the HP NonStop node. Conditional logic (STEP3) is then used to check the completion code of STEP1. If the completion code is greater than 4, no further processing will occur. Otherwise STEP4 will execute DATA1.CNTL(IEFBR14A) at the z/OS node.

The Connect:Direct for HP NonStop system cannot execute the RUN JOB statement; however, the Connect:Direct for HP NonStop node as the PNODE can submit a Process to an z/OS or VSE node, and the SNODE can execute the RUN JOB.

RUN	PROCESS	PNODE=CD.HP NONSTOP	-
		SNODE=SS.CD.0S390	
STEP1	RUN TASK	(PGM=FUP	-
		PARM= ('/OUT \$S.#STEST/',	-
		'VOLUME \$B.FILERESO',	-
		'PURGE STEST')	
STEP2	COPY FROM	(DSN=DATA1.SMALLER SNODE	-
		DISP=SHR)	-
	TO	(DSN=\$B.FILERESO.STEST PNODE	-
		DISP=NEW)	
STEP	IF (STEP1 GT 4) THEN		
	EXIT		
	EIF		
STEP4	RUN JOB	(DSN=DATA1.CNTL(IEFBR14A)) SNODE	

Run a Job on the z/OS Node from a Process Submitted on the HP NonStop Node

In this Process submitted from the HP NonStop node, STEP1 will execute FUP to purge \$B.FILERESO.STEST. STEP2 will copy DATA1.SMALLER from the z/OS node to \$B.FILERESO.STEST at the HP NonStop node. Conditional logic (STEP3) is then used to check the completion code of STEP1.

If the completion code is greater than 4, no further processing will occur. Otherwise STEP4 will execute DATA1.CNTL(IEFBR14A) at the z/OS node.

The Connect:Direct for HP NonStop system cannot execute the RUN JOB statement; however, the Connect:Direct for HP NonStop node as the PNODE can submit a Process to an z/OS or VSE node, and the SNODE can execute the RUN JOB.

```

RUN    PROCESS      PNODE=CD.HP NONSTOP      -
        SNODE=SS.CD.OS390
STEP1  RUN TASK    (PGM=FUP                  -
        PARM= ('/OUT $S.#STEST/',          -
        'VOLUME $B.FILERESO',            -
        'PURGE STEST '))
STEP2  COPY FROM  (DSN=DATA1.SMALLER SNODE   -
        DISP=SHR)                          -
        TO        (DSN=$B.FILERESO.STEST PNODE -
        DISP=NEW)
STEP   IF (STEP1 GT 4) THEN
        EXIT
EIF
STEP4  RUN JOB    (DSN=DATA1.CNTL(IEFBR14A)) SNODE

```

Execute Commands on UNIX from a Process Submitted from HP NonStop

This Process shows how to initiate a Process from HP NonStop that executes commands at a UNIX node. The SYSOPTS string must be in the proper case for UNIX and enclosed in double quotation marks.

```

RUNJOB1 PROCESS SNODE=CD.v1200      -
        snodeid=(user,pswd)
UNIXJOB  RUN JOB  SYSOPTS="ls -l > outjob.tan" SNODE

```

Microsoft Windows Run Job Statement

This **run job** statement executes the program testwin.exe on a remote Microsoft Windows system.

```

jobstep  run job  snode (dsn=WINNT)
                sysopts="pgm(c:\winnt35\system32  estwin.exe)"

```

Microsoft Windows Run Job Statement

This **run job** statement executes the program testwin.exe on a remote Microsoft Windows system.

```

jobstep  run job  snode (dsn=WINNT)
                sysopts="pgm(c:\winnt35\system32  estwin.exe)"

```

Submit a Process with a RUN JOB on OpenVMS

The following example shows a Process that submits the command procedure TEST_RJOB.COM from the directory JSMITH.TEST on disk DUC4 on the SNODE (OpenVMS). The SYSOPTS, or system-specific parameters, consist of four parameters, each enclosed in single quotes. P1, P2, and P3 are parameters passed to the TEST_RJOB.COM command procedure. LOG displays the TEST_RJOB.COM commands as they execute to view for error and completion messages. The entire SYSOPTS is enclosed in double quotation marks.

```

RUNJOB1  PROCESS  SNODE=CD.VMS.SMITH
STEP01   RUN JOB  (DSN='DUC4:[JSMITH.TEST]TEST_RJOB.COM' PNODE   -
        SYSOPTS="P1='DEF' P2='123' P3='BR549' NOPRINT  -
        LOG='DUC4:[JSMITH.TEST]TEST.LOG'"

```

Submit a Process with a RUN JOB on OpenVMS

The following example shows a Process that submits the command procedure TEST_RJOB.COM from the directory JSMITH.TEST on disk DUC4 on the SNODE (OpenVMS). The SYSOPTS, or system-specific parameters, consist of four parameters, each enclosed in single quotes. P1, P2, and P3 are parameters passed to the TEST_RJOB.COM command procedure. LOG displays the TEST_RJOB.COM commands as they execute to view for error and completion messages. The entire SYSOPTS is enclosed in double quotation marks.

```
RUNJOB  PROCESS  SNODE=CD.VMS.SMITH
STEP01  RUN JOB  (DSN='DUC4:[JSMITH.TEST]TEST_RJOB.COM' PNODE  -
                 SYSOPTS="P1='DEF' P2='123' P3='BR549' NOPRINT  -
                 LOG='DUC4:[JSMITH.TEST]TEST.LOG'"
```

Print and Delete the Log File on Connect:Direct for OpenVMS

In this example, a command procedure is submitted to the SNODE for execution in the batch queue. No parameters are specified. The log file is printed and deleted because the default parameters of PRINT and NOKEEP are assumed. Note that the command procedure NOTIFY.COM is executed from the default login directory of the submitter.

```
RUNJ1  PROCESS  SNODE=CD.VAX
        RUN JOB  (FILE='NOTIFY.COM' SNODE)
```

Keep the Log File on Connect:Direct for OpenVMS

In this example, a command procedure executes on the VAX in the batch queue. Only the NOPRINT subparameter is specified, resulting in the log file being kept and named after the first (or only) file in the job.

```
RUNJ2  PROCESS  SNODE=CD.VAX
        RUN JOB  (FILE='NOTIFY.COM' SNODE) -
                 SYSOPTS="NOPRINT"
```

Print and Keep the Log File on Connect:Direct for OpenVMS

In this example, NOTIFY.COM executes on the SNODE. During Process execution, the parameter (P1) is passed to the command procedure. The log file is printed and kept. LOG is not specified because KEEP ensures that the output is saved.

```
RUNJ3  PROCESS  SNODE=CD.OS390
        RUN JOB  (FILE='NOTIFY.COM' SNODE) -
                 SYSOPTS="P1='SUCCESS' PRINT KEEP"
```

Run Job on OpenVMS

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'"
```

Run a Job on UNIX from a Process Submitted from Another UNIX Node

This Process shows how to initiate a Process that executes commands at another UNIX node. The SYSOPTS string must be enclosed in double quotation marks.

```
proc2  process  snode=unix.node
step01 run job  snode
        sysopts="ls > /abc/file/user/us1/lsout; ls -lax"
pend
```

Run a Job on UNIX from a Process Submitted from Another UNIX Node

This Process shows how to initiate a Process that executes commands at another UNIX node. The SYSOPTS string must be enclosed in double quotation marks.

```
proc2  process  snode=unix.node
step01 run job  snode
        sysopts="ls > /abc/file/user/us1/lsout; ls -lax"
pend
```

Run a Job on z/OS from a Process Submitted on UNIX

This Process shows how to initiate a Process from UNIX to run a job at an z/OS node. The DSN string must be in uppercase characters to satisfy z/OS syntax requirements.

```
proc2  process  snode=OS390.node
        snodeid=(user01,pswd01)
step01 run job  (dsn=SRCDATA.SET(TEST))
        snode
pend
```

Run a Job on Microsoft Windows from a Process Submitted on UNIX

This Process shows how to initiate a Process from UNIX that executes commands at a Microsoft Windows node. In this example, the command **dir** is issued on the node with the output redirected to a file called list.out. When issuing a console command, such as **del**, **dir**, or **move**, specify the command in the **sysopts** parameter exactly the way it is issued at the command prompt directly on the Microsoft Windows system.

```
ntsub1 process  snode=cd.win
        retain=yes
        prty=yes
runj01 run job  snode
        sysopts="cmd(dir d:\users\jdoe1\*.*
>>d:\users\jdoe1\list.out)"
pend;
```

Example UNIX Run Job

In this example, the application **myjob** is submitted to the **pnode** system.

```
jobstep run job  sysopts="myjob param1"
```

RUN JOB Facility (VM to VM)

This Process sends the file named SIGNON CDOP in PUN format to the reader for CDA5A. This Process can be used to transmit jobs to be processed by VMBATCH.

```
RUNJOB2  PROCESS  SNODE=CD.VM.NODE NOTIFY=CDA8
STEP01   RUN JOB  (SNODE DSN='SIGNON CDOP' -
                  LINK=(CDA6,RCDA6,RR,191) BATCHID=CDA5A)
```

RUN JOB Facility (VM to VM)

This Process sends the file named SIGNON CDOP in PUN format to the reader for CDA5A. This Process can be used to transmit jobs to be processed by VMBATCH.

```
RUNJOB2  PROCESS  SNODE=CD.VM.NODE NOTIFY=CDA8
STEP01   RUN JOB  (SNODE DSN='SIGNON CDOP' -
                  LINK=(CDA6,RCDA6,RR,191) BATCHID=CDA5A)
```

Example VMESA Run Job

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 -
                  )
```

VSE Run Job Statement

The example VSE 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
```

VSE Run Job Statement

The example VSE 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
```

Submit a Job to the z/OS Internal Reader

You can submit a RUN JOB Process from either the SNODE or the PNODE. If you specify PNODE on the RUN JOB, the job is submitted to the primary node; if you specify SNODE, the job is submitted to the secondary node. In both cases, the job is submitted to the internal reader if both nodes are running Connect:Direct for z/OS.

The node that you submit the Process to is the PNODE or Process control node by definition; the other node involved in the Process is the SNODE.

In the following example, the Process, PROC1, is submitted from the PNODE, CDA. PROC1 then instructs the SNODE, CDB, to execute the job titled JOB123 in the library JOB.STREAM.LIB. The job is submitted to the z/OS internal reader on CDB.

The data set specified in the RUN JOB statement must exist on CDB.

```
PROC1  PROCESS  SNODE=CDB
STEP01 RUN JOB  (DSN=JOB.STREAM.LIB(JOB123) SNODE)
```

Alternatively, you can be signed on to CDA and submit the following RUN JOB Process from CDA. In this case, the data set specified in the RUN JOB must exist on CDA (PNODE).

In this example, the job contained in JOB.STREAM.LIB(JOB123) is submitted to the z/OS internal reader on the PNODE system.

```
PROC1  PROCESS  SNODE=CDB
STEP01 RUN JOB  (DSN=JOB.STREAM.LIB(JOB123) PNODE)
```

The JCL must exist on the node where you want it to execute.

Submit a Job to the z/OS Internal Reader

You can submit a RUN JOB Process from either the SNODE or the PNODE. If you specify PNODE on the RUN JOB, the job is submitted to the primary node; if you specify SNODE, the job is submitted to the secondary node. In both cases, the job is submitted to the internal reader if both nodes are running Connect:Direct for z/OS.

The node that you submit the Process to is the PNODE or Process control node by definition; the other node involved in the Process is the SNODE.

In the following example, the Process, PROC1, is submitted from the PNODE, CDA. PROC1 then instructs the SNODE, CDB, to execute the job titled JOB123 in the library JOB.STREAM.LIB. The job is submitted to the z/OS internal reader on CDB.

The data set specified in the RUN JOB statement must exist on CDB.

```
PROC1  PROCESS  SNODE=CDB
STEP01 RUN JOB  (DSN=JOB.STREAM.LIB(JOB123) SNODE)
```

Alternatively, you can be signed on to CDA and submit the following RUN JOB Process from CDA. In this case, the data set specified in the RUN JOB must exist on CDA (PNODE).

In this example, the job contained in JOB.STREAM.LIB(JOB123) is submitted to the z/OS internal reader on the PNODE system.

```
PROC1  PROCESS  SNODE=CDB
STEP01 RUN JOB  (DSN=JOB.STREAM.LIB(JOB123) PNODE)
```

The JCL must exist on the node where you want it to execute.

Run a Job on the i5/OS Node from a Process Submitted on the z/OS Node

This example is submitted on z/OS to run a job on i5/OS. DSN is necessary when submitting from z/OS. The contents of the SYSOPTS parameter define the program to run on the i5/OS node.

```
RJPROC01  PROCESS  SNODE=CD2200          -
          SNODEID=(USER1,PASSWD1)
STEP0001  RUN JOB  (DSN=AS400)  SNODE          -
          SYSOPTS="\ "          -
                  \CMD(\       -
                  \CALL\      -
                  \PGM(KRM/KJOBST)\ -
                  \)\         -
                  \"\         -
```

Execute Commands on UNIX from a Process Submitted from z/OS

This Process shows how to initiate a Process from z/OS that executes commands at a UNIX node. The SYSOPTS string must be in the proper case for UNIX and enclosed in double quotation marks.

```
PROC2     PROCESS  SNODE=UNIX.NODE
STEP01    RUN JOB  (DSN=UNIX)          -
          SNODE          -
          SYSOPTS="/company/payroll/monthly/jan.exe"
```

Use Run Job to Submit a Job on an i5/OS Node

In this example z/OS or z/OS Process, the job JANPMTS is submitted on the i5/OS node.

```
JOBSTEP   PROCESS  SNODE=CDAS400
STEP01    RUN JOB  SNODEID(USER01,PSWD01)
          (DSN=AS400) SNODE
          SYSOPTS="\ "
                  \CMD(\
                  \CALL PGM(INV/JANPMTS)\
                  \)\
                  \"\
```

Submit a Run Job on i5/OS from z/OS

In this example z/OS Process, the job JANPMTS is submitted to the i5/OS node.

```
JOBSTEP   PROCESS  SNODE=CDAS400
STEP01    RUN JOB  SNODEID(USER01,PSWD01)
          (DSN=AS400) SNODE
          SYSOPTS="\ "
                  \CMD(\
                  \CALL PGM(INV/JANPMTS)\
                  \)\
                  \"\
```

Run FUP (Sterling Connect:Direct for HP NonStop Run Task)

This Process starts FUP and copies a disk file to the spooler location \$S.#SPL2. Output from the FUP command (success/failure) is sent to \$S.#SPL1.

Note: This Process will only work in SNA environments.

```

PROC1  PROCESS    SNODE=SYSCLX
STEP1  RUN TASK    (PGM=FUP
                  PARM=(' /NAME $FP,OUT $S.#SPL1/'
                        , ' COPY $A.PROCVOL.ACCT,$S.#SPL2' )
                  )
PNODE

```

Run FUP (Sterling Connect:Direct for HP NonStop Run Task)

This Process starts FUP and copies a disk file to the spooler location \$S.#SPL2. Output from the FUP command (success/failure) is sent to \$S.#SPL1.

Note: This Process will only work in SNA environments.

```

PROC1  PROCESS    SNODE=SYSCLX
STEP1  RUN TASK    (PGM=FUP
                  PARM=(' /NAME $FP,OUT $S.#SPL1/'
                        , ' COPY $A.PROCVOL.ACCT,$S.#SPL2' )
                  )
PNODE

```

Run FUP (Sterling Connect:Direct for HP NonStop Run Task)

This Process starts FUP and copies a disk file to the spooler location \$S.#SPL2. Output from the FUP command (success/failure) is sent to \$S.#SPL1.

Note: This Process will only work in SNA environments.

```

PROC1  PROCESS    SNODE=SYSCLX
STEP1  RUN TASK    (PGM=FUP
                  PARM=(' /NAME $FP,OUT $S.#SPL1/'
                        , ' COPY $A.PROCVOL.ACCT,$S.#SPL2' )
                  )
PNODE

```

Submit a Process with a Connect:Direct for OpenVMS RUN TASK from an HP NonStop Node

This Process, submitted from the Connect:Direct for HP NonStop node, will issue a command to invoke a DCL command procedure. Output will be directed to the terminal. Upon successful execution of the command procedure, the terminal of the specified user will beep.

```

VAXRUN  PROCESS    PNODE=CD.HPNONSTOP
STEP1    RUN TASK    (PGM=VMS) SNODE
                  SYSOPTS="OUTPUT = '_NDC31'
                  CMD = 'DIR DUC4:[USER1.DATA]*.DAT'
                  CMD = 'REPLY/USER=USER1/BELL SUCCESS'"

```

Submit a Process with a Connect:Direct for z/OS RUN TASK from an HP NonStop Node

This Process, submitted from the Connect:Direct for HP NonStop node, performs a RUN TASK on the Connect:Direct for z/OS node, then copies from HP NonStop to z/OS.

Note: This Process will only work in SNA environments.

```

TANCPY  PROCESS      SNODE=CD.HPNONSTOP
STEP1   RUN TASK    ( SNODEID=(GRP.USR.PASWRD)
                      PGM=DMRTAMS,
                      PARM=(C"FREE=CLOSE,RETURN=(DD),SYSOUT=X
C" DELETE (CSDQA1.FILETEST.VSAM0016)
C" CLUSTER
C" DEFINE CLUSTER (
C" NAME(CSDQA1.FILETEST.VSAM0016)
C" RECORDS(1500 100)
C" VOLUMES(USER01)
C" INDEXED
C" KEYS(12 0)
C" RECORDSIZE(80 80)
C" SHAREOPTIONS(3)
C" )
C" DATA (
C" NAME(CSDQA1.FILETEST.VSAM0016.DATA)
C" CONTROLINTERVALSIZE(4096)
C" )
C" INDEX (
C" NAME(CSDQA1.FILETEST.VSAM0016.IDX)
C" CONTROLINTERVALSIZE(4096)
C" SHAREOPTIONS(3)
C" )
C" DATA (
C" NAME(CSDQA1.FILETEST.VSAM0016.DATA)
C" CONTROLINTERVALSIZE(4096)
C" )
C" INDEX (
C" NAME(CSDQA1.FILETEST.VSAM0016.IDX)
C" CONTROLINTERVALSIZE(4096)
C" )
                      ) SNODE
*
STEP2   COPY FROM  ( PNODE
                      DSN='\cycr.$user.qafiles.VSAM0001'
                      DCB=(LRECL=80)
                      SYSOPTS="SET XLATE ON"
                      )
                      TO ( SNODE
                          DSN=CSDQA1.FILETEST.VSAM0016
                          DISP=OLD
                          )

```

Use Symbolics in a IBM Connect:Direct HP NonStop Run Task to Place a Job In the Spooler On Hold

This IBM Connect:Direct HP NonStop Process uses symbolics to place a job in the spooler on hold. Output is sent to \$S.#SYMB. The job number will be resolved at Process submission. Double quotation marks enable the resolution of the &value in a quoted string.

```

PROC1   PROCESS      PNODE=CD.HPNONSTOP1
STEP1   RUN TASK    ( SNODE=CD.HPNONSTOP1
                      (PGM=SPOOLCOM)
                      SYSOPTS=("/OUT $S.#SYMB/JOB &JNUM,HOLD")
                      PNODE

```

The command to submit the Process is:

```
SUBMIT PROC PROC1 &JNUM=24
```

Create and Save an Online Save File Through Connect:Direct for i5/OS

This RUN TASK statement creates the online save file named SAVEFILE1 in the library TESTSV1 on the i5/OS. A copy of the library is saved to SAVEFILE1 in the library TESTDT1. All SYSOPTS keyword values must be enclosed in parentheses, and the entire SYSOPTS string must be enclosed in double quotation

marks. IBM Connect:Direct syntax requires using backslashes to continue the SYSOPTS over multiple lines. Bracketing backslashes allow for continuation of quotation marks when they begin and end on different lines.

```

STEP003  RUN TASK  (PGM=AS400) SNODE      -
          SYSOPTS=\ " \                   -
                    \CMD( \               -
                    \CRTSAVF \           -
                    \ FILE(TESTSV1/SAVEFILE1) \ -
                    \ TEXT('CREATED BY PROCESS CDTEST') \ -
                    \ ) \               -
                    \CMD( \               -
                    \SAVLIB \           -
                    \ LIB(TESTDT1) \    -
                    \ DEV(*SAVF) \     -
                    \ SAVF(TESTSV1/SAVEFILE1) \ -
                    \ ) \               -
          \ " \                           -

```

Create and Save an Online Save File Through Connect:Direct for i5/OS

This RUN TASK statement creates the online save file named SAVEFILE1 in the library TESTSV1 on the i5/OS. A copy of the library is saved to SAVEFILE1 in the library TESTDT1. All SYSOPTS keyword values must be enclosed in parentheses, and the entire SYSOPTS string must be enclosed in double quotation marks. IBM Connect:Direct syntax requires using backslashes to continue the SYSOPTS over multiple lines. Bracketing backslashes allow for continuation of quotation marks when they begin and end on different lines.

```

STEP003  RUN TASK  (PGM=AS400) SNODE      -
          SYSOPTS=\ " \                   -
                    \CMD( \               -
                    \CRTSAVF \           -
                    \ FILE(TESTSV1/SAVEFILE1) \ -
                    \ TEXT('CREATED BY PROCESS CDTEST') \ -
                    \ ) \               -
                    \CMD( \               -
                    \SAVLIB \           -
                    \ LIB(TESTDT1) \    -
                    \ DEV(*SAVF) \     -
                    \ SAVF(TESTSV1/SAVEFILE1) \ -
                    \ ) \               -
          \ " \                           -

```

Notify the i5/OS User of Successful Process Completion

This RUN TASK statement sends the message **PROCESS NEWACT HAS COMPLETED** to the message queue of workstation DSP07.

```

STEP012  RUN TASK  (PGM=AS400) SNODE      -
          SYSOPTS=\ " \                   -
                    \CMD( \               -
                    \ SNDBRKMSG \       -
                    \ MSG('PROCESS NEWACT HAS COMPLETED') \ -
                    \ TOMSGQ(DSP07) \   -
                    \ ) \               -
          \ " \                           -

```

Restore Libraries Through Connect:Direct for i5/OS

This Process restores to the i5/OS system the library named TESTDT1 in the save file named SAVEFILE2 in library TESTSV1. This library is restored to a library named TESTRL1.

```
STEP008B  RUN TASK  (PGM=AS400) SNODE          -
          SYSOPTS=\ "\                      -
                  \CMD( \                   -
                  \ RSTLIB \                 -
                  \ SAVLIB(TESTDT1) \       -
                  \ DEV(*SAVF) \           -
                  \ SAVF(TESTSV1/SAVEFILE2) \ -
                  \ RSTLIB(TESTRL1) \       -
                  \ ) \                     -
                  \ "\                      -
```

Submit a Process with a Connect:Direct for OpenVMS RUN TASK from an i5/OS Node

The example command sends a file from Connect:Direct for i5/OS to Connect:Direct for OpenVMS and performs a RUNTASK.

```
CDSND  SNODE(DWY1.TCP)
SNODENVIRN(OPENVMS)
FDSN('CDABC220/INITPARMS(INITPARMS)')
TDSN('DISK$SUP:<DYOUN1>AS400.RCV') FMSYSOPTS('TYPE(MBR)')
SNODEID(USERID USERPASSWORD) TDISP(RPL)TDCB(*N *N *N PS)
CDRUNTASK  SNODE(DWY2.TCP)
SNODENVIRN(OPENVMS)
CMD('CMD='`DEL
DISK$SUP:<DYOUN1>AS400.RCV;*`')
SNODEID(USERID USERPASSWORD)
```

Submit a Process with a Connect:Direct for HP NonStop RUN TASK from a Microsoft Windows Node

The following is an example of a Connect:Direct for HP NonStop RUN TASK submitted from a Sterling Connect:Direct for Microsoft Windows node.

```
NDM      PROCESS      SNODE=NDM.BAY1DR /*$HP NONSTOP$*/
          SNODEID=(NCC.NDM,xxxxxxx)
BEGINMSG RUN TASK      (PGM = BTFNDMLG
          SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
          /ND033 DNLOAD FUNBCC FUNBBETA TRU05")
          SNODE
COPYSTEP COPY FROM    (PNODE
          FILE = g:\vms\mftd\export\fusi\backup\TradeExport_20000818.002
          SYSOPTS = "DATATYPE(BINARY)STRIP.BLANKS(NO)XLATE(NO)" )
          TO          (SNODE
          FILE = $test.btfdata.beta5
          SYSOPTS = "'SET TYPE E' 'SET BLOCK 200' 'SET REC 200'"
          DISP = RPL)
          IF (COPYSTEP EQ 0) THEN
          GOTO OKMSG
          ELSE
ERRORMSG RUN TASK      (PGM = BTFNDMLG
          SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
          /ND035 DNLOAD FUNBCC FUNBBETA TRU05")
          SNODE
          EXIT
OKMSG    RUN TASK      (PGM = BTFNDMLG
          SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
          / ND034 DNLOAD FUNBCC FUNBBETA TRU05")
          SNODE
FT3IN    RUN TASK      (PGM = BTFNDMDN
          SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
          / DNLOAD FT3IN $test.btfdata.beta5 FUNBCC FUNBBETA TRU05")
```

Submit a Process with a Connect:Direct for HP NonStop RUN TASK from a Microsoft Windows Node

The following is an example of a Connect:Direct for HP NonStop RUN TASK submitted from a Sterling Connect:Direct for Microsoft Windows node.

```
NDM      PROCESS      SNODE=NDM.BAY1DR /*$HP NONSTOP$*/
                        SNODEID=(NCC.NDM,xxxxxxx)
BEGINMSG  RUN TASK    (PGM = BTFNDMLG
                        SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
                        /ND033 DNLOAD FUNBCC FUNBBETA TRU05")
COPYSTEP  COPY FROM  (PNODE
                        FILE = g:\vms\mftd\export\fusi\backup\TradeExport_20000818.002
                        SYSOPTS = "DATATYPE(BINARY)STRIP.BLANKS(NO)XLATE(NO)" )
                        TO      (SNODE
                        FILE = $test.btfdata.beta5
                        SYSOPTS = "'SET TYPE E' 'SET BLOCK 200' 'SET REC 200'"
                        DISP = RPL)
                        IF (COPYSTEP EQ 0) THEN
                            GOTO OKMSG
                        ELSE
ERRORMSG  RUN TASK    (PGM = BTFNDMLG
                        SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
                        /ND035 DNLOAD FUNBCC FUNBBETA TRU05")
                        SNODE
                        EXIT
OKMSG     RUN TASK    (PGM = BTFNDMLG
                        SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
                        / ND034 DNLOAD FUNBCC FUNBBETA TRU05")
                        SNODE
FT3IN    RUN TASK    (PGM = BTFNDMDN
                        SYSOPTS = "/NAME, IN $DATA2.BTFDATA.DEFTABLE, VOL $DATA2.BTFLOAD -
                        / DNLOAD FT3IN $test.btfdata.beta5 FUNBCC FUNBBETA TRU05")
```

Notify the i5/OS User of the Start of a Process

This Process performs a RUN TASK that sends the message PROCESS HAS STARTED to the message queue of workstation WSUSER01.

```
PROC01  PROCESS      SNODE=CD2200
                        SNODEID=(USER01,PSWD01)
STEP1    RUN TASK    SNODE (PGM=AS400)
                        SYSOPTS="CMD(SNDBRKMMSG MSG('PROCESS HAS STARTED!')
                        TOMSGQ(WSUSER01))"
PEND
```

Submit a Process from Connect:Direct for Microsoft Windows to Run DMRTSUB on z/OS

The following Process is submitted from Connect:Direct for Microsoft Windows to run DMRTSUB on a Connect:Direct for z/OS node. This Run Task using DMRTSUB will submit a job to run on z/OS and pass the symbolic for &NTDISP to the JCL it submits.

See the *IBM Sterling Connect:Direct for z/OS User Guide* for more information on DMRTSUB.

```

DMRTASK  PROCESS
          &NTDISP="RPL"
          SNODE=CSD.MVS.LEVEL1 /*$MVS$*/
          HOLD=NO
          CLASS=1
          PRTY=10
          EXECPRTY=10
          RETAIN=NO
          SNODEID=(USERID,USERPASSWORD)
RTASK01  RUN TASK  SNODE (PGM=DMRTSUB)
          SYSOPTS="C'DSN=JSMITH.PROCESS(SAMPLE),DISP=SHR'
                  C'NTDISP &NTDISP'"
          PEND

```

Microsoft Windows Run Task Statement

This **run task** statement starts testwin.exe and waits for it to finish:

```

jobstep  run task  snode (pgm=WINNT
                  sysopts="pgm(C:\winnt35\system32  estwin.exe)"

```

Submit a Process with a Connect:Direct for HP NonStop RUN TASK from an OpenVMS Node

This Process, submitted from the Connect:Direct for OpenVMS node, will invoke FUP at the Connect:Direct for HP NonStop node and write detailed information about the files in the subvolume to the spooler.

```

TANRUN  PROCESS  PNODE=CD.VMS -
                SNODE=CD.HPNONSTOP SNODEID=(GRP.USR,PASWRD) -
STEP1   RUN TASK (PGM=FUP -
                PARM=(' /OUT $S.#FUPOUT/' -
                    'INFO $DATA.TESTF.* ,DETAIL')) -
                SNODE

```

Submit a Process with a Connect:Direct for HP NonStop RUN TASK from an OpenVMS Node

This Process, submitted from the Connect:Direct for OpenVMS node, will invoke FUP at the Connect:Direct for HP NonStop node and write detailed information about the files in the subvolume to the spooler.

```

TANRUN  PROCESS  PNODE=CD.VMS -
                SNODE=CD.HPNONSTOP SNODEID=(GRP.USR,PASWRD) -
STEP1   RUN TASK (PGM=FUP -
                PARM=(' /OUT $S.#FUPOUT/' -
                    'INFO $DATA.TESTF.* ,DETAIL')) -
                SNODE

```

Submit a Process with a RUN TASK on OpenVMS from an OpenVMS Node

The following example shows a Process submitted from OpenVMS and executed on OpenVMS. The RUN TASK statement is coded with DCL commands that execute synchronously.

```

RTVMS   PROCESS  SNODE=CD.VMS.NODE1
STEP01  RUN TASK (PGM=VMS) PNODE -
                SYSOPTS="OUTPUT='DUC4:[JSMITH.TEST]RTLOG.LIS' -
                CMD='SET DEFAULT DUC4:[JSMITH.TEST]' -
                CMD='DIR' -
                CMD='SHOW TIME'"

```


Run Task on OpenVMS

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 for 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' "
```

Submit a Process from UNIX to Run a Program on z/OS

This example shows a Process initiating from UNIX that runs a program on z/OS. The **sysopts** string must be in uppercase characters to satisfy z/OS syntax requirements.

```
proc2    process    snode=OS390.node
step01   run task   snode (pgm=DMNOTIFY)
                    sysopts="CL44'DATA.BASE.P1',F'0010', XL8'FFA8' "
pend
```

Submit a Process from UNIX to Run a Program on z/OS

This example shows a Process initiating from UNIX that runs a program on z/OS. The **sysopts** string must be in uppercase characters to satisfy z/OS syntax requirements.

```
proc2    process    snode=OS390.node
step01   run task   snode (pgm=DMNOTIFY)
                    sysopts="CL44'DATA.BASE.P1',F'0010', XL8'FFA8' "
pend
```

Submit a Process with a Run Task on UNIX from Another UNIX Node

This Process initiates a Process from a UNIX node that executes commands at another UNIX node. The **sysopts** string must be enclosed in double quotation marks.

```
proc2    process    snode=unix.node
step01   run task   snode sysopts = "ls; ls -lax > lsout.ncr"
pend
```

Submit a Process from UNIX to Run a Program on i5/OS

This Process initiates a Process from a UNIX node that executes commands on an i5/OS node to delete two libraries. The **sysopts** string must be enclosed in double quotation marks.

```
proc1    process    snode=as400
                    snodeid=(userid,passwd)
step02   run task   (pgm=AS400)
                    snode
                    sysopts="CMD(DLTLIB LIB(URGRSSDT1)) CMD(DLTLIB LIB(URGRSS)) "
pend
```

Submit a Process from UNIX to Run a Program on Microsoft Windows

This Process copies a binary file from a Microsoft Windows node to a UNIX node. If the copy is successful, a **run task** statement is performed on the Microsoft Windows node (**snode**) that will delete the source

(from) file on the Microsoft Windows node. To delete the file, the keyword **cmd** is specified in the **sysopts** followed by the **del** command.

```

proc1 process      snode=CD.WIN
                   &file1="d:\data\out\reprts01.dat"
                   &file2="/data/in/reprts01.dat"
copy1 copy from    (file=&file1
                   sysopts="datatype(binary)"
                   snode)
                   to    (file=&file2
                   pnode)
run1  if (copy1 eq 0) then
run task      snode
             sysopts="cmd(del &file1)desktop(no)"
           eif
pend

```

In the previous example, rather than coding the specific file names in the process, symbolic variables are used in both the **copy** and **run task** statements. Since no user input is required for the delete command, the **desktop** parameter is set to NO and no console window is created on the Microsoft Windows desktop.

Example UNIX Run Task

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"

```

Execute DMRTDYN in a RUN TASK Environment (VM)

This Process calls DMRTDYN to determine if a file exists. If it does not exist, the user receives a nonzero return code, and a call is made to allocate the file.

```

DMRTDYN PROCESS  SNODE=SC.VM.USER1
LOCATE1 RUN TASK (PGM=DMRTDYN
                 PARM=(C'ALLOC',
                 C' DSN='PROFILE EXEC'',
                 C' DISP=(SHR) LINK=(IVVB,WIVVB,RR,191)',
                 C' DD=F1')) PNODE
IF(LOCATE1 NE 0) THEN
LOCATE2 RUN TASK (PGM=DMRTDYN
                 PARM=(C'ALLOC',
                 C' DSN='PROFILE EXEC'',
                 C' DISP=(NEW) LINK=(IVVB,WIVVB,RR,191)',
                 C' DD=F1')) PNODE
EIF
LOCATE3 RUN TASK (PGM=DMRTDYN
                 PARM=(C'UNALLOC '
                 C' DD=F1')) PNODE

```

Execute DMRTDYN in a RUN TASK Environment (VM)

This Process calls DMRTDYN to determine if a file exists. If it does not exist, the user receives a nonzero return code, and a call is made to allocate the file.

```

DMRTDYN  PROCESS  SNODE=SC.VM.USER1
LOCATE1  RUN TASK  (PGM=DMRTDYN                -
                   PARM=(C'ALLOC',            -
                           C' DSN='',PROFILE EXEC''', -
                           C' DISP=(SHR) LINK=(IVVB,WIVVB,RR,191)', -
                           C' DD=F1')) PNODE
                   IF(LOCATE1 NE 0) THEN
LOCATE2  RUN TASK  (PGM=DMRTDYN                -
                   PARM=(C'ALLOC',            -
                           C' DSN='',PROFILE EXEC''', -
                           C' DISP=(NEW) LINK=(IVVB,WIVVB,RR,191)', -
                           C' DD=F1')) PNODE
                   EIF
LOCATE3  RUN TASK  (PGM=DMRTDYN                -
                   PARM=(C'UNALLOC '          -
                           C' DD=F1')) PNODE

```

Resolve Symbolics Within DMRTDYN in a RUN TASK Environment (VM)

This example illustrates the structure of a IBM Connect:Direct Process that passes a parameter with single quotation marks in a DMRTDYN environment. Backslashes allow the resolution of the symbolic that must be entered between single quotation marks.

```

TESTSYM  PROCESS  SNODE=SC.VM.USER1 &XXX=XXX
LOCATE1  RUN TASK  (PGM=DMRTDYN                -
                   SYMBOL          &VMFILE=\xd5 MTT\  &XXX  \ FILETYPE'\
                   PARM=(C'ALLOC ',            -
                           C' DSN='', \&VMFILE\  \xd5 ',\
                           C' DISP=(SHR) LINK=(IVVB,WIVVB,RR,191)', -
                           C' DD=F1')) PNODE
LOCATE3  RUN TASK  (PGM=DMRTDYN                -
                   PARM=(C'UNALLOC '          -
                           C' DD=F1')) PNODE

```

Submit a Process with a RUN TASK on i5/OS from a VM Node

This example is initiated on a Connect:Direct for VM/ESA node to execute a RUN TASK on Connect:Direct for i5/OS. The SYSOPTS parameter specifies the i5/OS CL command DLTLIB.

```

RTVM     PROCESS  SNODE=CD.OS400
*****
* RUN TASK INITIATED FROM VM TO RUN ON OS400
*****
STEP3000 RUN TASK  (PGM=AS400) SNODE          -
                   SYSOPTS=\ "\
                           \CMD( \
                           \ DLTLIB \
                           \ LIB(TEST1) \
                           \ ) \
                           \ "\

```

Example VMESA Run Task

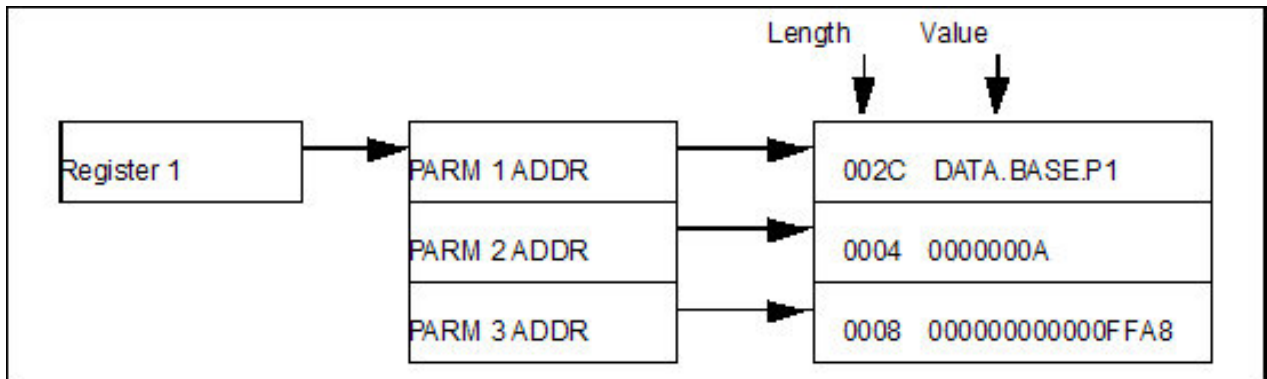
This example RUN TASK statement runs the program named MYTASK. It is attached to the Process on the secondary node (SNODE) and is passed a list of three parameter addresses.

```

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

```

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

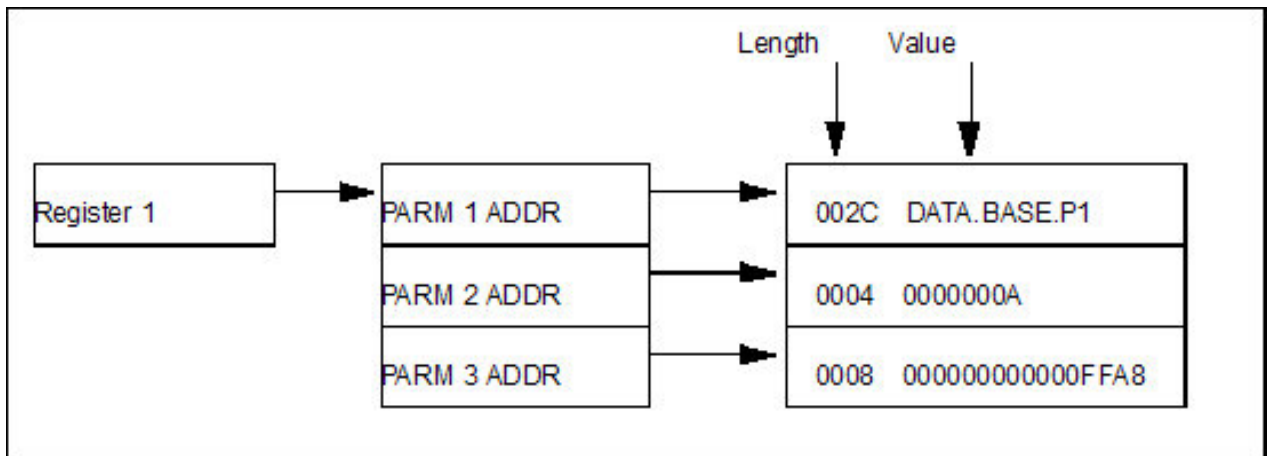


VSE Run Task Statement

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

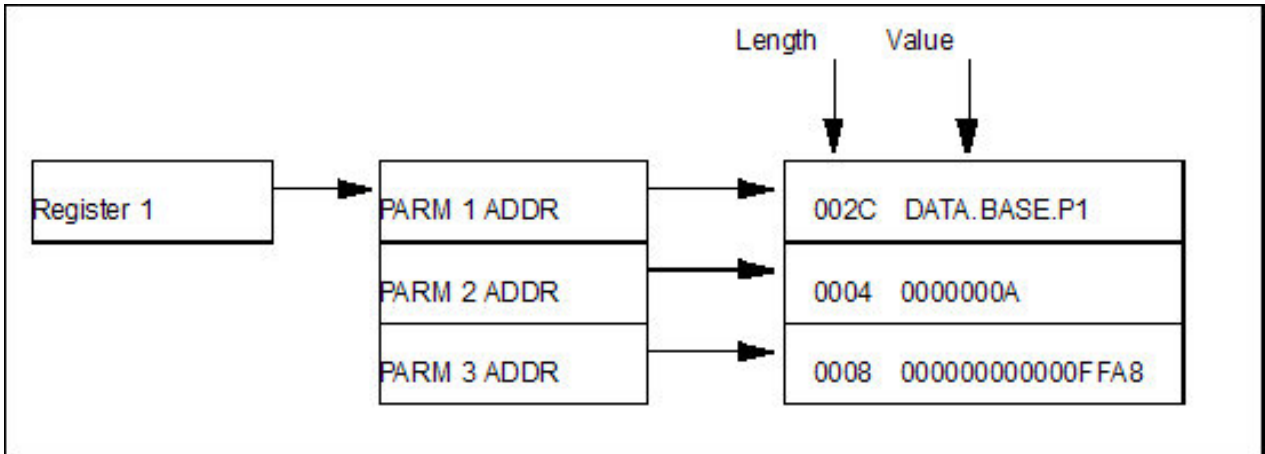


VSE Run Task Statement

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



RUN TASK Examples for CA-7 (z/OS to z/OS)

The following is an example of a RUN TASK statement that interfaces with the CA-7 scheduling package. The U7SVC program is supplied by CA-7, the job scheduling system by Computer Associates International, Inc. The U7SVC program may return a code of 0, regardless of completing the request.

For Processes with RUN TASK statements running U7SVC programs, replace xx with the length of the entire PARM including the d=.

```
STEP01 RUN TASK (PGM=U7SVC, -
                PARM=(CLxx"d=&dsn" ) -
                SNODE
```

In the following example, the symbolic variable, &UCC7DSN, is resolved to Z456789.TARGET, which is the proper format for the U7SVC program.

Note: To properly execute with CA-7, Connect:Direct for z/OS must not be running under CA-7 control.

```
PROC02 PROCESS          SNODE=CD.NODEB          -
                        &DSN1=Z123456.SRC      -
                        &DSN2=Z456789.TARGET   -
HF201  COPY FROM      (DSN=&DSN1 DISP=SHR)      -
        TO            (DSN=&DSN2 DISP=(RPL,CATLG)) -
        SYMBOL        &UCC7DSN=&DSN2
HF202  IF (HF201=0) THEN
        RUN TASK (PGM=U7SVC, PARM=(CLxx"D=&UCC7DSN" ) -
        SNODE
EIF
```

The following RUN TASK statement shows another way to interface with CA-7:

```
STEP01 RUN TASK PGM=U7SVC, -
                PARM=(\xd5 DEMAND,JOB=\ || &POSTJOB || -
                \,SCHID=001'\)) -
                SNODE
```

RUN TASK Examples for CA-7 (z/OS to z/OS)

The following is an example of a RUN TASK statement that interfaces with the CA-7 scheduling package. The U7SVC program is supplied by CA-7, the job scheduling system by Computer Associates International, Inc. The U7SVC program may return a code of 0, regardless of completing the request.

For Processes with RUN TASK statements running U7SVC programs, replace xx with the length of the entire PARM including the d=.

```
STEP01 RUN TASK (PGM=U7SVC, -
                PARM=(CLxx"d=&dsn" ) -
                SNODE
```

In the following example, the symbolic variable, &UCC7DSN, is resolved to Z456789.TARGET, which is the proper format for the U7SVC program.

Note: To properly execute with CA-7, Connect:Direct for z/OS must not be running under CA-7 control.

```
PROC02 PROCESS SNODE=CD.NODEB -
        &DSN1=Z123456.SRC -
        &DSN2=Z456789.TARGET -
HF201 COPY FROM (DSN=&DSN1 DISP=SHR) -
        TO (DSN=&DSN2 DISP=(RPL,CATLG)) -
        SYMBOL &UCC7DSN=&DSN2
HF202 IF (HF201=0) THEN
        RUN TASK (PGM=U7SVC,PARM=(CLxx"D=&UCC7DSN")) -
        SNODE
EIF
```

The following RUN TASK statement shows another way to interface with CA-7:

```
STEP01 RUN TASK PGM=U7SVC, -
                PARM=(\xd5 DEMAND,JOB=\ || &POSTJOB || -
                \,SCHID=001'\)) -
                SNODE
```

RUN TASK Using Control-M

The following RUN TASK statement shows an example of a Control-M interface:

```
RUN_TASK PROCESS SNODE=CCC.RZ1
STEP01 RUN TASK (PGM=CTM@IF10 -
                PARM=( -
                C'FUNC=LOADNSKE', -
                C'UNIQID=SKA#Y40TH4T', -
                C'DATREC=SKELLIB :JOBP.FT1A.SKEL', -
                C'DATREC=SKELNAME:CDSKEL', -
                C'DATREC=&&JOBNAME=Y40TH4T', -
                C'DATREC=&&JCLL=JOBP.CI1A.JCLL', -
                C'DATREC=&&DSN=', -
                C'DATREC=&&GROUP=CONNECT-DIRECT', -
                C'DATREC=&&DESC=CSLUX_TESTAUSLOESUNG')) -
                SNODE
```

RUN TASK Using DMRTDYN (z/OS)

This Process, SAMPLE1, is submitted from a Connect:Direct for z/OS node. Through the RUN TASK statement, DMRTDYN determines if the data set exists at the SNODE. If this data set exists, the LOCATE step receives a return code of 0, and the next step, DELETE, deletes the data set. STEP01 executes, regardless of the return code received from the LOCATE step. (STEP01 copies from the PNODE to the SNODE.)

```

SAMPLE1  PROCESS      PNODE=&PNODE          -
                SNODE=&SNODE          -
                HOLD=NO              -
LOCATE   RUN TASK    (PGM=DMRTDYN,    -
                PARM=(C"LOCATE DSN=&HLQ2..SAMP.OUT")) -
                SNODE                -
DELETE   IF (LOCATE = 0) THEN          -
RUN TASK (PGM=DMRTDYN,                -
                PARM=(C"ALLOC DSN=&HLQ2..SAMP.OUT DISP=(OLD,DELETE)" -
                F'-1'                  -
                C"UNALLOC DSN=&HLQ2..SAMP.OUT")) -
STEP01   EIF
COPY    FROM      (DSN=&HLQ1..SAMPLE.IN) -
        TO        (DSN=&HLQ2..SAMP.OUT  -
        DISP=RPL)

```

The command used to submit SAMPLE1 follows:

```

SUBMIT PROC=SAMPLE1          -
      &PNODE=CD.LOCAL        -
      &SNODE=CD.REMOTE       -
      &HLQ1=$LOCAL           -
      &HLQ2=$REMOTE

```

Copy a Member of an Object from i5/OS to a PDS Member and then Deleting the Library

This Process copies a member of an object from i5/OS to a member of a PDS on z/OS. The RUN TASK then deletes the library (with all its objects) from the i5/OS node.

```

PROC#001  PROCESS      SNODE=OS400.CD1      -
                PNODE=SC.OS390.CDA        -
                PRTY=8                    -
                NOTIFY=%USER              -
                CLASS=4                   -
                HOLD=NO                   -
                SNODEID=(RSMITH,RSMITH)    -
STEP001   COPY    FROM  (
                SNODE                      -
                DSN='LIBRY/RSMITH(LRECL80)' -
                SYSOPTS="TYPE(MBR)"        -
                DISP=SHR                   -
                )                          -
                COMPRESS                   -
                TO    (
                PNODE                      -
                DSN=RSMITH.OS400.CNTL(LRECL80) -
                DISP=SHR                   -
                )
STEP002   RUN TASK  (PGM=AS400) SNODE      -
                SYSOPTS="\ \              -
                \ \  CMD( \              -
                \ \  DLTLIB \           -
                \ \  LIB(LIBRY) \       -
                \ \  ) \                -
                \ \  \"

```

Submit a Process with a RUN TASK on OpenVMS from an z/OS Node

The following example shows a Process with a RUN TASK statement submitted from z/OS to execute on an OpenVMS node:

```

RUNTASK  PROCESS      SNODE=CD.NODE
STEP01   RUN TASK    (PGM=VMS) SNODE      -
                SYSOPTS=\ "CMD='SUBM/LOG=CDL:CD/USER=USR/PARA -
                (\ || &FILENM || \)CDC:RCV_CD'" \

```

Initiate a RUN TASK Statement at the HP NonStop Node (z/OS to HP NonStop)

Using conditional logic, this Process executes a program based on the completion code of STEP01. The PGM statement limits the user to eight characters. Concatenation characters are required within the RUN TASK statement because the parameters being passed to FUP are on multiple lines. TERM is coded on the RUN TASK statement so IBM Connect:Direct Processes can continue uninterrupted in the event the program being executed abends. The Process is submitted on z/OS.

```
PROC1  PROCESS      SNODE=CD.TAN          -
        SNODEID=(117.202,PSWRD)
STEP01  COPY FROM (PNODE DSN=JSMITH.GENPROC.LIB(COPY) DISP=SHR) -
        TO (SNODE DSN=$B.ROGER.ACCTJAN DISP=RPL)
STEP02  IF (STEP01 GT 0) THEN
        EXIT
        EIF
STEP03  RUN TASK    (PGM=FUP
        SYSOPTS="( /IN $USER.BGK.T1, TERM $ZTNP1.#PTH001H/)" -
        SNODE
```

Use Symbolics with a RUN TASK Statement (z/OS to HP NonStop)

This Process is submitted from the z/OS node to run a program at the HP NonStop node. The Process is structured so that the program name (PGM) and associated run-options for the HP NonStop RUN command are symbolics. (Symbolics allow you to predefine a Process that can be used for multiple applications.) The symbolics will be resolved when the Process is submitted. Because the NAME parameter is followed by a space, HP NonStop will assign a name to the HP NonStop process.

Note: This Process will only work in SNA environments.

```
PROCESS1  PROCESS    PNODE=CD.OS390          -
        SNODE=CD.HPNONSTOP
STEP01    RUN TASK  (PGM = &PGM
        PARM=(\  "/IN      \ || &INFILE  || -
              \  ,OUT     \ || &OUTFILE  || -
              \  ,PRI     \ || &PRI      || -
              \  ,CPU     \ || &CPU      || -
              \  ,NAME    \
              \  /")\))
        SNODE
```

The following IBM Connect:Direct syntax rules apply to this Process:

- The string of HP NonStop RUN command options must be enclosed in forward slashes (/). This is an HP NonStop syntax requirement.
- Bracketing backslashes (\) are positioned around variables in the string so that strings containing special characters can continue across multiple lines. Symbolics (&value) are not enclosed in bracketing backslashes.

Note: Run options for the HP NonStop RUN command must be separated by commas.

- Because the PNODE is the z/OS node, two vertical bars preceded and followed by blanks (||) are used to concatenate the value of a symbolic to the string. Resolution of the symbolic occurs before concatenation.

The command used to submit PROCESS1 is as follows:

```
SUB PROC=PROCESS1 &PGM=FUP &INFILE=FUPIN &OUTFILE=$S.#SPL1 &PRI=100 &CPU=0
```

Use Bracketing Backslashes and Quotation Marks (z/OS to HP NonStop)

PROCESS1 is coded to be submitted from the z/OS node to run a program at the HP NonStop node. The RUN TASK statement executes FUP to copy ACCTJAN with shared access to the spooler, \$S.#SPL1, at

the HP NonStop node. Any HP NonStop process-related error messages are directed to \$TERM1. IBM Connect:Direct-related messages are directed to \$.#SPL1.

Note: This Process will only work in SNA environments.

PROCESS1 shows a Process with a parameter (PARM) continuing over multiple lines.

```

PROCESS1  PROCESS      PNODE=CD.OS390                -
                SNODE=CD.HPNONSTOP                -
                SNODEID=(127.201,RSMITH)           -
STEP01    RUN TASK    (PGM = FUP                    -
                PARM=(\ "/OUT $.#SPL1           \ || -
                \ ,TERM $TERM1/           \ || -
                \ COPY $SYSTEM.BILLPROC.ACCTJAN,,SHARE") \) -
                SNODE

```

The following IBM Connect:Direct and HP NonStop syntax rules apply to both of these Processes:

- Within a IBM Connect:Direct Process submitted from an z/OS node, single quotation marks or double quotation marks must be used to allow special characters to be embedded within a file name.
- The string of HP NonStop RUN command options must be enclosed in forward slashes (/). This is an HP NonStop syntax requirement.
- Because the PNODE is an z/OS node (that is, the Process is submitted on the z/OS node), backslashes and vertical bars must be used to continue a string across multiple lines.

Bracketing backslashes are not valid when the PNODE is an HP NonStop node.

Use Concatenation Characters in a Run Task (z/OS to HP NonStop)

PROC1, PROC2, and PROC3 demonstrate the use of concatenation characters within a Connect:Direct for HP NonStop RUN TASK statement.

Note: These Processes will only work in SNA environments.

```

PROC1    PROCESS      SNODE=SYSCLX
STEP01  RUN TASK    (PGM=FUP                    -
                PARM=("/OUT $.#TEST,TERM $.#TMP/", -
                "COPY $.SMITH.TACLCSTM,,SHARE")) -
                SNODE

```

PROC2 and PROC3 require concatenation characters because the parameters being passed to FUP are on multiple lines.

```

PROC2    PROCESS      SNODE=SYSCLX
STEP01  RUN TASK    (PGM=FUP                    -
                PARM=(\ "/OUT $.#TEST,NAME $FUP   \ || -
                \ ,TERM $.#TMP,PRI 150/           \ || -
                \ COPY $.SMITH.TACLCSTM,,FOLD")   \) -
                SNODE

```

TERM is coded on the RUN TASK statement for both PROC2 and PROC3 so IBM Connect:Direct Processes can continue uninterrupted in the event the program being executed abends. If an abend occurs, any abend message will be sent to the device specified by the TERM command. If TERM is not coded, all abend messages will be directed to the terminal from which the Connect:Direct for HP NonStop system was started (HOMETERM). If HOMETERM is not paused, the abend message will not be displayed and the RUN TASK will hang until HOMETERM is paused.

```

PROC3  PROCESS  SNODE=SYSCLX
STEP01 RUN TASK  (PGM=FUP          -
                 PARM=(\"/OUT $$.#TEST,PRI 150  \ \ \ -
                       \,TERM $$.#TMP,NAME $FUP/  \ \ \ -
                       \ COPY $$.SMITH.TACLCSTM,, \ \ \ -
                       \ SHARE") \)              -
                 SNODE

```

Select Statistics for the Current Day (Sterling Connect:Direct for HP NonStop Run Task)

This Process performs a RUN TASK to select statistics for the current day. The resulting spooler file is then sent to a disk file on z/OS, where it can be printed or viewed.

Note: This Process will only work in SNA environments.

```

RUNT0005  PROCESS  SNODE=HPNONSTOP.CD
STEP1     RUN TASK  (PGM=NDMCOM          -
                   PARM=( '/OUT $$.#JOECOM/' , -
                           ' SEL STAT STARTT=(TODAY,)' , -
                           ' EXIT ' )         -
                   ) SNODE
STEP2     COPY FROM (SNODE              -
                   DSN=$$.#JOECOM          -
                   DISP=OLD)              -
                   TO (PNODE             -
                       DSN=JOE.FILETEST.COMDOC -
                       DISP=(NEW,CATLG)      -
                       SPACE=(TRK,(2,1),RLSE) -
                       DCB=(DSORG=PS,       -
                             RECFM=FBA,     -
                             LRECL=133,     -
                             BLKSIZE=3990  -
                             UNIT=SYSDA    -
                             VOL=SER=M80006) -
                       )

```

Define a VSE VSAM File and Copying a Sequential File from z/OS

This multistep Process, initiated from an z/OS node, consists of RUN TASK statements and a COPY statement. STEP1 runs the DMRTAMS utility to delete and then define a target VSAM cluster on a Connect:Direct for VSE/ESA node. STEP2 runs the DMRTDYN utility to unallocate the SYSOUT output data set generated by STEP1. STEP3 copies a sequential file from an z/OS node to a VSE node.

```

VSEVSAM PROCES      PNODE=SC.OS390.NODE1  SNODE=SC.VSE.NODE1
STEP1    RUN TASK  (PGM=DMRTAMS,
                  PARM=(C" MSG=YES DSN=SYSOUT.SYS011 DD=123 DISP=SHR",
                        C" DELETE VSE.VSAM.TEST CLUSTER
                        C" DEFINE CLUSTER
                        C" (NAME(VSE.VSAM.TEST)
                        C" RECORDS(25000 5000)
                        C" VOLUMES(VSAM01)
                        C" INDEXED
                        C" REUSE
                        C" KEYS(8 6)
                        C" RECORDSIZE(262 880)
                        C" NOREPLICATE
                        C" SPANNED
                        C" SHR(2))
                        C" DATA(
                        C" NAME(VSE.VSAM.TEST.DATA)
                        C" CISZ(4096))
                        C" INDEX(
                        C" NAME(VSE.VSAM.TEST.INDEX)
                        C" CISZ(512))
                  ))
STEP2    RUN TASK  (PGM=DMRTDYN
                  PARM=(C'UNALLOC DSN=SYSOUT.SYS011 DD=123')) SNODE
STEP3    COPY FROM (DSN=OS390.SEQ.DATASET
                  DISP=(SHR,KEEP)
                  PNODE)
          TO       (DSN=VSE.VSAM.TEST
                  DISP=NEW
                  DCB=(DSORG=VSAM)
                  SNODE)

```

Submit a Process from z/OS to Execute UNIX Commands

This Process initiates a Process from z/OS that executes commands on UNIX. The `SYSOPTS` string must be in the proper case for UNIX and enclosed in double quotation marks.

```

PROC2    PROCESS   SNODE=UNIX.NODE
STEP01  RUN TASK  (PGM=UNIX) SYSOPTS="ls -lax > lsout.ncr"

```

Submit a Process from z/OS to Execute Microsoft Windows Programs

This Process initiates a Process from z/OS that runs a program on Microsoft Windows. The `desktop` parameter is set to YES to allow the program to interact with the desktop. The `sysopts` string is enclosed in backslashes.

```

PROC2    PROCESS   SNODE=NT.NODE
STEP01  RUN TASK  (PGM="NT") SNODE
                  SYSOPTS=\xd5 CMD(D:\CDCLNT\PROGPACK.EXE
                  D:\CDCLNT\PROGRAM.PAC\ D:\CDCLNT\*.*) DESKTOP (YES)\

```

Use Run Task to Create a Save File on i5/OS

In this RUN TASK statement submitted from z/OS or z/OS, 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)\
                  \)\
                  \" \
```

Submit a Run Task from z/OS to i5/OS

In this RUN TASK statement submitted from z/OS, the CRTSAVF command creates a save file named SAVFILE. The file has the text description CREATED BY PROCESS TEST. The SAVLIB command 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)\
                  \)\
                  \" \
```

z/OS RUN TASK to Execute a COBOL Program

The following example shows how to execute a COBOL program from a IBM Connect:Direct Process by using the RUN TASK statement.

```
ALLOC1  RUN TASK  (PGM=DMRTDYN
                  PARM=(C'ALLOC DSN=XYZ.TEMP DISP=SHR DD=OLD1')) PNODE      -
ALLOC2  RUN TASK  (PGM=DMRTDYN
                  PARM=(C'ALLOC DSN=XYZ.TEMP2 DISP=SHR DD=OLD2')) PNODE      -
CUSTRT  RUN TASK  (PGM=COBOL program name)
UNALLO1 RUN TASK  (PGM=DMRTDYN
                  PARM=(C'UNALLOC DD=OLD1')) PNODE
UNALLO2 RUN TASK  (PGM=DMRTDYN PARM=(C'UNALLOC DD=OLD2')) PNODE
```

Pass Mixed Case Parameters through an z/OS RUN TASK to DMRTSUB

z/OS RUN TASK parameters are always passed as upper case data, with the exception of parameters passed by RUN TASK to the DMRTSUB program. Mixed case parameters can be passed to DMRTSUB by using the SYSOPTS parameter.

The following example shows how to pass mixed case parameters to DMRTSUB through SYSOPTS in an z/OS RUN TASK statement.

```

STEP1  RUN TASK  (PGM=DMRTSUB )                /* RUNTASK PGM */ -
        SYSOPTS=\\"\\                          /* START SYSOPTS */ -
        'DSN=PROD.JCL.LIB(TESTPGM),DISP=SHR'  /* DATASET NAME */ -
        'FIRSTNM Bill'                        /* PARAMETER #1 */ -
        'LASTNM Smith'                       /* PARAMETER #2 */ -
        \\"\\                                    /* END SYSOPTS */ -
        SNODE

```

The next example shows passing mixed case parameters to DMRTSUB using variable substitution. The use of single and double quotes are reversed from the previous example.

```

STEP1  RUN TASK (PGM=DMRTSUB )                /* RUNTASK PGM */ -
        SYSOPTS=\\"\\                          /* START SYSOPTS */ -
        "DSN=PROD.JCL.LIB(TESTPGM),DISP=SHR" /* DATASET NAME */ -
        "FIRSTNM &FIRST"                     /* PARAMETER #1 */ -
        "LASTNM &LAST"                       /* PARAMETER #2 */ -
        \\"\\                                    /* END SYSOPTS */ -
        SNODE

```

You must use SYSOPTS rather than PARM to pass mixed-case parameters.

Use of Conditional Statements (OpenVMS to z/OS)

This Process uses conditional logic to check the completion code of STEP01. If the completion code is equal to 0, STEP03 copies a file from OpenVMS to z/OS. If the completion code is greater than 0, STEP04 initiates the RUN TASK statement to notify the OpenVMS user that the copy was unsuccessful.

```

PROC1  PROCESS      SNODE=XYZNODE                -
        SNODEID=(JSMITH,JERRY)                  -
STEP01  COPY FROM   (DSN=JSMITH.DATA              -
                   SNODE)                       -
        TO         (DSN=U1:[RJONES.CDTEST]RUSS.DAT -
                   DISP=RPL)                    -
STEP02  IF (STEP01 = 0) THEN                      -
STEP03  COPY FROM   (DSN=U1:[RJONES.DATA_FILES]TEXT.DAT -
                   TO (DSN=JSMITH.YES SNODE      -
                   DCB=(DSORG=PS,RECFM=V) DISP=RPL) -
        ELSE
STEP04  RUN TASK    (PGM=VMS) PNODE SYSOPTS="      -
                   CMD = 'REPLY/USER=RJONES COPY_UNSUCCESSFUL'" -
        EIF

```

Use of Conditional Statements (OpenVMS to z/OS)

This Process uses conditional logic to check the completion code of STEP01. If the completion code is equal to 0, STEP03 copies a file from OpenVMS to z/OS. If the completion code is greater than 0, STEP04 initiates the RUN TASK statement to notify the OpenVMS user that the copy was unsuccessful.

```

PROC1  PROCESS      SNODE=XYZNODE                -
        SNODEID=(JSMITH,JERRY)                  -
STEP01  COPY FROM   (DSN=JSMITH.DATA              -
                   SNODE)                       -
        TO         (DSN=U1:[RJONES.CDTEST]RUSS.DAT -
                   DISP=RPL)                    -
STEP02  IF (STEP01 = 0) THEN                      -
STEP03  COPY FROM   (DSN=U1:[RJONES.DATA_FILES]TEXT.DAT -
                   TO (DSN=JSMITH.YES SNODE      -
                   DCB=(DSORG=PS,RECFM=V) DISP=RPL) -
        ELSE
STEP04  RUN TASK    (PGM=VMS) PNODE SYSOPTS="      -
                   CMD = 'REPLY/USER=RJONES COPY_UNSUCCESSFUL'" -
        EIF

```

Use of Conditional Logic (z/OS to VSE)

The following Process issues a VSE/Power command based on the return code of the previous COPY step. If the transfer is successful, the VSE/Power command releases GSVSE01. If the transfer fails, GSVSE02 is released. This Process is designed to be submitted from the z/OS node.

TESTVSE	PROCESS		SNODE=CD.VSE.NODE	-
			PNODE=CD.OS390.NODE	
STEP01	COPY	FROM	(DSN=JSMITH.CNTL.JCL(VSAMDEL)	-
			DISP=SHR	-
			UNIT=SYSDA)	-
			CKPT=0K	-
		TO	(DSN=JSMITH.CNTL.JCL(XXXX)	-
			DISP=RPL	-
			LIBR=(SUBLIB=JCL,TYPE=JCL)	-
			DCD=(DSORG=VSAM)	
	IF (STEP01 NE 0)	THEN		
		RUN TASK	(PGM=DMRTPWR	-
			PARM=(F'5'	-
			F'3'	-
			('R RDR,GSVSE01')	-
			SNODE	
			EXIT	
	ELSE			
		RUN TASK	(PGM=DMRTPWR	-
			PARM=(F'5'	-
			F'3'	-
			('R RDR,GSVSE02')	-
			SNODE	
			EXIT	
	EIF			

Use of Conditional Logic (i5/OS to z/OS)

This Process copies a file from an Connect:Direct for i5/OS node to z/OS. STEP02 checks the completion code from STEP01 and executes STEP04 if the completion code is greater than 0. If STEP01 receives a completion code of 0, STEP03 will execute and send a message to terminal DSP03 indicating the file transfer was successful. If STEP01 receives a completion code greater than 0, STEP04 will execute and send a message to terminal DSP03 indicating the file transfer failed.

```

COPY02  PROCESS  SNODE=CD.SANFRAN.AS400      -
              PRTY=8                          -
              NOTIFY=USERID                   -
              CLASS=4                          -
              SNODEID=(USERID,PSWD)           -
STEP01  COPY     FROM (                        -
              SNODE                           -
              DSN='OS400/FILE001'            -
              SYSOPTS="TYPE(MBR)"           -
              DISP=SHR                        -
              )                                -
              TO (                             -
              PNODE                           -
              DSN=OS390.FILE002              -
              DCB=(DSORG=PS,BLKSIZE=6160,LRECL=080,RECFM=FB) -
              DISP=RPL                        -
              )                                -
STEP02  IF (STEP01 > 0) THEN                  -
        GOTO STEP04                          -
        ELSE                                  -
STEP03  RUN TASK (PGM=OS400) SNODE           -
              SYSOPTS="\\"                   -
              \CMD(\                          -
              \SNDBRKMSG\                     -
              \MSG('FILE TRANSFER SUCCESSFUL')\ -
              \TOMSGQ(DSP03)\                -
              \)\                              -
              \"\                              -
        EIF                                  -
        EXIT                                  -
STEP04  RUN TASK (PGM=OS400) SNODE           -
              SYSOPTS="\\"                   -
              \CMD(\                          -
              \SNDBRKMSG\                     -
              \MSG('FILE TRANSFER FAILED!!  ')\ -
              \TOMSGQ(DSP03)\                -
              \)\                              -
              \"\

```

Use of Conditional Logic (HP NonStop to z/OS)

In this multi-step Process, STEP01 will execute FUP to purge files FILE1, FILE2, FILE3, and FILE4 from \$B.FILERESO on the PNODE. A message will be sent to the spooler (\$S.#FUPTEST) that indicates whether FUP executed successfully.

STEP02 will copy DATA1.FILEA from the z/OS node (SNODE) to \$B.FILERESO.FILE1 at the PNODE. The file will default to the same type of file being copied.

Conditional logic in STEP03 is then used to check the completion code of STEP02. If the completion code is greater than 4, then STEP04 will execute. If the completion code from STEP02 is 4 or less, then DATA1.FILEB will be copied from the z/OS node to \$B.FILERESO.FILE2 at the HP NonStop node.

STEP04 will then execute and copy DATA1.FILEC from the z/OS node to \$B.FILERESO.FILE3 at the HP NonStop node. If the completion code is greater than 8, then no further processing will occur. If the completion code of STEP03 is greater than 4, DATA1.FILED will be copied from the z/OS node to \$B.FILERESO.FILE4 at the HP NonStop node.

```

COND1  PROCESS    PNODE=CD.HP NONSTOP      -
STEP01  RUN TASK  (PGM=FUP          -
                  PARM=( '/OUT $S.#FUPTEST/ ', -
                          'VOLUME $B.FILERESO', -
                          'PURGE FILE1! ', -
                          'PURGE FILE2! ', -
                          'PURGE FILE3! ', -
                          'PURGE FILE4! ' )) -
STEP02  COPY FROM (DSN=DATA1.FILEA DISP=SHR SNODE) -
          TO      (DSN=$B.FILERESO.FILE1 DISP=NEW PNODE)
STEP03  IF      (STEP02 GT 4) THEN -
          GOTO STEP04
          ELSE -
          COPY FROM (DSN=DATA1.FILEB SNODE) -
          TO      (DSN=$B.FILERESO.FILE2 DISP=NEW PNODE)
          EIF -
STEP04  COPY FROM (DSN=DATA1.FILEC SNODE) -
          TO      (DSN=$B.FILERESO.FILE3 DISP=NEW PNODE)
          IF      (STEP03 GT 8) THEN -
          EXIT -
          EIF -
          IF      (STEP03 LT 4) THEN -
          COPY FROM (DSN=DATA1.FILED SNODE) -
          TO      (DSN=$B.FILERESO.FILE4 DISP=NEW PNODE)
          EIF

```

Use of Conditional Statements to Test for Process Completion on OpenVMS

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

```

Use of Conditional Statement on OpenVMS

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

```


Use of Conditional Statements in a UNIX Process

This Connect:Direct for UNIX Process contains all of the conditional statements.

```
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"
step04  if      (step03 >= 8) then
         exit
step05  if      (step03 lt 4) then
step06  copy  from  (file=xyz pnode)
         to      (file=uvw snode)
         eif
step07  exit
         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.

Use of Conditional Statements in VMESA

The following example contains all conditional VMESA 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
STEP07  EXIT
         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.

Use of Conditional Statements in a VSE Process

The following VSE 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.

Use of Conditional Statements in a Microsoft Windows Process

This Microsoft 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=WINNT
           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=Windows
           sysopts="pgm(failjob.exe)"
           exit
```

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 failjob.exe.

Use of Conditional Logic to Test Message ID in z/OS

```
MSGID PROCESS SNODE=CD.SNODE
*
* THIS PROCESS ILLUSTRATES HOW TO TEST THE MESSAGE ID OF THE PREVIOUS
* STEP AND TAKE ACTION BASED ON A SPECIFIC MESSAGE ID. IN THE CASE
* IF THE PUSH STEP GENERATES A COMPLETION MESSAGE OF SCPA000I THEN
* EXIT AND DO NOT EXECUTE STEP PULL.
*
PUSH COPY FROM (DSN=CDZ.TCPIP.DATA.FILE -
                DISP=(SHR) PNODE) -
           TO   (DSN=CDZ.FTST.XYZ -
                DISP=(RPL) SNODE) -
IF (%MSGID EQ C'SCPA000I') THEN
  EXIT
EIF
PULL COPY FROM (DSN=CDZ.TCPIP.DATA.FILE -
                DISP=(SHR) SNODE) -
           TO   (DSN=CDZ.FTST.XYZ -
                DISP=(RPL) PNODE)
```

Use of Conditional Logic to Test Highest Return Code with %MAXRC in z/OS

```
CDRESET PROCESS SNODE=CD.SNODE &USERID=%USER
*
* This Process illustrates how the MAXRC variable could be used to test
```

```

* the highest return code of all steps and the use of the utility
* DGADWAIT to reset the MAXRC value to zero.
*
LOC   RUN TASK (PGM=DGADTDYN -
        PARM=(C'LOCATE DSN=CDZ.TCPIP.DATA.FILE.OUT'))
      IF (LOC EQ 8) THEN
RES   RUN TASK (PGM=DGADWAIT PARM=( '%MAXRC,0' )) PNODE
      EIF
PUSH  COPY FROM (DSN=CDZ.TCPIP.DATA.FILE -
        DISP=(SHR) -
        PNODE) -
        TO (DSN=CDZ.TCPIP.DATA.FILE.OUT -
        DISP=(RPL) -
        SNODE)
PULL  COPY FROM (DSN=CDZ.TCPIP.DATA.FILE -          DISP=(SHR)
        -          SNODE) -          TO
        (DSN=CDZ.TCPIP.DATA.FILE.OUT
        -          DISP=(RPL) -
        -          PNODE)
      IF (%MAXRC GT 0) THEN
NOGOOD RUN TASK (PGM=DMNOTFY2 -
        PARM=(CL4'FAIL',C'NOT GOOD ON PNODE',&USERID)) PNODE -
      ELSE
GOOD   RUN TASK (PGM=DMNOTFY2 -
        PARM=(CL4'GOOD',C'GOOD ON PNODE',&USERID)) PNODE -
      EIF

```

Use Conditional Logic to Test Variables in z/OS

```

CDEMPY PROCESS SNODE=CD.SNODE HOLD=YES -
        &USERID=%USER -
        &EMPTY=NO -
        &XDATA=TODAY
*
* THIS PROCESS ILLUSTRATES HOW TO TEST A VARIABLE THAT WAS PASSED TO THE
* PROCESS AND MAKE A DECISION BASED ON THE VALUE. THE VALUE FOR EMPTY
* AND XDATA COULD BE PASSES BY THE SUBMIT COMMAND.
*
STEP1 COPY FROM (DSN=CDZ.TCPIP.DATA.FILE -
        DISP=(SHR) PNODE) -
        TO (DSN=CDZ.FTST.XYZ -
        DISP=(RPL) SNODE)
      IF (&EMPTY EQ C'NO') THEN
R01 RUN TASK PNODE (PGM=DMNOTFY2 -
        PARM=(CL4'GOOD',C'EMPTY IS NO',&USERID))
      ELSE
R02 RUN TASK PNODE (PGM=DMNOTFY2 -
        PARM=(CL4'GOOD',C'EMPTY IS YES',&USERID))
      EIF
      IF (&XDATA EQ C'YESTER') THEN
R03 RUN TASK PNODE (PGM=DMNOTFY2 -
        PARM=(CL4'GOOD',C'XDATA IS YESTER',&USERID))
      ELSE
R04 RUN TASK PNODE (PGM=DMNOTFY2 -
        PARM=(CL4'GOOD',C'XDATA IS TODAY',&USERID))
      EIF

```

Example UNIX pend Statement

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

```

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as shown in the next column.

© 2015.

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. 2015.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium and the Ultrium Logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Connect Control Center®, Connect:Direct®, Connect:Enterprise, Gentran®, Gentran®:Basic®, Gentran:Control®, Gentran:Director®, Gentran:Plus®, Gentran:Realtime®, Gentran:Server®, Gentran:Viewpoint®, Commerce™, Information Broker®, and Integrator® are trademarks, Inc., an IBM Company.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,

INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.



Part Number:
Product Number: 5655-X01

(1P) P/N: