PN 13

CONNECT:Direct ™ for VM/ESA

Administration Guide

Version 3.2

Sterling Commerce
An IBM Company

CONNECT:Direct for VM/ESA Administration Guide
Version 3.2
First Edition

This document was prepared to assist licensed users of the Sterling Commerce, Inc., CONNECT:Direct for VM/ESA system; its contents may not be used for any other purpose without prior written permission. The material contained herein is supplied without representation or warranty of any kind and is based on typical use. Any unusual use may produce unpredictable results. Sterling Commerce, therefore, assumes no responsibility and shall have no liability of any kind arising from the supply or use of this document or the material contained herein.

References in this manual to Sterling Commerce products, programs, or services do not imply that Sterling Commerce intends to make these available in all countries in which Sterling Commerce operates.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR 52.227-19.

Printed in the United States of America.

Product Enhancement Request Form

Company Name:_____

Address:_____

City:_____ **State:**_____ **Zip:**_____

Requester:_____ **Telephone:**_____

Product:

___ CONNECT:Direct for MVS
___ CONNECT:Direct for VSE/ESA
___ CONNECT:Direct for VM/ESA
___ CONNECT:Direct for MSP
___ CONNECT:Direct for NetWare
___ CONNECT:Direct for Tandem
___ CONNECT:Direct for UNIX
___ CONNECT:Direct for Stratus VOS
___ CONNECT:Direct for OpenVME
___ CONNECT:Direct for OpenVMS
___ CONNECT:Direct for OS/400
___ CONNECT:Direct for OS/2
___ CONNECT:Direct for OS/2 Warp
___ CONNECT:Direct for MS-DOS
___ CONNECT:Direct for Windows NT
___ CONNECT:Direct for Windows 95
___ CONNECT:Direct for RemoteWare
___ CONNECT:Direct Requester for Motif
___ CONNECT:Direct Requester for Windows
___ CONNECT:Mailbox for VSE
___ CONNECT:Mailbox for MVS
___ CONNECT:Mailbox for OS/400
___ CONNECT:Mailbox SPC Option
___ CONNECT:Mailbox for UNIX
___ CONNECT:Supertracs for MVS
___ CONNECT:Supertracs for VSE
___ CONNECT:Tracs for OS/2
___ CONNECT:Tracs for MVS
___ CONNECT:Tracs for MS-DOS
___ CONNECT:Tracs for VSE
___ CONNECT:Queue
___ CONNECT:Firewall
___ CONNECT:Remote for Windows NT

Current Product Release Level: _____

Describe enhancement:

_____

_____

_____

_____

Over

How will enhancement be used?

_____

_____

_____

_____

_____

_____

Would you install a new release of this CONNECT product to get the enhancement?

___ Yes **___ No**

Enhancement priority:

___ Need ASAP
___ Have immediate plan to use when available
___ Good idea; nice to have

Additional comments:

_____

_____

_____

_____

_____

_____

_____

Mail or fax this completed form to:

Sterling Commerce
Communications Software Group
Attention: Product Marketing
5215 North O'Connor Boulevard, Suite 1500
Irving, Texas 75039-3771
Fax: (972) 868-5142

Thank you for your interest in the Sterling Commerce CONNECT Product Family.

Contents

Chapter 3 Controlling Security

Chapter 4 Maintaining User Authorization

Chapter 5 Maintaining the Type File

Chapter 6 Maintaining the Network Map

Appendix A Initialization Parameters

Glossary

Index

Preface

This manual is for systems personnel responsible for the administration, security, and maintenance of CONNECT:Direct on IBM's Virtual Machine/ Enterprise System Architecture (VM/ESA) operating system.

This manual assumes knowledge of the VM/ESA operating system and Interactive Systems Productivity Facility (ISPF).

What This Guide Contains

The CONNECT:Direct for VM/ESA Administration Guide contains the following chapters:

- Chapter 1, *About CONNECT:Direct for VM/ESA*, provides a brief overview of CONNECT:Direct and the CONNECT:Direct for VM/ESA internal components and concepts.
- Chapter 2, *Using Administration Commands*, introduces the CONNECT:Direct Primary Options Menu and defines the format, parameters, and purpose of the various administrative commands.
- Chapter 3, *Controlling Security*, explains the Security Exits and the Authorization Facility.
- Chapter 4, *Maintaining User Authorization*, describes the Authorization File and its maintenance using CONNECT:Direct commands.
- Chapter 5, *Maintaining the Type File*, defines the purpose, format, and parameters of the CONNECT:Direct commands for Type files. It describes how to use the commands through Batch Interface or IUI.

- Chapter 6*, Maintaining the Network Map*, defines the Network Map, command format and parameters, and maintenance procedures through the Batch Interface or IUI.
- Chapter 7*, Selecting, Queuing, and Recovering Processes*, describes CONNECT:Direct Process logic and selection, defines the Wait, Execution, Hold, and Timer logical queues (TCQ,) and describes Process recovery and checkpoint/restart procedures.
- Chapter 8, *Using CONNECT:Direct Exits*, explains the Statistics, Submit, and Allocation Exits.
- Chapter 9*, Customizing CONNECT:Direct*, outlines a method for customizing, fine tuning, and maintaining CONNECT:Direct.
- Chapter 10, *Administering Statistics,* describes the statistics facility, the process of archiving and monitoring statistics records, and maintaining your statistics file.
- Chapter 11, *Using the Program Interface*, describes User Application Programs and DMCHLAPI.
- Chapter 12, *DBCS Support*, provides an overview of Double-byte Character Set (DBCS) translation tables and a list of parameters for customizing translation tables.
- Appendix A, *Initialization Parameters*, provides a detailed description of each CONNECT:Direct and VSAM initialization with the default value listed for each initialization parameter.

Glossary defines terms used throughout this manual.

Notational Conventions

This section describes the notational conventions used in this guide.

Uppercase Letters

Uppercase letters in the command format indicate that you type in information as shown.

Uppercase and Lowercase Letters

A statement, command, or parameter in uppercase letters followed by lowercase letters indicates an alternative to typing the entire command.

For example, SELect PROCess means that you need only type SEL PROC for the command to be valid.

Lowercase Letters

Lowercase letters or words in commands or syntax boxes require substitution by the user. For example, PNODE=primary-node-name indicates that you must provide the name of the primary node.

Bold Letters

Bold print in syntax boxes indicates CONNECT:Direct commands and required parameters. For example, **DSN=filename** indicates that the parameter *DSN* is required.

Commands, Process statements, parameters, and special keys are sometimes bold in text to differentiate them from other words.

Underlined Letters

Underlining indicates default values for parameters and subparameters. For example, RETAIN=Yes|No|Initial specifies that the default for *RETAIN* is *NO*.

Vertical Bars

Vertical bars indicate that you can supply one of a series of values separated by the vertical bars. For example HOLD=Yes|No|Call specifies that *Yes* or *No* or *Call* is valid.

Brackets

Brackets indicate that information is optional. For example, STARTT=([date|day][,hh:mm:ssXM]) indicates that you can specify either a date or a day, a date or a day plus a time, or just a time.

Additional Notations

Code all commas and parentheses as they appear.

Process, as shown with a capital **P**, refers to a CONNECT:Direct Process.

`Monospaced characters` (characters of equal width) represent information for screens, commands, Processes, and reports.

Italics indicate book, chapter, and section titles or show emphasis in the text.

CONNECT:Direct for VM/ESA Documentation

Use the *CONNECT:Direct for VM/ESA Administration Guide* in conjunction with other CONNECT:Direct product documents. The following manuals make up the CONNECT:Direct for VM/ESA library:

- *CONNECT:Direct for VM/ESA Release Notes* is a document shipped with CONNECT:Direct for VM/ESA that lists system requirements, maintenance updates, and enhancements.
- *CONNECT:Direct for VM/ESA User's Guide* defines all CONNECT:Direct commands and explains how to use them in Interactive User interface (IUI), Batch Interface, CMS Command line Interface, and other utilities.
- *CONNECT:Direct for VM/ESA Installation Guide* is a manual for planning and installing CONNECT:Direct for VM/ESA.
- *CONNECT:Direct Process Guide* provides you with the information needed to write a CONNECT:Direct Process for the MVS, VSE/ESA, VM/ESA, MSP, OS/400, Tandem, UNIX, VOS, and VMS platforms.
- *CONNECT:Direct Console Operator's Guide* is a manual for the operator who initiates CONNECT:Direct for MVS, VSE/ESA, and VM/ESA data transfer activities from the operator console.
- *CONNECT:Direct Problem Isolation Guide* is a manual that explains how to fix errors encountered when performing CONNECT:Direct for MVS, VSE/ESA, and VM/ESA functions.
- *CONNECT:Direct Quick Reference* is a capsulized reference of CONNECT:Direct for MVS, VSE/ESA, and VM/ESA Process statements, commands, and installation parameters.
- *CONNECT:Direct Product Overview* is an introduction to the CONNECT:Direct product family and its data transfer applications.
- *CONNECT:Direct Event Services Support System Guide* provides information on the system architecture, system operation, and event data format for Event Services Support.

Chapter 1

About CONNECT:Direct for VM/ESA

CONNECT:Direct distributes information and manages production activities among multiple mainframes, minicomputers, workstations, and personal computers in diverse operating system environments.

CONNECT:Direct allows data centers within and across networks to:

- Move large amounts of data
- Share information
- Schedule related application activities
- Automate data distribution
- Control and audit network activities
- Maintain network security
- Use a common command structure and environment-specific interfaces

CONNECT:Direct goes beyond traditional file transfer systems by eliminating the time-consuming, error-prone operator procedures associated with moving data. CONNECT:Direct capabilities extend from basic data movement functions to the management of data movement activities.

CONNECT:Direct for VM/ESA Internal Components

This section provides an overview of the internal components of CONNECT:Direct for VM/ESA and a brief description of each of the

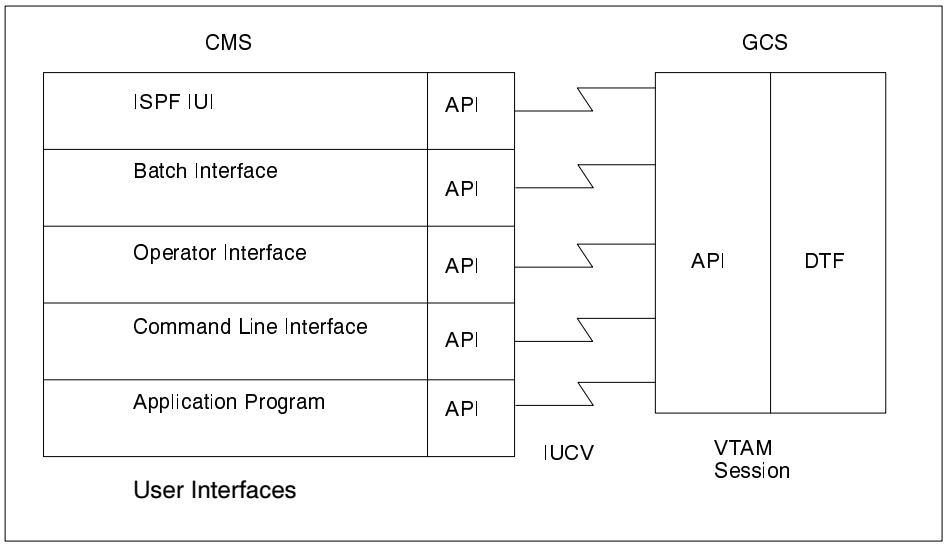
CONNECT:Direct interfaces. The three internal components of CONNECT:Direct for VM/ESA are:

- Various user interfaces to CONNECT:Direct for VM/ESA
- Application Program Interface (API)
- Data Transmission Facility (DTF)

While CONNECT:DIrect runs as an application on the VM/ESA operating system, system components interact to execute the statements and commands submitted through the CONNECT:Direct interfaces.

Note: All statements and commands must pass through the API regardless of the interface from which they are submitted.

The following figure shows a single CONNECT:Direct for VM/ESA. Each piece of the diagram is discussed in the pages that follow.



User Interfaces

There are five ways to communicate with the CONNECT:Direct API:

- ISPF Interactive User Interface (IUI)
- Batch Interface
- Operator Interface
- Command Line Interface (CLI)
- Application Program

The following sections describe each interface.

ISPF Interactive User Interface (IUI)

The IUI is a screen and dialog component running under the Interactive System Productivity Facility (ISPF) in CMS.

The IUI builds CONNECT:Direct commands based on information provided on the panels and submits them to the API. The IUI and API validate the syntax of statements as they are created, issuing messages indicating acceptance or rejection of the request. CONNECT:Direct acts upon a request and then displays a message to the user.

Use the IUI to create and submit Processes, as well as perform the following tasks (based on the security level of the user):

- Initiate file transfers
- Schedule file transfers
- Establish defaults for COPY attributes
- Monitor and modify CONNECT:Direct for VM/ESA task activity
- Concurrently signon to other CONNECT:Direct sessions
- Display statistics online or off–line
- Display or print messages online
- Authorize new users
- Stop CONNECT:Direct
- Initiate traces
- Display, update, and insert Network Map entries
- Monitor and modify all CONNECT:Direct Process activity (using commands such as SUBMIT, CHANGE, and SELECT)

See the *CONNECT:Direct for VM/ESA User's Guide* to learn how to use commands through the IUI.

Batch Interface

The Batch Interface allows you to issue CONNECT:Direct commands from a batch stream through the use of the DMBATCH program. Refer to the *Issuing Commands Through the Batch Interface* chapter of the *CONNECT:Direct for VM/ESA User's Guide* for more information on the the batch interface.

Operator Interface

The Operator Interface, which executes as a subtask of the DTF, enables the console operator to submit commands through CLISTs for system administration and maintenance. Refer to the *CONNECT:Direct for VM/ESA Console Operator's Guide* for a description of the Operator Interface.

Command Line Interface

The Command Line Interface (CLI) allows users to request CONNECT:Direct services from CMS. The command line interface runs under CMS and utilizes the DMCHLAPI interface to communicate with the DTF running under GCS. Refer to the *Issuing Commands Through the Command Line Interface* in the *CONNECT:Direct for VM/ESA User's Guide* for more information on how to use this interface.

Application Programs

A high-level language Application Program Interface (API) allows user programs to interface directly to the CONNECT:Direct API. Refer to the *Using the Program Interface* chapter beginning on page 11-1 for more information on the API.

Application Program Interface

The Application Program Interface (API) consists of the programs that allow the IUI, the Batch Interface, the Operator Interface, Command Line Interfaces, and application programs to communicate with the DTF or CONNECT:Direct Server. The API performs the following functions:

- Interprets commands from the various interfaces
- Validates the command format
- Passes the command across a VTAM session for DTF processing
- Receives the appropriate response or the requested data back from the DTF after the command is processed

Data Transmission Facility

The Data Transmission Facility (DTF) performs the following functions:

- Executes instructions, coded as Processes, passed to it from the API
- Starts and terminates all sessions
- Selects the next Process to execute
- Controls I/O requests for Shared File System(SFS) data sets
- Controls information distribution to other CONNECT:Direct nodes in the network

---

Note: Define start-up parameters that govern the overall activity of the DTF during CONNECT:Direct for VM/ESA installation.

Transmission Control Queue

The Transmission Control Queue (TCQ) controls Process execution. CONNECT:Direct stores submitted Processes in the TCQ. The TCQ is divided into four logical queues: Wait, Execution, Hold, and Timer.

The Process is put in the appropriate queue based on Process statement parameters that affect scheduling. Examples of such parameters are the RETAIN and HOLD parameters and the STARTT parameter that indicates the day and time the Process should be executed.

CONNECT:Direct selects Processes in a first-in, first-out manner for execution within Process class and priority as sessions are available. You can access the queues and manipulate the Processes with commands entered through one of the user interfaces.

Chapter 2

Using Administration Commands

This chapter describes how to perform the following tasks:

- Accessing the CONNECT:Direct Administrative Options Menu
- Displaying and flushing tasks
- Using the Native command structure
- Displaying the Settings of the Initialization Parameters
- Running diagnostics
- Stopping CONNECT:Direct

CONNECT:Direct Administrative Options Menu

The Administrative Options Menu is not available to all users. Access to CONNECT:Direct functions is controlled through the User Authorization file described in the *Maintaining User Authorization* chapter beginning on page 4-1.

Accessing the Administrative Options Menu

To access the Administrative Options Menu, perform one of the following procedures:

- Select **ADMIN** from the CONNECT:Direct Primary Options Menu.
- Type **=ADMIN** on any CONNECT:Direct screen command line and press **ENTER**.

Refer to the *CONNECT:Direct for VM/ESA User's Guide* for more information on the Administrative Options Menu.

Using the Administrative Options Menu

To request one of the functions from the CONNECT:Direct Administrative Options Menu, type the abbreviation for the function on the command line (CMD) and press **ENTER**.

```
node.name          CONNECT:Direct ADMINISTRATIVE OPTIONS MENU
CMD ==>

SELECT ONE OF THE FOLLOWING:
   ST    - VIEW TYPE RECORD                          ********************
   IT    - INSERT/UPDATE TYPE RECORD                 *                  *
   DT    - DELETE TYPE RECORD                        * TODAY: yyyy/mm/dd*
                                                     * TIME:  hh:mm     *
   SU    - VIEW USER AUTHORIZATION RECORD            *                  *
   IU    - INSERT/UPDATE USER AUTHORIZATION RECORD   ********************
   DU    - DELETE USER AUTHORIZATION RECORD

   TS    - VIEW CONNECT:DIRECT TASKS
   TF    - FLUSH A CONNECT:DIRECT TASK

   MD    - MODIFY CONNECT:DIRECT TRACE CHARACTERISTICS
   C     - ENTER A NATIVE CONNECT:DIRECT COMMAND
   SN    - TERMINATE CONNECT:DIRECT

   NM    - VIEW THE CONTENTS OF THE CONNECT:DIRECT NETWORK MAP
   UNM   - UPDATE THE CONNECT:DIRECT NETWORK MAP

   INQ   - INQUIRE ABOUT DTF INTERNAL STATUS
   STAT  - PERFORM STATISTICS FUNCTIONS
```

A brief description of each option follows.

Understanding the Administrative Options Menu

From the Administrative Options Menu you can select options to branch to screens that will allow you to:

- Maintain the Type Defaults file that contains file attribute information used during Process submission:

  ST displays the Select Type screen where you can examine a record in the Type file and select the output to go to a file, table, or printer.

  IT displays the Insert/Update Type screen where you can add or change a record in the Type Defaults file.

  DT displays the Delete Type screen where you can remove a record from the Type file.

- Maintain the User Authorization file that controls access to CONNECT:Direct functions:

  SU displays the Select User screen where you can examine the profile of a user in the CONNECT:Direct Authorization file.

IU displays the Insert/Update User screen where you can add a user to the system or change a user's privileges on the system.

DU displays the Delete User screen where you can remove a user from the CONNECT:Direct Authorization file.

- Select and flush tasks:

  TS accesses the Select Task screen where you can determine the task ID, task type, and task number for a task.

  TF displays the Flush Task screen where you can remove a task from the execution queue.

- Initialize traces, enter native commands, terminate CONNECT:Direct, and display the Inquire DTF Internal Status screen options.

  MD displays the Modify screen where you can request traces and modify system functions.

  C displays the Native Command screen where you can enter and execute any CONNECT:Direct for VM/ESA command by providing it in native syntax.

  SN displays the Stop CONNECT:Direct screen where you can stop the operation of CONNECT:Direct.

- View and maintain the Network Map.

  NM displays the Select Network Map screen, where you choose for display or printed output the defined CONNECT:Direct nodes from the Network Map file and network addresses. See the *CONNECT:Direct for VM/ESA User's* Guide for use of the SELECT NETMAP command.

  UNM allows you to update the Network Map dynamically.

- Inquire about DTF internal status and perform statistics functions.

  INQ displays the following Inquire DTF Internal Status screen from which you can request information about DTF initialization parameter settings, statistics logging facility, and the statistics archive file directory of the CONNECT:Direct for VM/ESA product.

```
node.name                INQUIRE DTF INTERNAL STATUS
CMD ==>                                                          hh:mm
                                                                 yyyy/mm/dd
                                                                 yy.ddd


             Please select one of the following:

             ISTA   - Inquire about Statistics Facility status
             IPRM   - Inquire about DTF Initialization parameters
             IDIR   - Inquire about the Statistics Archive Directory
```

STAT displays the Statistics Command screen from which you can request functions relative to the Statistics files, such as initiating statistics file pair switching, confirm statistics file archival, enable statistics recording, and disable statistics recording.

```
node.name               STATISTICS Command                       hh:mm
CMD ==>                                                          yyyy/mm/dd
                                                                 yy.ddd


    SELECT ONE OF THE FOLLOWING OPTIONS:


    FS  - INITIATE STATISTICS FILE PAIR SWITCH

    CF  - CONFIRM STATISTICS FILE ARCHIVAL, FILE PAIR NUM ==> __

    EN  - ENABLE STATISTICS RECORDING, RCD TYPES ==> __ __ __ __ __ __

    DI  - DISABLE STATISTICS RECORDING, RCD TYPES ==> __ __ __ __ __ __
```

Displaying and Flushing Tasks

CONNECT:Direct Tasks are service functions or requests processed by the DTF represented by a Task Control Area (TCA) that is used by the DTF to process and manage the request. CONNECT:Direct Tasks can be divided into two categories:

System Tasks

perform the services needed for the operation of the CONNECT:Direct DTF.

User Tasks

represent work done within the DTF on behalf of a user request.

The following table lists the CONNECT:Direct tasks and their functions.

Type	Task	Function
System	STATISTICS/ACTIVITY RECORDING (A)	Controls statistics logging
	CONSOLE (OPERATOR) TASK (C)	Allows the user to communicate to the DTF through the operator console
	FITSCAN TCA (F)	Performs Process initialization services
	LOGON TCA (FOR DMCONLOG) (L)	Reserved for use during logon processing
	MASTER TASK (M)	Controls the dispatching and logon processing for the DTF
	OPEN/CLOSE TASK (O)	Opens and closes files
	TIMER TASK (T)	Performs timer services for the master task and Process related timer functions
	TCP/IP SERVER (U)	Listens for incoming TCP/IP session requests
	File Server Interface Task (V)	Communicates with the CMS SFS Server TAsk using IUCV
	Statistics Archive Submit Task (Z)	Submits the Statistics File Archive process
User	PRIMARY SESSION TASK (P)	Manages the work related to a request that initiated the current session
	SECONDARY SESSION TASK (S)	Manages the work related to a partner PNODE Task
	INTERACTIVE TASK (I)	Manages the requests from a session with an IUI user

Displaying Tasks

Use the SELECT TASK command to select and display the status of CONNECT:Direct tasks. SELECT TASK has the following format.

Label	Command	Parameters
(optional)	**SEL**ect **TASK**	**PR**int | **TAB**le

PRint | TABle

specifies where the output of the command is directed.

PRint specifies that output is to go to the log printer.

TABle specifies that output be displayed in tabular format.

Batch Interface Use of SELECT TASK

To use this command from the Batch Interface, perform these steps:

1. Place your commands in a batch job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.
2. Ensure CONNECT:Direct is running.
3. Submit the job.

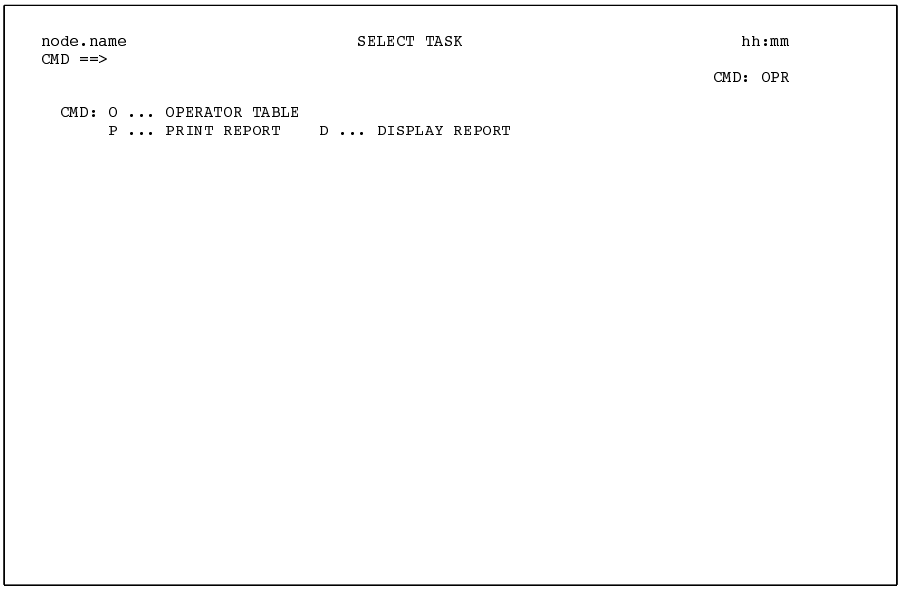
The following example illustrates using the SELECT TASK command and sending the output to the log printer.

```
SEL TASK PRINT
```

IUI Use of SELECT TASK

To issue the SELECT TASK command from the IUI, perform these steps:

1. Access the SELECT TASK screen by selecting option **TS** from the Administrative Options Menu.

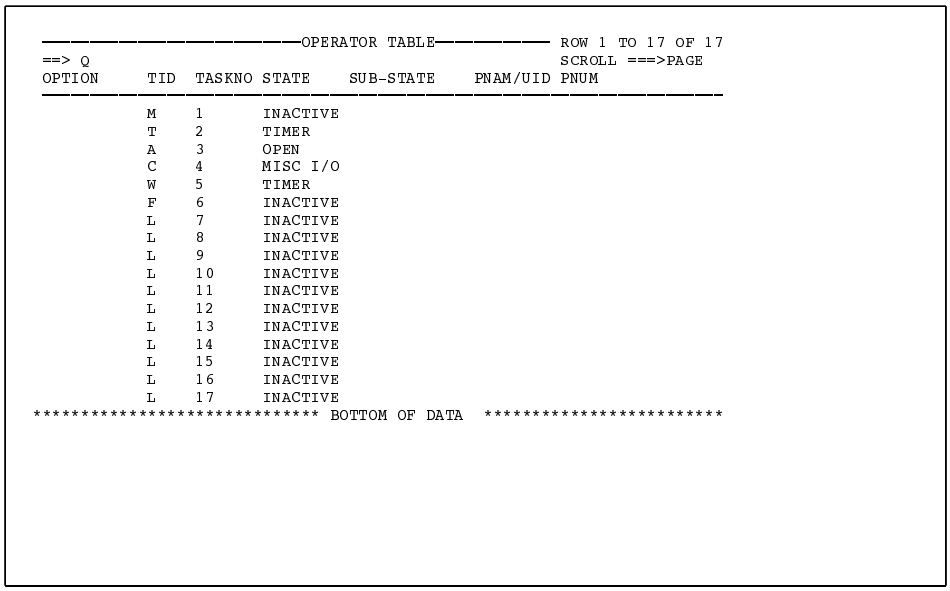


2. Select the output display type. Refer to the following table for a description of each of the output display types.

This table describes each of the output display types for the SELECT TASK display:

Display Type	Definition
P	Sends a copy of the Select Task Table to the printer.
O	Accesses the Select Task Operator Table.
D	Displays a copy of your tasks with IDs and type designations.

The following example illustrates using the SELECT TASK command and sending the output to the Operator Table.



```
————————————————————OPERATOR TABLE———————————— ROW 1 TO 17 OF 17
==> Q                                           SCROLL ===>PAGE
OPTION     TID  TASKNO STATE    SUB-STATE    PNAM/UID PNUM
————————————————————————————————————————————————————————————
           M    1      INACTIVE
           T    2      TIMER
           A    3      OPEN
           C    4      MISC I/O
           W    5      TIMER
           F    6      INACTIVE
           L    7      INACTIVE
           L    8      INACTIVE
           L    9      INACTIVE
           L    10     INACTIVE
           L    11     INACTIVE
           L    12     INACTIVE
           L    13     INACTIVE
           L    14     INACTIVE
           L    15     INACTIVE
           L    16     INACTIVE
           L    17     INACTIVE
****************************** BOTTOM OF DATA  ************************
```

Note: Refresh the display by pressing the **ENTER** key while **Q** is displayed at the command prompt.

The following example illustrates using the SELECT TASK command and displaying the output.

```
NDMAPI       LISTING    A1   V   121    Trunc=121   Size=35   Line=0   Col=1   Alt=0
===>
* * * Top of File * * *
============================================================
                        SELECT TASK
============================================================
 TASK TASK           XMIT          PNAME/
  ID  NUM  STATE     STATE                  PNUM
 ==USERID==
____________________________________________________________
  M   001  INACTIVE
  T   002  TIMER
  A   003  INACTIVE
  Z   004  INACTIVE
  C   005  MISC I/O
  F   006  INACTIVE
  L   027  INACTIVE
  L   028  INACTIVE
  L   029  INACTIVE
  L   030  INACTIVE
  L   031  INACTIVE
  L   032  INACTIVE
  L   033  INACTIVE
  L   034  INACTIVE
  L   035  INACTIVE
  L   036  INACTIVE
  L   037  INACTIVE
  U   007  VTAM I/O
```

Note: The tasks display includes IDs and type designations.

Removing or Flushing Tasks From Execution

Use the FLUSH TASK command to remove a task from execution. You must identify the task by its task number.

Note: Only use the FLUSH TASK command if you cannot flush the Process using the FLUSH PROCESS command described in the *CONNECT:Direct for VM/ESA User's Guide*.

The FLUSH TASK command has the following format and associated parameters. Required parameters appear in bold print.

Label	Command	Parameters
(optional)	**FLush TASK**	**WHERE (**
		TASK = (tasknumber | (list)
		)
		FORCE

Required Parameter

WHERE is the required parameter for the FLUSH TASK command.

WHERE (TASK = (tasknumber | (list))
specifies which task(s) to flush.

TASK = tasknumber | (list) specifies the task(s) to flush by either task number or a list of task numbers.

Optional Parameter

FORCE is the optional parameter for the FLUSH TASK command.

FORCE
specifies that the task flush will be forced.

WARNING: Do not use the FORCE parameter when the task is executing on a LU6.2 session. If FORCE is specified, the session will terminate immediately and statistics for the task will not be exchanged between the two nodes.

If you do not specify the FORCE option for the FLUSH TASK command, then an indicator notifies the program executing on behalf of the task that a FLUSH TASK command was issued for that task.

If that program is not in control (for example, if it is waiting on a request outside of CONNECT:Direct code to complete), then it does not see the FLUSH TASK indicator and the task is not flushed; otherwise, the program looks for the FLUSH TASK indicator and takes the appropriate action.

When you specify the FORCE option, then the action taken depends on the STATE and SUBSTATE of the task for which you issued the FORCE FLUSH. Refer to the *Suspending, Flushing, and Deleting Processes* section of the *Controlling Process in the TCQ* chapter of the *CONNECT:Direct for VM/ESA User's Guide* for the actions taken for the specific STATE and SUBSTATE. You can determine the STATE and SUBSTATE of the task by doing a SELECT TASK command.

Batch Interface Use of FLUSH TASK

To use this command from the Batch Interface, perform these steps:

1. Place your commands in the DMBATCH batch job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.

2. Ensure CONNECT:Direct is running.

3. Submit the job.

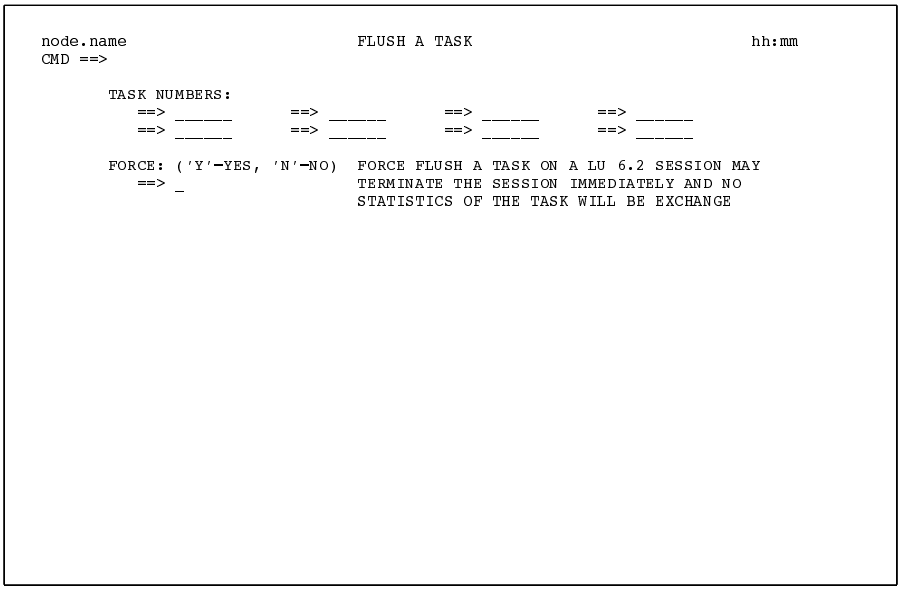
The following example illustrates using the FLUSH TASK command to force flush three CONNECT:Direct tasks.

```
FLUSH TASK WHERE (TASK=(100,105,120) FORCE)
```

IUI Use of FLUSH TASK

To issue the FLUSH TASK command in the IUI, perform these steps:

1. Access the FLUSH A TASK screen by selecting option **TF** from the Administrative Options Menu.



2. Input the numbers of the tasks you want to flush.

3. Type in **Y** if you want to force the flush. The default is N.

Note: A list of the requested tasks is displayed indicating if the flush was successful.

---

4. Verify the results.

Using the FLUSH TASK from the Operator Table

To flush a task from the IUI's Operator Table, complete these steps:

1. Access the Operator Table as described on page 2-7.

The following example illustrates the use of the SELECT TASK command with output to the Operator Table.

```
————————————————————OPERATOR TABLE———————————— ROW 1 TO 17 OF 17
==> Q                                           SCROLL ===>PAGE
OPTION     TID  TASKNO STATE    SUB-STATE    PNAM/UID PNUM
————————————————————————————————————————————————————————————
           M    1      INACTIVE
           T    2      TIMER
           A    3      OPEN
           C    4      MISC I/O
           W    5      TIMER
           F    6      INACTIVE
           L    7      INACTIVE
           L    8      INACTIVE
           L    9      INACTIVE
           L    10     INACTIVE
           L    11     INACTIVE
           L    12     INACTIVE
           L    13     INACTIVE
           L    14     INACTIVE
           L    15     INACTIVE
           L    16     INACTIVE
           L    17     INACTIVE
****************************** BOTTOM OF DATA  ************************
```

2. Type **F** next to the task ID to flush the task.

Note: You can also suspend a task in the Operator table by typing **P** next to the task ID to suspend the task.

Using the Native Command Structure

Use the Native Command structure to build a more detailed list of parameters than the command panels allow. You can enter any CONNECT:Direct command or series of CONNECT:Direct commands.

Accessing the Native Command Structure

To access the Native Command Screen, select option **C** from the Administrative Options Menu. The following display shows the Native Command Screen.

```
node.name              NATIVE COMMAND SCREEN                  hh:mm

CMD ==>


 ENTER COMMAND TEXT:

   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
   ==> _____________________________________________________________
```

Note: You must observe certain rules when you use the Native Command Screen as defined in the following *Understanding the Native Command Structure* section.

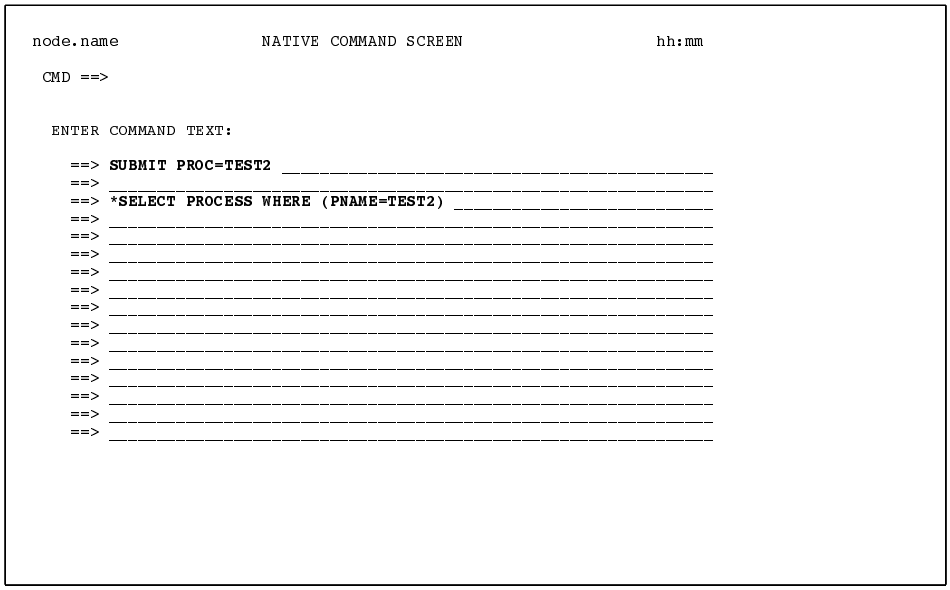
Understanding the Native Command Structure

Observe the following rules when you type your command:

- Keywords must start on the next line or must be broken at the separator (blank or comma).
- To use comments on the Native Command Screen, type an asterisk in the first column of the input line. This enables you to issue commands without retyping them.
- You cannot continuously wrap commands across lines on the Native Command Screen.

A command that creates a temporary file displays the temporary file for you to browse after the command executes.

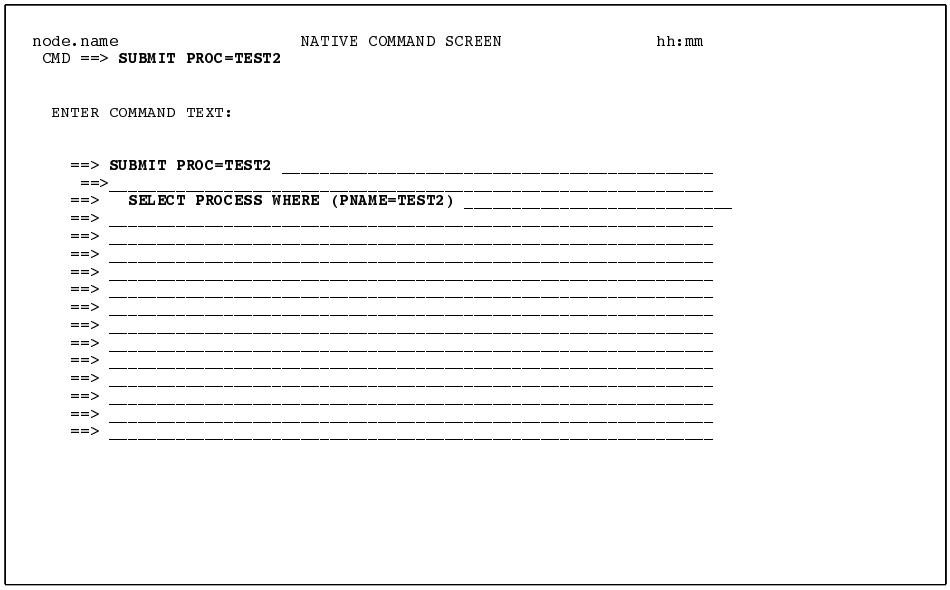
In the following example, when you press the **ENTER** key, you submit the Process called TEST2.



To monitor the progress of TEST2, complete the following steps:

1. Type an asterisk in column 1 of the first input line (before SUBMIT).
2. Delete the asterisk from the third input line (before SELECT) and press **ENTER**.

In this screen sample, you submit the Process TEST2 from the Command line. The SELECT PROCESS takes place just as in the previous example.



Entering Commands From the ISPF Command Line

In addition to the Native Command structure, you can use the ISPF Command Line to build a more detailed list of parameters than the command panels allow. You can enter any CONNECT:Direct command or series of CONNECT:Direct commands.

For example, you can use one command string from an ISPF command line that does these tasks in this order:

- Signs on to the IUI
- Accesses the Native Command Screen
- Issues CONNECT:Direct commands
- Returns to ISPF without seeing any IUI screens

You must specify Signon defaults to bypass the CONNECT:Direct Signon screen.

Note: Although the Native Command Screen is not displayed, CONNECT:Direct handles the request as if it originated from the command line. For this reason, the Native Command Screen must be specified before the first CONNECT:Direct command.

Execute the commands from any ISPF command line. The length of the command string is limited only by the space available on the ISPF command line.

Example

In this example, the following actions occur from the ISPF command line:

- The Process named T3 is submitted
- The Process named T2 is released
- All Processes are selected for printing

```
=N;ADMIN;C;SUB PROC=T3;CH PROC WHERE(PNAME=T2) RELEASE;
SEL PROC WHERE() PRINT;=SIGNOFF
```

This table describes each of the commands issued from the previous example:

Command	Task
N	Executes IUI signon
ADMIN	Accesses the Administrative function menu
C	Accesses the Native Command Screen
SUB PROC=T3	Submits the Process named T3
CH PROC WHERE (PNAME=T2) RELEASE	Releases the Process named T2
SEL PROC WHERE() PRINT	Selects all Process(es) for printing
=SIGNOFF	Signs the user off CONNECT:Direct

You can also request the Native Command Screen and issue the same requests from the screen command line. To do so, type the following on the Native Command Screen after the CMD prompt and press **ENTER**.

```
SUB PROC=T3;CH PROC WHERE (PNAME=T2) RELEASE;
SEL PROC WHERE() PRINT;=SIGNOFF
```

Displaying the Settings of the Initialization Parameters

The INQUIRE INITPARM command displays the current settings of the DTF initialization parameters. This following is a partial sample report.

```
=====================================================================
node.name       *INQUIRE INITPARM*  DATE: mm/dd/yyyy  TIME: hh:mm:ss
=====================================================================

CKPT                    => 1000
CKPT.MODE               => (RECORD  BLOCK   NOPDS   PDS     )
CKPT.DAYS               => 4
MAX.TAPE                => 10
MAXPRIMARY              => 12
MAXSECONDARY            => 12
MAXBATCH                => 0
MAXUSERS                => 12
MAXPROCESS              => 24
PRTYDEF                 => 10
PRTYINT                 => 00:30:00
                        .
                        .
```

The INQUIRE INITPARM Command Format

The INQUIRE INITPARM command has the following format.

Label	Command	Parameters
(optional)	**INQuire INITparm**	

There are no required parameters for this command.

Batch Interface Use of INQUIRE INITPARM

To use the INQUIRE INITPARM command from the Batch Interface, perform the following steps:

1. Place your commands in the DMBATCH batch job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.
2. Ensure CONNECT:Direct is running.
3. Submit the job.
4. Verify the results.

Note: Set the fifth character of the DMBATCH output parameter specification to a **Y** to print the result of the command that is in the temporary data set.

IUI Use of INQUIRE INITPARM

To display the DTF initialization parameters, perform these steps:

1. Select option **INQ** from the Administrative Options Menu.
2. Type **IPRM**, and press **ENTER** to display a report of the settings for the initialization parameters.
3. Use the report to verify your initialization parameter settings.

Running Diagnostics

The MODIFY command allows you to run CONNECT:Direct diagnostics. You can dynamically modify operational functions and request traces. Most traces begin as soon as you issue the command.

Note: The MODIFY command yields the same types of traces as the DEBUG initialization parameter, except that MODIFY does not require that you bring down and restart CONNECT:Direct.

The MODIFY Command Format

The MODIFY command has this format and associated parameters.

Label	Command	Parameters
(optional)	MODIFY	BITS.OFF = X'nnnnnnnn'
		BITS.ON = X'nnnnnnnn'
		CLOSE = ddname
		DDNAME = (ddname, nn)
		DYN (batch) = 'dynamic
		allocation string'
		DYN (IUI) dynamic
		allocation string
		SESSIONS = Quiesce | Resume
		LOG.PRINTER = OFF | ON | luname
		MODDIR.TRACE = YES

The DYN parameter is represented by both batch and the IUI. The parameter DYN (batch) has an equal sign before and quotes around dynamic allocation string. The parameter DYN (IUI) does not have an equal sign before or quotes around dynamic allocation string.

Required Parameters

There are no required parameters for the MODIFY command.

Optional Parameters

The following are descriptions for the optional parameters used with the MODIFY command.

BITS.OFF = X'nnnnnnnn'
BITS.ON = X'nnnnnnnn'

See the column named *DEBUG Setting* in the following table for valid values of nnnnnnnn.

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 DMCBCOPY	DMCBCOPY
00040000	GETMAIN/FREEMAIN trace	RADBDD16
00008000	I/O buffer trace	RADBDD21
00004000	WTO all dynamic allocation parameters	RADBDD22
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

CLOSE = ddname

specifies the DD name to be closed in the DTF.

DDNAME = (ddname, nn)

specifies a DD name related to a requested trace. All trace information generated as a result of the BITS.ON setting is directed to the DDNAME indicated in the parameter list, based on the CONNECT:Direct TASKID number *nn*. This provides a consolidated trace of all activity associated with the task. The DDNAME format is R00000*nn*, where *nn* is the TASKID.

DYN = ′dynamic allocation string′ (batch)
specifies that dynamic allocation is invoked in the DTF by use of a specified allocation string. The parameter DYN (batch) has an equal sign before and quotes around dynamic allocation string.

DYN dynamic allocation string (IUI)
specifies that dynamic allocation is invoked in the DTF by use of a specified allocation string. The parameter DYN (IUI) does not have an equal sign before or quotes around dynamic allocation string.

SESSIONS = Quiesce | Resume
controls the automatic establishment of DTF-to-DTF sessions.

Quiesce specifies that no new DTF-to-DTF sessions are started after executing Processes complete. Interactive users can sign on. Any Process that would normally be executed is placed in the WAIT queue.

Resume terminates a quiesce state and returns CONNECT:Direct to normal operation.

LOG.PRINTER = OFF | ON | luname
specifies whether the log printer is turned off or on or whether the LU name changed. If a new LU name is specified, the old log printer is disconnected and the new log printer is attached.

MODDIR.TRACE=YES
requests a module trace.

Batch Interface Use of MODIFY

To use this command from the Batch Interface, perform these steps:

1. Place the MODIFY commands in the DMBATCH job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.

2. Ensure that CONNECT:Direct is running.

3. Submit the job and verify the modifications.

Batch Interface Examples of the Modify Command

This section illustrates the use of the MODIFY command in the Batch Interface. The following command terminates a quiesce state and returns CONNECT:Direct to normal operation.

```
MODIFY SESSIONS = RESUME
```

The following command turns on the short RPLOUT trace.

```
MODIFY BITS.ON = X'00000040'
```

The following command turns off the short RPLOUT trace.

```
MODIFY BITS.OFF = X'00000040'
```

This command invokes dynamic allocation in the DTF to the allocated DDNAME RPLERRCK.

```
MODIFY DYN = 'DD=RPLERRCK'
```

IUI Use of MODIFY (TRACE) COMMAND

You can perform the same tasks in the CONNECT:Direct IUI that you can perform with the Batch Interface.

To use the MODIFY command features in the CONNECT:Direct IUI, perform the following steps:

1. Access the MODIFY (TRACE) COMMAND screen by selecting option **MD** from the Administrative Options Menu.

```
node.name             MODIFY (TRACE) COMMAND                hh:mm
CMD ==>

    MODIFY BITS.ON       ==> _________     (nnnnnnnn)
    MODIFY BITS.OFF      ==> _________     (nnnnnnnn)
    MODIFY DDNAME        ==> ___________   (ddname,nn)
    MODIFY CLOSE         ==> _________     (ddname)
    MODIFY MODDIR.TRACE  ==> ___           (YES)
    MODIFY LOG.PRINTER   ==> _________     (OFF/ON/luname)
    MODIFY DYN           ==> ____________________________________
    MODIFY SESSIONS      ==> _             (Quiesce or Resume)
```

2. Enter the values in the appropriate fields. For a description of valid values of the parameters, refer to the definitions beginning on page 2-17 or press **PF1** to use the online help.

3. Press **ENTER** to execute the MODIFY command after you have completed your entries.

Stopping CONNECT:Direct

The STOP CD command stops CONNECT:Direct operation by means of a forced, immediate, quiescent, or step-wise shutdown. A message informs you of the pending shutdown, except when a forced shutdown is used.

Note: The STOP CD command is usually issued for system maintenance.

The STOP CD Command Format

The STOP CD command has the following format and associated parameters. The default value for the parameter is underlined.

Label	Command	Parameters
(optional)	STOP CD	[Force | Immediate | Quiesce | Step]

Force

stops CONNECT:Direct by means of a user 4095 abend, and produces a sizable dump. Use this option only in a problem situation.

Immediate

specifies that all active transmissions be terminated immediately. CONNECT:Direct writes the statistics record, closes the files, and shuts down. All Processes resume execution when CONNECT:Direct is reinitialized.

If a Process was checkpointed and CONNECT:Direct stops using this parameter, when the Process resumes execution, CONNECT:Direct repositions from the last checkpoint and resumes transferring data.

Quiesce

specifies that all active transmissions continue to run until all steps of all executing Processes are complete. No new transmissions are started, and no additional Processes are accepted. Quiesce is the default.

All interactive sessions are terminated except for the issuer of the STOP CD command. All Processes currently running must complete, and the issuer of the command must sign off before the CONNECT:Direct operation stops.

Step

specifies that all active transmissions run until the current Process step of each executing Process is complete. CONNECT:Direct then writes the statistics records, closes the files, and shuts down. All Processes resume execution when CONNECT:Direct is reinitialized.

Batch Interface Use of STOP CD

To use the STOP CD command from the Batch Interface, perform the following steps:

1. Place the STOP CD command in the DMBATCH job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.
2. Ensure that CONNECT:Direct is running.
3. Submit the job.
4. Verify the results.

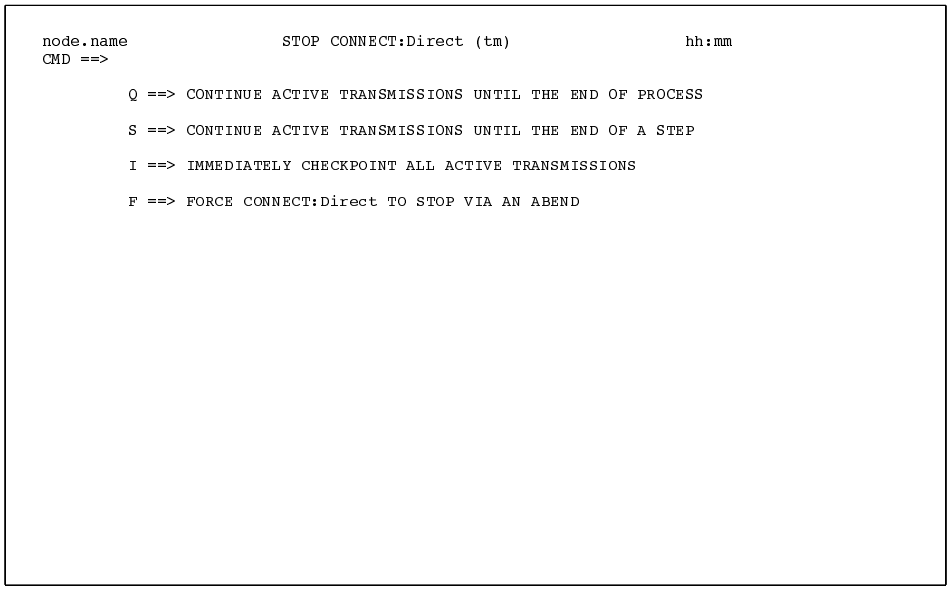
This example illustrates the use of the STOP CD command in the Batch Interface. The following command stops CONNECT:Direct, with the IMMEDIATE parameter.

```
STOP CD  I
```

Using STOP CD Through the IUI

To issue this command through the IUI, perform the following steps:

1. Access the Stop CONNECT:Direct screen, select option **SN** from the CONNECT:Direct for Administrative Options Menu.



2. Make your selection by typing in any of the four options on the command line, and press **ENTER**.

Refer to the description of the valid parameters of the STOP CD command beginning on page 2-21. Quiesce (Q) is the default.

Chapter 3

Controlling Security

CONNECT:Direct security can vary in different environments from no security support to a total security package controlling access to all data. CONNECT:Direct provides a range of options to meet security requirements in different environments.

Some of these options are built into CONNECT:Direct itself, some are provided as interfaces to other security software, some are supplied as sample exits, and some are available from user-customized exit routines.

This chapter explains how security works with CONNECT:Direct and provides the information to plan and install CONNECT:Direct security in your particular environment. It includes discussions of the following:

- Planning for security
- Security exits invoked during processing
- Implementing security exits
- CONNECT:Direct Secure Point-of-Entry
- Security system requirements

Planning for Security

CONNECT:Direct supports signon security checking through the CONNECT:Direct Authorization Facility and through security exits interfacing with CA-ACF2, Resource Access Control Facility (RACF), and VMSECURE. Any of these packages can be used to control a user's access to CONNECT:Direct functions.

CONNECT:Direct security support includes but is not limited to:

- Files
  - CA-ACF2
  - RACF
  - VMSECURE
- Application programs
  - Run Task Exit
- Users
  - CONNECT:Direct Authorization Facility
  - CA-ACF2
  - RACF
  - VMSECURE
- CONNECT:Direct functions
  - CONNECT:Direct Authorization Facility
  - CA-ACF2
  - VMSECURE

Using SECURITY.TYPE Initialization Parameter to Provide Minidisk Access Control

The SECURITY.TYPE initialization parameter is used in conjunction with the SECURITY.EXIT value to provide minidisk access control.

CONNECT:Direct for VM/ESA uses the Alternate Userid interface of VM to provide minidisk access control.

- For RACF: The CONNECT:Direct DTF userid must be designated as a surrogate controller through the PERMIT command.
- For CA-ACF2: The CONNECT:Direct DTF userid must be defined as one of the VM batch service machines in the CA-ACF2 SRVMOPTS macro. In addition, the CONNECT:Direct DTF userid must have an @SRF macro located in the ACFFDR. The ACFGCS LOADLIB must be available during execution of CONNECT:Direct for VM/ESA.
- For VMSECURE: The CONNECT:Direct DTF userid must be granted authorization to function as a batch service machine by

adding the following statements to the VMSECURE configuration file:

```
GRANT SURROGATE TO userid
GRANT DIAGPCHK TO userid
```

Using the SECURITY.EXIT Initialization Parameter to Specify Stage 2 Security Exits

CONNECT:Direct provides the SECURITY.EXIT initialization parameter so that a Stage 2 security exit can be specified. This exit is invoked during processing of the Signon command and Process start and data set access.

With a Stage 2 security exit, when a request is made to CONNECT:Direct for signon or file access, the request is passed directly to the security exit for authorization checking. See the *CONNECT:Direct for VM/ESA Administration Guide* for a complete description of the SECURITY.EXIT parameter.

The CONNECT:Direct sample library provides security exit routines that can be used with CA-ACF2, RACF, and VMSECURE.

Note: Assembler H or High-level Assembler is required to assemble the sample security exits.

Authorization Facility

For installations with no security packages installed, you can use the CONNECT:Direct Authorization Facility. If the SECURITY.EXIT initialization parameter is not specified, or is commented out, the CONNECT:Direct Authorization Facility will be used for signon security and assigning functional authority within CONNECT:Direct.

The CONNECT:Direct Authorization Facility provides no data set access security checking. Please see the *Maintaining User Authorization* chapter beginning on page 4-1 for additional information about User Authorization.

Security Exits Invoked During Processing

Before you can begin to plan implementation of CONNECT:Direct security, you should understand your host security environment and security system. You should also understand the flow of work through

the CONNECT:Direct system and the processing flow when security controls are invoked.

There are three CONNECT:Direct security exits:

- Stage 1 Signon Security exit
- Stage 2 Security exit
- Run Task Security exit

Processing Flows for Security Exits

CONNECT:Direct has two major processing flows through which allow you to invoke these security exits:

- SIGNON command sequence
- Process execution sequence

SIGNON Command Sequence

This command sequence is the first flow through which a terminal user, console operator, or batch application gains access to CONNECT:Direct functions. One or more of the following control points is invoked:

- Stage 1 Signon Security exit
- CONNECT:Direct Authorization Facility
- Stage 2 Security exit

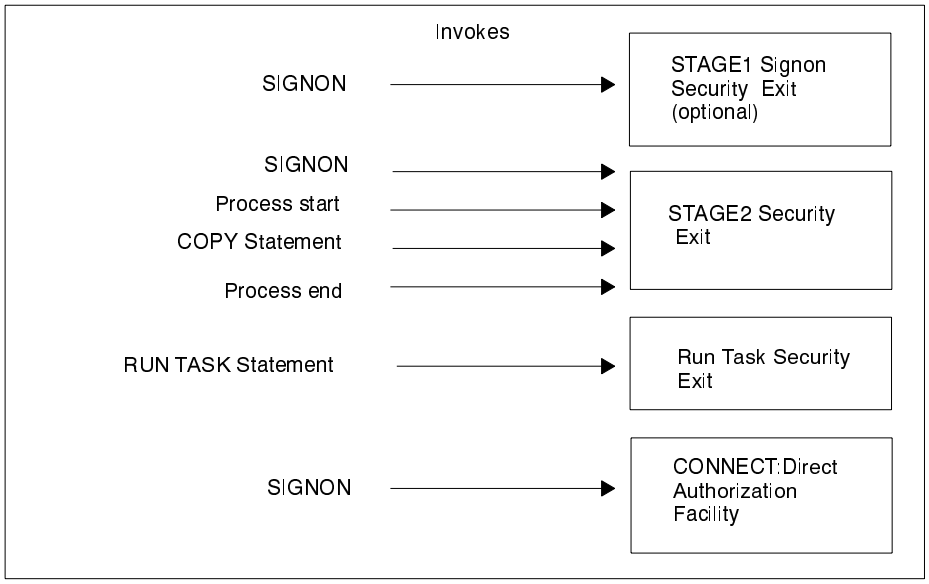
Process Execution Sequence

The Process execution sequence is the second flow through which CONNECT:Direct services execute on behalf of a user request. One or more of the following control points is invoked:

- Process start invokes the Stage 2 Security exit
- CONNECT:Direct Copy statement invokes the Stage 2 Security exit
- CONNECT:Direct Run Task statement invokes the Run Task Security exit
- Process end invokes the Stage 2 Security exit
- CONNECT:Direct Run Job statement invokes the Stage 2 Security exit or the Run Job security exit or both

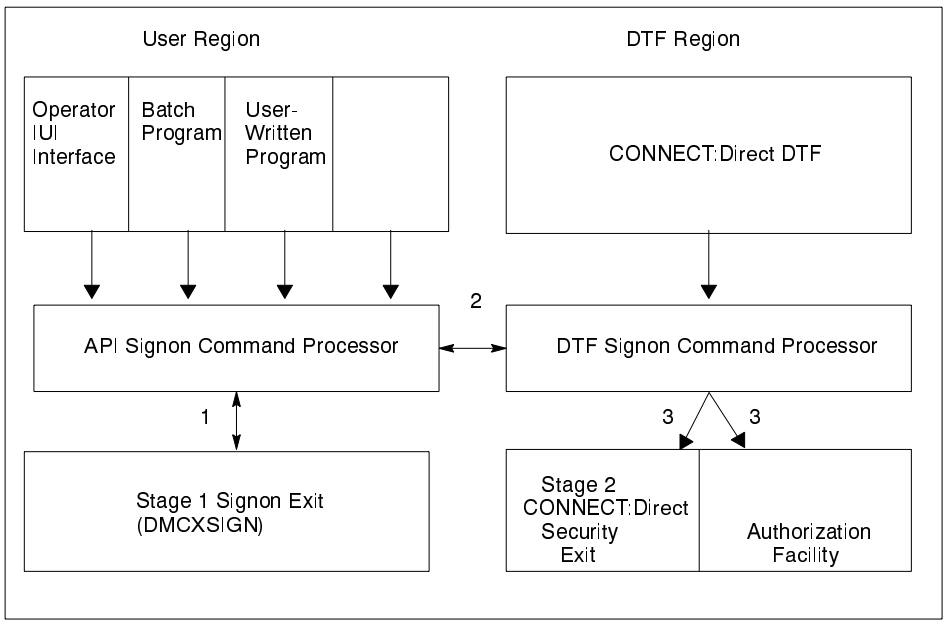
Example of SIGNON Command Sequence and Process Execution Sequence

This figure illustrates both the CONNECT:Direct SIGNON command sequence and the Process execution sequence.



Security During Signon Command

When you execute a Signon command (either through the batch, interactive, or operator interface), there are security control points in both the CONNECT:Direct user region or API and the CONNECT:Direct DTF region. As shown in the following figure, the initial control point is the Stage 1 Signon security exit.



The Stage 1 security checkpoint allows verification of the format and contents of the Signon command. This optional control point is in the form of a user exit that gains control in the user region. The exit can inspect and modify the CONNECT:Direct Signon command parameters. Security checks are performed in both the CONNECT:Direct CMS machine of the user and the CONNECT:Direct DTF GCS machine.

The next control point occurs in the DTF region and can take the form of a Stage 2 Security exit or the CONNECT:Direct Authorization Facility.

Understanding the SIGNON Command Flow

Refer to the numbers in the previous figure on page 3-5 as you trace the SIGNON command flow described in this section.

1. When you issue a CONNECT:Direct SIGNON command, the API SIGNON command processor calls the Stage 1 Signon exit. If the Stage 1 exit is not found, normal signon processing continues.

   When invoked, the Stage 1 exit receives a pointer to the CONNECT:Direct User Interface Control Block (UICB) that contains information regarding the signon attempt.

   If you specified a password on the SIGNON command, the Stage 1 exit returns control to CONNECT:Direct without making any modifications to the UICB, and the signon processing proceeds. In this case, the Stage 2 exit will verify the USERID and PASSWORD that were coded on the SIGNON command for system entry validation and all subsequent security calls.

   If you did not specify a password on the SIGNON command, CONNECT:Direct will extract the USERID from the security system control block that has already been built for this address space and put that USERID into the UICB.

   Note: It is important to remember that the Stage 1 exit keys off the password, not the userid. So, if a PASSWORD has not been specified but a USERID has, the Stage 1 exit will ignore that userid and overlay it with the address space USERID that is picked up from the security system control block.

   Once the USERID has been moved to the UICB, the exit will fill in a special password of IUI, BATCH, or STC. This depends upon what environment the signon came from (because CONNECT:Direct cannot access the PASSWORD for the address space USERID), and control is returned to CONNECT:Direct.

The benefit of running with a Stage 1 Signon Exit is that CONNECT:Direct batch jobs need not have hard-coded passwords in their SYSIN data streams.

Note: As the sample stage 1 exit is shipped, the dummy passwords of IUI, BATCH, and STC are coded in the exit. You should change these passwords for each installation, to avoid the chance of another site using the same dummy passwords. You can change these passwords by editing the source for DMCXSIGN and the appropriate validation in the macro DMGSECUR (for the stage 2 exit).

---

2. If the Stage 1 Processing is successful, the API Signon command processor passes the SIGNON command to the DTF where the DTF Signon command processor is invoked.

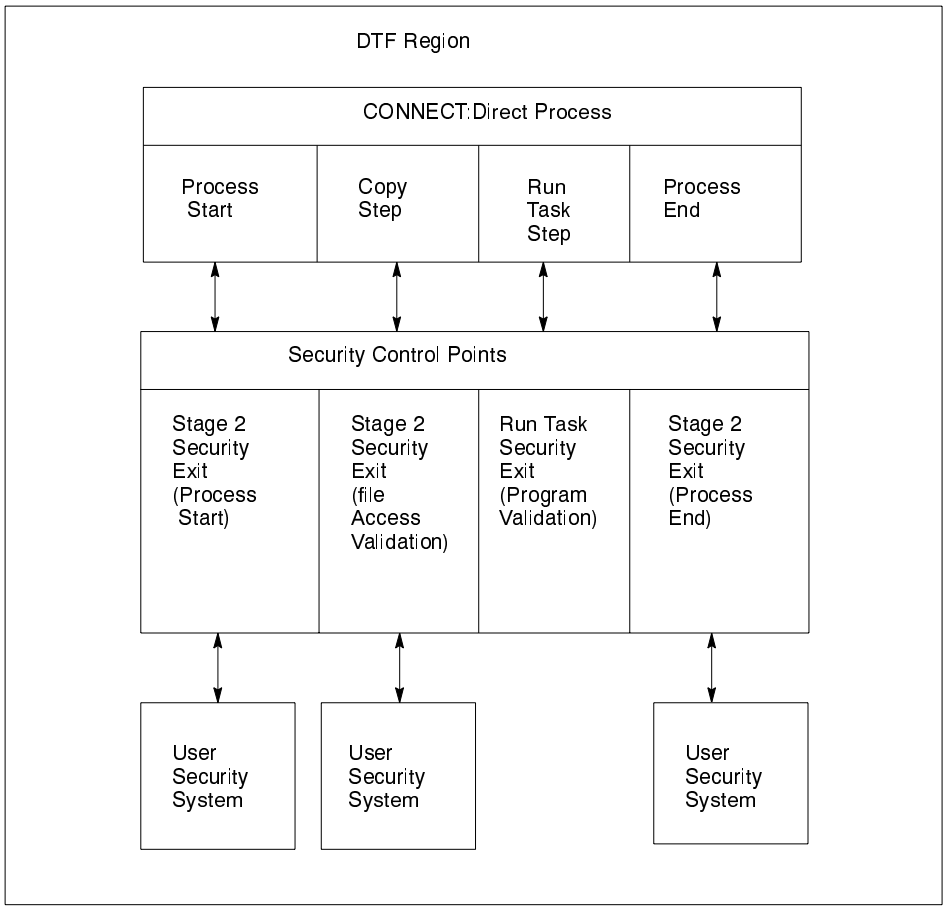
3. The DTF Signon command processor calls the Stage 2 Security exit or the CONNECT:Direct Authorization Facility.

   The Stage 2 exit recognizes special passwords of IUI, BATCH, and STC as having been assigned by the Stage 1 exit, and all calls to the security system for verifications will verify authorizations by USERID only.

   Regardless of your system implementation, this processing flow verifies the requesting authority of the user to perform CONNECT:Direct functions by checking the ABM (Authorization Bit Mask) for this user. The ABM is built through the Stage 2 Security Exit or through the CONNECT:Direct Authorization Facility at signon and Process start.

Security During Process Execution

When CONNECT:Direct executes a Process on behalf of a user, there are several Stage 2 security control points and a Run Task security checkpoint in the DTF, as shown in the following figure.



1. **Process start**—This point in the stage 2 security exit gains control whenever a Process begins initial execution or restart execution and allows verification of the authority of the requesting user to perform the CONNECT:Direct functions contained in the Process.

2. **File access**—This point in the stage 2 security exit gains control during Process execution whenever a COPY statement is encountered. It allows verification of the access for the user to read or write the file defined in the COPY statement.

3. **Run Task**—This exit point allows for program validation and gains control when these conditions exist:

   - RUN TASK statement is encountered during Process execution
   - RUN.TASK.EXIT initialization parameter is specified

4. **Process end**—This point in the stage 2 security exit gains control whenever a Process terminates, whether normally or abnormally. This exit point assists in cleaning up the security resources involved in Process execution.

Note: Copy and Run Task exit functions are entered for every occurrence of the associated statement in a CONNECT:Direct Process.

Implementing Security Exits

CONNECT:Direct provides a number of options for implementing security. The choice is generally dictated by the environment in which CONNECT:Direct is being installed. Since some options will only work in certain environments and other options are mutually exclusive, each of the control points described earlier is listed here with the applicable environments and the implementation details.

Note: ASSEMBLER H or High-level Assembler is required to assemble the sample exits.

Stage 1 Signon Security Exit

The primary function of the Stage 1 exit is to provide a CONNECT:Direct userid and password if the user does not provide one when signing on to CONNECT:Direct for VM/ESA. The Stage 1 Signon is a user-supplied exit that gains control in the GCS machine running the GCS API when a Signon command is processed. This security checkpoint is applicable to all environments.

This control point allows verification of the format and contents of the Signon command.

- The CONNECT:Direct Stage 1 Signon exit is implemented as an executable load module.
- The name of the load module must be DMCXSIGN. DMCXSIGN is invoked in the GCS machine running DMGCSAPI and must be available in the LOADLIB for that machine.
- The CONNECT:Direct LOADLIB contains a sample source module exit called DMCXSIGN with the filetype ASSEMBLE. The sample

source may be edited to reflect your security requirements. The exit must then be assembled and link-edited.

Note: All CONNECT:Direct nodes involved in cross-domain signon (or CONNECT:Direct multi-session signon) with a CONNECT:Direct node that uses the Stage 1 Signon exit must also use the Stage 1 Signon exit.

Stage 2 Security Exit

This control point is applicable to all environments and is implemented as a user-supplied exit. It provides a standard interface for user ID and password verification and for establishing CONNECT:Direct functional authority in the CA-ACF2, RACF, and VMSECURE environments, as well as file access verification in CA-ACF2. While it can be used for many different purposes, the Stage 2 Security exit is designed to provide the interface to your security system.

- The CONNECT:Direct Stage 2 Security exit is implemented as an executable load module.
- The name of the load module is user-definable, but it must not conflict with any CONNECT:Direct load modules.
- Activation of the Stage 2 Security exit is achieved by specifying SECURITY.EXIT (modname,ALL) in the CONNECT:Direct initialization parameters. This initialization parameter also specifies whether the exit is to be used for ALL security checking or just DATASET access validation. (ALL is recommended.)
- The module must be link-edited re-entrant and reusable and placed in a load library that can be accessed by the CONNECT:Direct DTF.
- On systems with 31-bit addressing, the module should be link-edited with AMODE 31 and RMODE 24 and be capable of executing in 31-bit mode unless ALLOC.STORAGE=BELOW is specified in the DTF initialization parameters. This is because information passed to the exit by CONNECT:Direct is located above 16 megabytes in this case.
- The CONNECT:Direct for VM/ESA distribution tape contains a sample source code for the CA-ACF2, RACF, and VMSECURE security systems. These sample routines all invoke a common macro called DMGSECUR. This macro is the actual source code for the sample exits and is conditionally assembled based on the security system in use. Other systems can be accommodated by using the sample code as a model.

Supplied exit routines include:

- DMGACF2 for CA-ACF2
- DMGRACF for RACF
- DMGVMSE2 for VMSECURE
- DMGVMSEC for VMSECURE (Alternative Method)

Note: RACF and VMSECURE provide file access control at the minidisk level only. CA-ACF2 provides file access control at the file level or the minidisk level. If you are implementing CA-ACF2 on VM/SP Release 4, select minidisk level access control at a minimum. For more information, refer to the *Exit Implementation in the CA-ACF2 Environment* section on page 3-33 of this chapter for the description of the LINKCHK parameter.

Stage 2 Exit Parameters

Each supplied module is similar in content. Edit the module to select the appropriate parameters. The parameters are described in each source module and are summarized here.

TYPE=[ACF2 | RACF | VMSECUR]
identifies the type of exit you want to use.

STAGE1=[YES | NO]
identifies whether the Stage 1 Signon exit is implemented.

The STAGE1 parameter must be consistently defined on all nodes that might communicate with a particular node. For example, if STAGE1=YES is selected for NODEA, all nodes that will communicate with NODEA must be defined as STAGE1=YES.

Note: If STAGE1 is not consistently defined within a CONNECT:Direct network, errors will be generated during multiple session signon and process execution.

NEWPASS=[YES | NO] (CA-ACF2 only)
specifies whether a user can request that his or her security system password be changed, either on the SIGNON command or on PNODEID or SNODEID values on the Process statement or Submit statement.

PNODEID=[YES, | NO]

specifies whether this exit will allow security override of a PNODE userid if specified on the CONNECT:Direct Process statement when submitting the Process.

SNODEID=[YES | NO]

specifies whether this exit will allow an incoming node to use an SNODEID. (If Point-of-Entry security is in use, specify NO.)

LINKCHK=[YES | NO] (CA-ACF2 and VMSECURE)

specifies whether this exit will verify LINK level access to the specified minidisk. Specify LINKCHK=YES if you are implementing CA-ACF2 on VM/SP Release 4. If you are implementing CA-ACF2 on VM/SP Release 5 or later, you do not need to specify the LINKCHK parameter. In VM/SP Release 5 or later, CONNECT:Direct uses the alternate ID function of VM to issue the LINK request under the requesting user's ID. If the user is not authorized, the LINK fails.

Note: If LINKCHK=NO is specified, the CONNECT:Direct AUTH file will be used to grant authority. If LINKCHK=YES is specified, then AUTHFIL must be coded as NO. The CONNECT:Direct DTF machine must be given the authority to issue the VMSECURE CAN command.

For example, add the GRANT command in the AUTORIZ CONFIG file in the VMSECURE service machine:

```
GRANT CAN TO *ALL
GRANT NOPASS CAN TO userid
```

In the previous example, userid is the CMS id for the CONNECT:Direct DTF machine.

VMSECID=vmsecure id (VMSECURE only)

specifies the CMS machine ID of the VMSECURE service machine.

GCSAPI= dtf id(VMSECURE only)

specifies the CMS ID of the CONNECT:Direct DTF machine.

ADMVAL=ccuu (VMSECURE only)
specifies the virtual minidisk address for ADMIN.

OPRVAL=ccuu (VMSECURE only)
specifies the virtual minidisk address for OPERATOR.

GENVAL=ccuu (VMSECURE only)
specifies the virtual minidisk address for GENERAL USER. A value of DEFAULT causes GENUSR authority to default if the first two checks fail.

FILECHK=[YES | NO] (CA-ACF2 only)
specifies whether this exit will perform CMS file level access checking. If FILECHK=YES is selected, access checking will be active even if CMS file access checking is disabled in CA-ACF2.

TEST=[YES | NO]
special debugging option. TEST specifies whether the exit will provide a security exit trace or not. If the exit is assembled with TEST=YES specified, and a FILEDEF of SECURITY is added to the CONNECT:Direct start-up job stream, the exit will produce a trace. The trace will include information passed to the exit by CONNECT:Direct, information passed to the security system, and the feedback from those calls.

The SECURITY FILEDEF can be directed to the terminal or a disk file.

Caution: Use this parameter only during testing of the exit because the output produced will include userids and passwords in clear text.

Levels of CONNECT:Direct Functional Authority

There are three different levels of CONNECT:Direct functional authority. The following explains the system access allowed for each level of CONNECT:Direct functional authority:

Administrator
authorizes performance of all CONNECT:Direct statements and commands.

Operator

authorizes DELETE, CHANGE, DISPLAY, FLUSH and SUBMIT CONNECT:Direct Processes; STOP CONNECT:Direct; START and STOP traces; and DISPLAY, ADD, DELETE and UPDATE TYPE.

General user

authorizes DELETE, CHANGE, DISPLAY, and FLUSH his or her own CONNECT:Direct Processes; SUBMIT CONNECT:Direct Processes; and DISPLAY, ADD, DELETE and UPDATE Types.

Note: See page 3-20 for a description of how the options in the security exits can be selected to establish CONNECT:Direct functional authority for different CONNECT:Direct users.

CONNECT:Direct Secure Point-of-Entry

Because adjacent nodes need access to local nodes in order to transfer files, security on a local node is a concern. (Secure point-of-entry translation occurs at Process execution only.) CONNECT:Direct administrators have three options for CONNECT:Direct security on transfers initiated at a remote node:

- No security for either functional authority or data protection
- Matching userid-password combinations for all adjacent nodes
- SNODEID-SNODE password overrides on incoming access requests

When implemented, point-of-entry security works with your current security setup (including all current exits) to provide additional security that satisfies concerns about users from other nodes knowing a userid-password combination on your system.

Both data protection and CONNECT:Direct functional authority are handled with exits. Point-of-entry security exists only to further secure the entry of an *outside* user to your system.

Understanding the Point-of-Entry Concept

When a Process is submitted by another node, the receiving CONNECT:Direct has access to the userid of the person who submitted the process and the name of the node on which the process was submitted.

Example of the Point-of-Entry Concept

For example, your node is CD.VALLEY.FORGE, and a user LILY submits a process on CD.SOMERSET to copy a file to CD.VALLEY.FORGE. By placing an entry of LILY/CD.SOMERSET into the local CONNECT:Direct authorization file, the security administrator at Valley Forge can associate this user with a valid userid-password on the local system.

In this example, the CONNECT:Direct Authorization File appears as:

```
USERID         = LILY
NODE           = CD.SOMERSET
SECURITY ID    = CICERO
SECURITY PSWD  = DALLAS
```

In this scenario, the user from the Somerset node never needs to know a valid userid-password on the Valley Forge node.

Note: Point-of-Entry processing is internal to CONNECT:Direct and happens prior to calling the Security Exit for validations.

In the previous example, the following actions occur:

- LILY on the CD.SOMERSET node submits a Process to run with node CD.VALLEY.FORGE
- the functional authority and the data set validation for that process will be done under the authority of userid CICERO, which is a valid userid on CD.VALLEY.FORGE.

Example of a System Without a Secure Point-of-Entry

In order to produce a completely secure point-of-entry security system, it is best to disallow SNODEID overrides. To disallow SNODEID overrides, specify SNODEID=NO in your Stage 2 Security Exit.

For example, if an SNODEID=(NOLAN,RANGERS) is coded in the Process submitted by LILY from CD.SOMERSET, the Valley Forge node will do the security validation with the authority of userid NOLAN, not userid CICERO.

In this scenario, the following actions occur if the Valley Forge node allows SNODE overrides:

- user LILY on the CD.SOMERSET node submits a Process with an SNODEID parameter to run with node CD.VALLEY.FORGE

- the Valley Forge Authorization File will not be checked and the translation of the userid-password will not be done

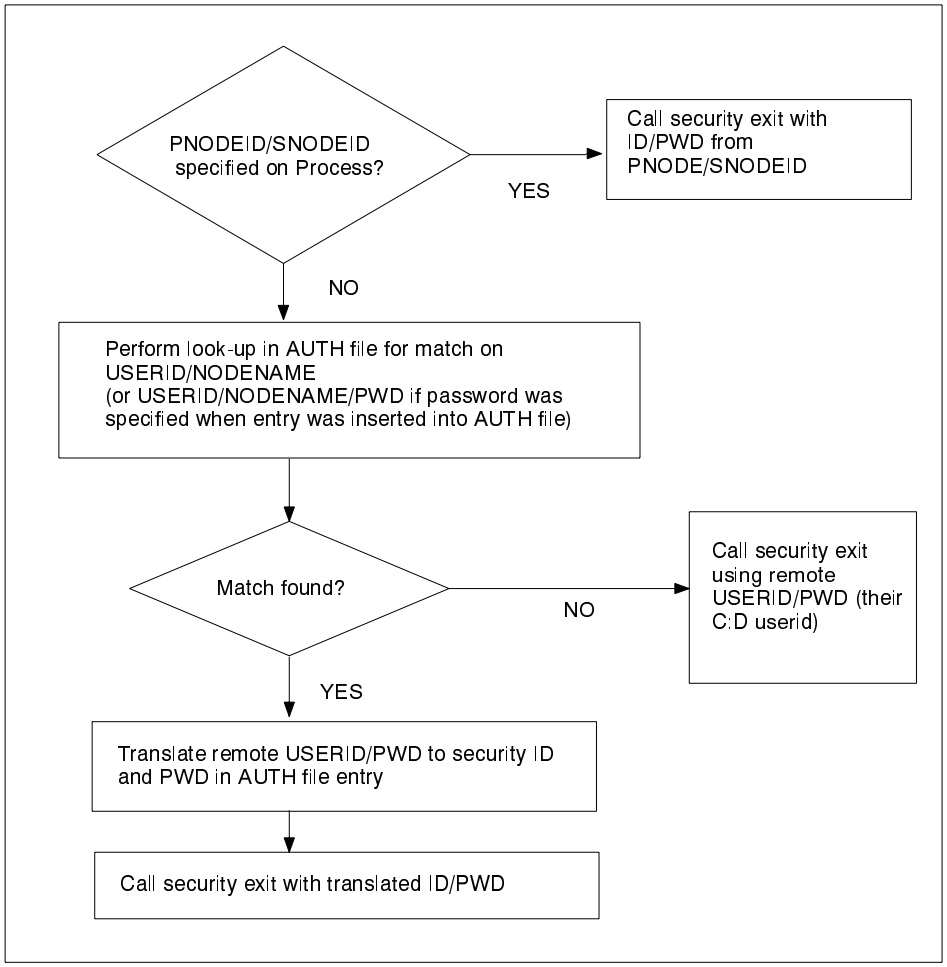
Maintaining the Point-of-Entry System

Although the point-of-entry system needs some maintenance of the Security ID and Security Password fields in the Authorization File, you can assign the same userid/password combination on your system to multiple incoming users.

For instance, CICERO/DALLAS could be the userid/password for all incoming users from CD.SOMERSET. If you are running a Stage 1 Signon exit, you could specify the security password for all users as IUI, BATCH, or STC, and avoid updating the Authorization File as passwords change.

Note: CONNECT:Direct for VMS is not able to pass a VMS password with the VMS userid. If you are using secure point-of-entry with incoming CONNECT:Direct for VMS nodes, you must leave the User /Password field blank in your Authorization File, or all incoming VMS Processes will fail.

This figure illustrates the security checking for secure point-of-entry.



Implementing Secure Point-of-Entry

CONNECT:Direct automatically checks every incoming Process. No parameter is required to activate point-of-entry security. To implement secure point-of-entry, you must add userid-nodename combinations to your Authorization File.

For instructions on manipulating the CONNECT:Direct Authorization File, refer to the *Maintaining User Authorization* chapter beginning on page 4-1.

Security for Specific Nodes

These sections describe security control you can implement at the node level.

Trusted Node Security

The Trusted Node Security feature permits you to enforce more restrictive security parameters when dealing with specific nodes in your network. This allows you to define each adjacent node in the Network Map to be *internal* or *external* in its relationship to the local node of that Network Map.

When a Process begins execution, the security exit gets control and a bit in the Security Control Block (SQCB), SQEXTNOD, turns on if the adjacent node is defined as EXTERNAL. If the adjacent node is defined as INTERNAL, the bit turns off. Based on this information, the security exit can take the appropriate action.

In the adjacent node definition, the fifth positional parameter is required for the Trusted Node Security option. The parameter description follows.

EXTERNAL|EXT
indicates this is an external node.

INTERNAL|INT
indicates this is an internal node. This is the default.

Data Direction Restriction

In addition to the Trusted Node feature, the Data Direction Restriction specifies whether each adjacent node can initiate a RECEIVE, SEND, or RECEIVE and SEND to or from the local node in the Network Map. The bits located in the SQCB, SQRECV, and SQSEND indicate the sending and receiving status. Reassemble the security exit with the versions of DMGSECUR and DMFSQCB that use the Data Direction Restriction option.

In the adjacent node definition, the sixth positional parameter is used to restrict the direction of data on a transfer with a specific adjacent node. This security applies regardless of where the Process is submitted, for example, local or remote node. The parameter description follows.

RECEIVE|RECV
indicates this adjacent node is allowed to receive data from the local node. It will not be allowed to send data to the local node.

SEND
indicates this adjacent node is allowed to send data to the local node. It is not allowed to receive data from the local node.

BOTH

indicates this adjacent node is allowed to send data to the local node and receive data from the local node. This is the default.

NONE

indicates this adjacent node will not be allowed to send data to the local node or receive data from the local node.

Trusted Node and Data Direction Restriction Example

This figure represents the Trusted Node Security and Data Direction Restriction features defined in the Network Map. The parameters are the fifth and sixth positional parameters in the adjacent node definition.

```
LOCAL.NODE=(CD.LOCAL LOCAPPL,,SUPUSRPW) -
      TCQ=(CD.TCX CD.TCQ))
ADJACENT.NODE=(PARSESS=(4 2)    (CD.LOCAL LOCAPPL -
      APPLIDS=(1011  1012  1013))
ADJACENT.NODE=(PARSESS=(4  2) -
      (CD.REMOTE  RMTAPPL , , , EXTERNAL,RECV) -
      APPLIDS=(1011 1012 1013))
```

SQSNODE and SQIDXLAT Bits

The following two bits are identified in the security exit:

- The SQSNODE bit is checked to see if the node where the security exit is running is the SNODE for this Process. The bit is *on* if the node is the SNODE and *off* if the node is the PNODE.
- The SQIDXLAT bit identifies if a point-of-entry security ID translation was performed prior to calling the security exit. If a PNODEID/SNODEID is not specified when the Process is submitted and a match is found in the CONNECT:Direct Authorization File for that USERID and NODE combination, then the bit turns *on* when the security exit gets control.

Cross-Domain Signon

An extension of the Trusted Node Security feature in the cross-domain signon environment is available. This feature allows you to easily identify whether a signon is entering from an *internal* or *external* node.

You can use the same Network Map parameters for cross-domain Trusted Node Security as the node-to-node Trusted Node Security enhancement, for example, EXTERNAL or INTERNAL in the adjacent

node definition. A new bit, SQEXTSGN, in the security exit Control Block (SQCB) designates this feature.

When a cross-domain signon is entering from a node defined as EXTERNAL in the local node Network Map definition, the SQEXTSGN bit is *on* when the security exit gets control during signon processing. The security exit can then be modified to take whatever action is appropriate for that installation.

The DMGSECUR macro, included in the SAMPLIB file, contains the code to implement this enhancement. The lines of code with 117200 identify the Trusted Node Security feature in the cross-domain signon environment.

Security System Requirements

Depending on the security system in use and the operating system environment, special considerations apply when implementing the security exits. The following section describes these considerations.

Stage 2 Security Exit Methods Used to Establish Functional Authority

User authority to perform specific CONNECT:Direct commands and Process statements is defined in the Stage 2 Security exit. For example, only operations personnel could be allowed to issue a STOP CONNECT:Direct command, while general users could be limited to submitting CONNECT:Direct Processes.

To provide a command limit hierarchy, each CONNECT:Direct command is associated with a specific bit representation in a 20-byte field. The security exit is responsible for building this authorization bit mask by filling in the SQABM field of the SQCB (Security Request Control Block). The SQCB is defined in the DMFSQCB member of the CONNECT:Direct for VM/ESA distribution MACLIB.

This table shows the four methods of establishing the authorization mask provided by the sample security exits and the parameters that are used to define how the exit will establish functional authority.

Method	Where Functional Authority Is Defined	Required Parameters	Applicable Security Environments
1	CONNECT:Direct Authorization file	AUTHFIL=YES DEFLTID=[userid | DEFAULT]	All
2	CA-ACF2 LIDREC parameter	LIDREC=YES LIDFLD=field name ADMVAL=character value OPRVAL=character value GENVAL=character value	CA-ACF2 only
3	User-defined procedure	AUTHFIL=NO LIDREC=NO	All
4	VMSECURE	LINKCHK=YES VMSECID=vmsecure id GCSAPI=CMS id of the DTF ADMVAL=ccuu OPRVAL=ccuu GENVAL=ccuu	VMSECURE only

These parameters are defined in the DMGSECUR macro, located in the CONNECT:Direct for VM/ESA distribution MACLIB, and specified in the appropriate security exit (DMGACF2, DMGRAC, or DMGVMSEC).

The following sections describe these methods and the exit parameters that allow the user to select the exit authorization desired from the previous table.

METHOD 1—All Environments

This method uses the CONNECT:Direct authorization file and is applicable in all security environments. It is selected by specifying AUTHFIL=YES in either the DMGACF2, DMGRACF, or DMGVMSEC security exit.

AUTHFIL=YES
specifies whether or not the CONNECT:Direct authorization file will be used to set CONNECT:Direct functional authority. If AUTHFIL=YES and LIDREC=YES are specified together, the Stage 2 exit will not assemble.

DEFLTID=[userid | DEFAULT]
specifies either a userid or a default reserved word used to set authorization. The DEFLTID parameter can only be specified with the AUTHFIL=YES parameter.

userid specifies a 1–8 character ID that is used to set authorization if the ID of the user who submitted the request is not found.

DEFAULT is a reserved word that, when used, causes the exit to set general user authority when a specific userid is not found in the CONNECT:Direct authorization file.

Determining CONNECT:Direct Functional Authorization with Method 1

The access to a record in the CONNECT:Direct authorization file, using Method 1, is based on a pairing of the nodename and userid. The security exit calls a module named DMGAUTH, and passes the address of the SQCB as the only parameter. This routine uses SQUNODE and SQUID to access the authorization file and fills in SQABM with the authorization values from the record in the CONNECT:Direct authorization file. Refer to the SETAUTH label in the appropriate security exit for more details.

If the userid is not found in the CONNECT:Direct authorization file, the exit will use the userid specified in the DEFLTID parameter. The action taken depends on the following conditions:

- If the value DEFAULT is specified in the DEFLTID parameter, the exit will set general user authorization. This will allow CONNECT:Direct access without searching the CONNECT:Direct authorization file.
- If any value other than DEFAULT is specified for the DEFLTID parameter, that value is used as the userid in the nodename / userid pair to search the CONNECT:Direct authorization file. If that record is found, the authorization from that record is supplied. If that record is not found, access to CONNECT:Direct is denied.

The following table shows how a sample CONNECT:Direct authorization file can be used to develop functional authorization.

Node	Userid	Authority
NODEA	ADMIN	Administrator
NODEA	OPER	Operator
NODEA	GENUSER	General user
NODEB	GENUSER	General user

The entries assume the following:

- Two CONNECT:Direct nodes, NODEA and NODEB, are involved. NODEA is the local node, and NODEB is the remote node that can initiate a file transfer with this node.

- NODEA has a defined administrator, operator and general user. NODEB, a remote system, has only a general user defined at this system.

- The specified DMGSECUR macro parameters are: AUTHFIL=YES and DEFLTID=GENUSER.

If, in the preceding example, a userid of ADMIN issues a Signon command to the local node, the user will receive a specific set of authorizations appropriate for an administrator. If the userid is not found for a NODEA user, the default value of GENUSER is used. Since NODEB only has a general userid, all Processes initiated from NODEB will have general user authority.

Refer to the *Insert User and Update User Commands* section beginning on page 4-2 for more information on maintaining the CONNECT:Direct Authorization Facility.

METHOD 2—CA-ACF2 Only

This method uses the CA-ACF2 LIDREC parameter and is applicable only in a CA-ACF2 security environment. It is selected by specifying LIDREC=YES, in addition to a series of subparameters, in the DMGACF2 security exit.

LIDREC=YES specifies the logon ID record will be used to set CONNECT:Direct functional authority. If AUTHFIL=YES and LIDREC=YES are specified together, the Stage 2 exit will not assemble.

The following are subparameters for the LIDREC parameter.

LIDFLD=field name
: specifies the name of a one-byte field in the LIDREC that will be used to determine CONNECT:Direct functional authority.

ADMVAL=character value
: specifies a character (C'c') or hex (X'xx') value in the LIDFLD that represents CONNECT:Direct administrator authority.

OPRVAL=character value
: specifies a character (C'c') or hex (X'xx') value in the LIDFLD that represents CONNECT:Direct operator authority.

GENVAL=[character value|DEFAULT]

specifies a character (C'c') or hex (X'xx') value in the LIDFLD that represents authority for a CONNECT:Direct user who is not defined as an administrator or operator. GENVAL can be specified as DEFAULT to cause any LID that does not match either ADMVAL or OPRVAL values to be granted general user authority. If a LID otherwise fails to match the ADMVAL, GENVAL or OPRVAL values, CONNECT:Direct access will be denied.

Note: The Stage 2 Security exit assumes that the LIDFLD name is a single-byte character field, and the ADMVAL, OPRVAL and GENVAL values are one-byte character or hexadecimal values.

If other representations of LIDREC parameter values are desired, the user must modify the security exit. See the LIDAUTH label in the DMGSECUR macro for details.

METHOD 3—All Environments

The third method uses a user-defined procedure and is applicable to all security environments. It is selected by specifying the AUTHFIL=NO and LIDREC=NO parameters in the appropriate security exit.

AUTHFIL=NO

specifies that the CONNECT:Direct authorization file will not be used to set CONNECT:Direct functional authority.

LIDREC=NO

specifies the logon ID record (LIDREC) will not be used to set CONNECT:Direct functional authority.

Note: The Stage 2 Security exit sets general user authority for all users if this option is selected. If some other authorization is desired, the exit must be modified. See the USERAUTH label in the DMGSECUR macro for details.

METHOD 4—VMSECURE

The fourth method is applicable to the VMSECURE environment. The DMGSECUR macro has been modified to allow setting CONNECT:Direct authorization on the authority of the user to access certain minidisk. By assigning ADMVAL, OPRVAL, and GENVAL and calling VMSECURE, the exit can determine if the user is allowed the following access:

- WRITE LINK to the ADMIN minidisk and grant ADMIN authority

- WRITE LINK to the OPER minidisk and grant OPER authority
- WRITE LINK to the GENUSER minidisk and grant GENUSER authority

The three minidisks can be dummy disks and VMSECURE RULES can be provided to define the authority for the user to LINK to the minidisk.

For example, with the following VMSECURE RULES defined in the CONNECT:Direct DTF machine, user QATEST1 would be granted ADMIN authority, user QATEST2 would be granted OPER authority, and all other users would granted GENUSER authority.

```
ACCEPT * LINK 9E3 W (NOPASS
ACCEPT QATEST LINK 9E1 W (NOPASS
ACCEPT QATEST2 LINK 9E2 W (NOPASS
REJECT * LINK 9E1 *
REJECT * LINK 9E2 *
```

The required parameters are as follows:

LINKCHK=YES
indicates that minidisk checking will be performed.

Note: If LINKCHK=NO is specified, the CONNECT:Direct AUTH file will be used to grant authority. If LINKCHK=YES is specified, then AUTHFIL must be coded as NO. The CONNECT:Direct DTF machine must be given the authority to issue the VMSECURE CAN command.

For example, add the GRANT command in the AUTORIZ CONFIG file in the VMSECURE service machine:

```
GRANT CAN TO *ALL
GRANT NOPASS CAN TO userid
```

VMSECID=vmsecure id
specifies the CMS machine ID of the VMSECURE service machine.

GCSAPI=dtf id
specifies the CMS ID of the CONNECT:Direct DTF machine.

ADMVAL=ccuu
specifies the virtual minidisk address for ADMIN.

OPRVAL=ccuu
specifies the virtual minidisk address for OPERATOR.

GENVAL=ccuu
specifies the virtual minidisk address for GENERAL USER. A value of DEFAULT causes GENUSR authority to default if the first two checks fail.

Tracing Security Exit Activities

In addition to the previously listed parameters, a special debugging parameter, TEST=YES, is defined in the DMGSECUR macro. If the exit is assembled with TEST=YES specified, and a DD name of SECURITY is allocated through a FILEDEF in the PROFILE GCS file of the DTF GCS machine, the exit will produce a trace of information passed to the exit by CONNECT:Direct, information passed to the security system and the feedback from those calls.

SECURITY can be defined through a GCS FILEDEF to DUMMY or a disk file. DUMMY causes the output to be displayed on the DTF console.

***Caution*:** This parameter should be used only during testing of the exit because the output produced will include userids and passwords in decrypted format.

Run Task Security Exit

The Run Task security exit control point is applicable to all CONNECT:Direct environments. It provides a standard interface to verify the user is authorized to run the specified program. The exit is passed security information about the user, the program name, and the parameters being passed to the program.

Specific implementation details include the following:

- The CONNECT:Direct Run Task exit is implemented as an executable load module.
- The name of the load module is user-definable but must not conflict with any existing CONNECT:Direct load module names.

- Activation of the Run Task exit is achieved by specifying RUN.TASK.EXIT=(modname) in the CONNECT:Direct initialization parameters.

- The module must be link-edited as re-entrant and placed in a load library that is accessed by the CONNECT:Direct DTF.

- The DMCXRT ASSEMBLE file contains a sample source exit. The sample exit can be used as a model to implement specific requirements. In this example, bold type represents ACF2 only.

```
/* REXX EXEC TO ASSEMBLE AND LINKEDIT CD SECURITY EXITS*/
TRACE O
PARSE ARG PROG
“STATE” PROG “ASSEMBLE”
IF RC <> 0 THEN DO
 SAY PROG “ASSEMBLE NO FOUND”
 EXIT
END
GLOBAL MACLIB CDV3200 DMSGPI DMSOM HCGPI MVSXA OSMACRO OSMACRO1
OSVSAM [ACF2VM]
“FILEDEF SYSIN DISK” PROG “ASSEMBLE A”
HASM ”(OBJECT NODECK RENT”
SAY PROG ”ASSEMBLY STARTED”
HLSAM “(OBJECT NODECK RENT”
SAY PROG “ASSEMBLY RC=” RC
IF RC <> 0 THEN EXIT
”FILEDEF SYSLIB DISK CDV3200 TXTLIB *”
”FILEDEF ACFGCS DISK ACFGCS TXTLIB *” (ACF2 ONLY)
”FILEDEF SYSLMOD DISK CDEXITS LOADLIB A (RECFM U”
SAY PROG “LKED STARTED”
”LKED” PROG ”(LET LIST XREF RENT REUS AMODE ANY RMODE 24”
SAY PROG “LKED RC=” RC
```

CONNECT:Direct Authorization Facility

The CONNECT:Direct Authorization Facility can control access to CONNECT:Direct functions and serve as the source of security information as an alternative to the Stage 1 Signon exit and Stage 2 Security exit. If the CONNECT:Direct Authorization Facility is used, all CONNECT:Direct users must be identified in all nodes that will participate in Process execution. See page 4-2 of the *Maintaining User Authorization* chapter for more information on the Authorization file.

Example of CONNECT:Direct Authorization Facility Use

The following example shows how the CONNECT:Direct Authorization Facility can be used. In this example there are two CONNECT:Direct nodes, called NODEA and NODEB. Joe has access to NODEA under the CONNECT:Direct userid of JOEA and access to NODEB under the CONNECT:Direct userid of JOEB.

Joe requires two entries in the CONNECT:Direct Authorization Facility of each system as shown in the following tables. This gives him access

to CONNECT:Direct on both systems and the authorization to move files between both systems.

The following table describes Joe's NODEA Authorization File:

Node	CONNECT:Direct Userid	CONNECT:Direct Password	Authorized Functions
NODEA	JOEA	[pswd]	Y,Y,N,Y,...
NODEB	JOEB	[pswd]	N,Y,...

The following table describes Joe's NODEB Authorization File:

Node	CONNECT:Direct Userid	CONNECT:Direct Password	Authorized Functions
NODEA	JOEA	[pswd]	Y,Y,N,Y,...
NODEB	JOEB	[pswd]	N,Y,...

The combination of logical node name and userid is used to access the authorization file on the remote node to obtain the userid and password, as well as associated functional authority.

For example, if Joe were to send a file from NODEA to NODEB, the combination of NODEA and JOEA would be used to access the authorization file on NODEB. This entry would then be used to determine what CONNECT:Direct functional authority Joe has on NODEB when coming from NODEA.

Note: The CONNECT:Direct password is optional, but if specified in the CONNECT:Direct Authorization Facility, it must also be specified on the CONNECT:Direct Signon command and be available at Process execution time through the signon or SNODEID override.

See the *Maintaining User Authorization* chapter beginning on page 4-1 for information on how to maintain the User Authorization File.

Note: Because of the extensive maintenance involved with this facility, it is not recommended for use unless no other security system is available.

Assembling and Linking CONNECT:Direct for VM/ESA Security Exits

This section describes instructions for assembling and linking the Stage 1 Signon exit or Stage 2 Security exit. Assembly and linkage of security exits can be done from CMS.

Security exit installation requirements for CA-ACF2, RACF, and VMSECURE are listed in the *Exit Implementation* sections later in this chapter. These include sample EXECs to assemble and link-edit the security exits.

Note: CONNECT:Direct sample signon and security exits require Assembler H or High-level Assembler for correct assembly.

Assembling a Stage 1 Signon Exit or Stage 2 Security Exit

This section contains the three commands required to complete assembly of the Stage 1 Signon exit or Stage 2 Security exit for CA-ACF2, RACF and VMSECURE.

1. Specify the macro libraries (MACLIBs) to be searched during execution of the assembly process, as in the following command. Prior to issuing this command, users must verify that their system is linked and accessed to all disks where the CONNECT:Direct for VM/ESA distribution MACLIB resides.

```
GLOBAL MACLIB CDVnnnn DMSGPI DMSOM HCPGPI MVSXA OSMACRO OSMACRO1
OSVSAM [ACF2VM]
```

GLOBAL MACLIB identifies that the MACLIB libraries are to be searched.

CDVnnnn is the name of the CONNECT:Direct distribution MACLIB, where *nnnn* is the release, modification and maintenance level of CONNECT:Direct for VM/ESA.

DMSGPI and **DMSOM** contains CMS macros.

HCPGPI contains GCS macros.

OSMACRO contains OS macros that CMS supports.

OSMACRO1 contains OS macros that CMS does not support.

OSVSAM contains VSAM macros.

[ACF2VM] (CA-ACF2 only) contains CA-ACF2 macros.

Note: The macro libraries in the preceding command must be entered in the order in which they appear to result in proper assembly.

---

2. Enter the following command to define the input file for the assembler.

```
FILEDEF SYSIN DISK EXIT ASSEMBLE A
```

FILEDEF SYSIN issues a data definition for the DDNAME SYSIN to be established.

DISK specifies that the I/O device is disk.

EXIT is the filename of the module being assembled.

ASSEMBLE is the filetype of the module being assembled.

A is the filemode of the disk where the source is located.

3. Finally, enter the following command to initiate the assembly of the Stage 1 Signon exit or Stage 2 Security exit:

```
HASM (OBJECT NODECK RENT
```

HASM is the H Assembler module name.

OBJECT (Assembler option) specifies that an object module will be created.

NODECK (Assembler option) specifies that the object module will be written directly to the the text file.

RENT (Assembler option) specifies that warnings should be generated for program re-enterability violations.

Upon completion of the assembly, the user should check either the assembler return code or the LISTING file to ensure that the modules have been correctly assembled.

Linking a Stage 1 Signon Exit or Stage 2 Security Exit

To link the assembled exits, three sets of commands must be entered for the RACF and VMSECURE exits, and four sets of commands must be entered for the CA-ACF2 exits.

1. First, the ddname SYSLIB must be defined for the linkage editor, as in the following command:

```
FILEDEF SYSLIB DISK CDVnnnn TXTLIB *
```

FILEDEF SYSLIB issues a data definition for the ddname SYSLIB to be established.

DISK specifies that the I/O device is disk.

CDVnnnn is the filename of the CONNECT:Direct distribution text library, where nnnn is the release, modification and maintenance level of CONNECT:Direct for VM/ESA.

TXTLIB is the filetype of the CONNECT:Direct distribution text library.

***** is the filemode of the CONNECT:Direct distribution text library.

For CA-ACF2 Only:

In addition to issuing the previous command, CA-ACF2 users must additionally define ACFGCS as input to the linkage editor when linking the Stage 2 security exit, as in the following command:

```
FILEDEF ACFGCS DISK ACFGCS TXTLIB *
```

FILEDEF ACFGCS issues a data definition for the DDNAME ACFGCS to be established.

DISK specifies that the I/O device is disk.

ACFGCS is the filename of the CA-ACF2 distribution text library.

TXTLIB is the filetype of the CA-ACF2 distribution text library.

***** is the filemode of the CA-ACF2 distribution text library.

2. Define the ddname SYSLMOD for the linkage editor. This is the name of the destination load library for linked modules:

```
FILEDEF SYSLMOD DISK CDEXITS LOADLIB A (RECFM U
```

FILEDEF SYSLMOD issues a data definition for the ddname SYSLMOD to be established.

DISK specifies that the I/O device is disk.

CDEXITS is the filename of the load library where the linked module will be placed.

LOADLIB is the filetype of the load library where the linked module will be placed.

A is the filemode of the load library where the linked module will be placed.

(RECFM U specifies the record format of the output load library is undefined.

3. Enter the following command to execute the linkage editor:

```
LKED EXIT (LET LIST XREF RENT
```

LKED is the command creating a load library member.

EXIT is the name of the object file to be processed. This must have a filetype of TEXT.

LET (Linkage editor option) specifies the link edit should complete even if invalid data is found in the input file.

LIST (Linkage editor option) specifies that control statements should be listed in the output file.

XREF (Linkage editor option) produces an external symbol cross-reference for the processed module.

RENT (Linkage editor option) specifies that warnings should be generated for program re-enterability violations.

Note: Successful execution of linkage will clear all temporary FILEDEFs. Each time a user links an exit, the DDNAMEs must be redefined.

Exit Implementation in the CA-ACF2 Environment

The following steps are required to implement the CA-ACF2 exit.

Step 1—Edit DMGACF2 ASSEMBLE

Edit DMGACF2 ASSEMBLE with the correct options for your CA-ACF2 environment. The following is a sample DMGACF2 exit.

```
               PUNCH 'INCLUDE ACFGCS(ACFDIAGG)'
               PUNCH 'INCLUDE ACFGCS(ACFBLMSG)'
               PUNCH 'INCLUDE ACFGCS(ACFVMCSI)'
               PUNCH 'INCLUDE ACFGCS(ACFBLRCK)'
               PUNCH 'INCLUDE ACFGCS(ACFBLCVT)'
               PUNCH ' ORDER DMGACF2'
DMGACF2 DMGSECUR TYPE=ACF2,                                    X
               STAGE1=YES,                                     X
               NEWPASS=YES,                                    X
               PNODEID=YES,                                    X
               SNODEID=YES,                                    X
               LINKCHK=YES,                                    X
               FILECHK=YES,                                    X
               LIDREC=YES,                                     X
               LIDFLD=LIDNDM,                                  X
               ADMVAL=C'A',                                    X
               OPRVAL=C'O',                                    X
               GENVAL=C'G',                                    X
               TEST=YES
          END
```

The DMGACF2 exit performs the following:

STAGE1=YES supports the Stage 1 Signon exit.

NEWPASS=YES allows users to change passwords through CONNECT:Direct.

PNODEID=YES supports the PNODEID overrides.

SNODEID=YES supports the SNODEID overrides.

LINKCHK=YES provides minidisk LINK access checking (required for VM/SP Release 4).

FILECHK=YES provides CMS file level access checking even if it is disabled in CA-ACF2.

LIDREC=YES uses the CA-ACF2 LIDREC field called LIDNDM to determine functional authority.

LIDFLD=LIDNDM uses the CA-ACF2 LIDREC field called LIDNDM to determine functional authority.

ADMVAL=C'A' is the CONNECT:Direct administrator authority value.

OPRVAL=C'O' is the CONNECT:Direct operator authority value.

GENVAL=C'G' is the CONNECT:Direct general user authority value.

TEST=YES provides a security exit trace since this is a first time installation.

Step 2—Build REXX EXEC

Build the REXX EXEC, as shown on page 3-27, on your A-disk called CDASM EXEC A. The sample REXX EXEC can be used to assemble and link-edit the Stage 1 Signon exit and Stage 2 Security exit in the CA-ACF2 environment.

Step 3—Assemble and Link-edit Stage 1 Signon

Enter the following commands to assemble and link-edit the Stage 1 Signon exit:

```
CDASM DMCXSIGN
```

Note: If you have comment statements in your link-edit input string, you will receive message IEW0222 for each comment statement. These can be ignored.

Step 4—Assemble and Link-edit Stage 2 Security Exit

Enter this command to assemble and link-edit the Stage 2 Security exit:

```
CDASM DMGACF2
```

Note: CONNECT:Direct sample exits require Assembler H or High-level Assembler for correct assembly.

Step 5—Verify CA-ACF2 Parameters

Verify that the correct CA-ACF2 parameters are active for the userid used for the CONNECT:Direct DTF GCS machine. The CA-ACF2 definitions must do the following:

- The LIDREC for CONNECT:Direct must have the following CA-ACF2 privileges:
  - SRF
  - ACCOUNT
  - SECURITY
- The ACFFDR must have an @SRF macro defined for the CONNECT:Direct LID. For example:
  - @SRF TESTCD,MODE=LOG
  - The SRVMOPTS macro must have the CONNECT:Direct LID defined as a batch service machine. For example:
  - BATCH=(CMSBATCH,TESTCD,...)

Step 6—Add Initialization Parameters

Add the following parameters to your CONNECT:Direct initialization parameter file:

```
SECURITY.EXIT=(DMGACF2,ALL)
SECURITY.TYPE=ACF2
```

Step 7—Add FILEDEF Statement

Add a FILEDEF statement to your PROFILE GCS file for the ddname SECURITY to receive trace information at the console.

For example:

```
FILEDEF  SECURITY  DUMMY
```

Step 8—Start CONNECT:Direct

IPL your GCS machine and start CONNECT:Direct. When CONNECT:Direct is initialized, sign on to CONNECT:Direct with a valid userid and password, and test CONNECT:Direct functional authority. This can be done by signing on with different levels of authorization IDs and performing tests.

Note: The security exit(s) must be made available to the CONNECT:Direct DTF GCS machine through the GLOBAL LOADLIB statement. In addition, if you are running with ACF2 Release 3.1, the ACFGCS LOADLIB must be made available in that same GLOBAL LOADLIB statement. However, if you are running with ACF2 Release 3.2 or higher, the ACFSRF LOADLIB must be made available in that same GLOBAL LOADLIB statement.

Exit Implementation in the RACF Environment

The following steps are required to implement the RACF exit.

Step 1—Edit DMGRACF ASSEMBLE

Edit DMGRACF ASSEMBLE with the correct options for your RACF environment. The following is a sample DMGRACF exit.

```
         DMGRACF DMGSECUR TYPE=RACF,                          X
                          STAGE1=YES,                         X
                          PNODEID=YES,                        X
                          SNODEID=YES,                        X
                          TEST=YES,                           X
                          AUTHFIL=YES,                        X
                          DEFLTID=DEFAULT
                      END
```

The DMGRACF exit performs the following:

STAGE1=YES supports the Stage 1 Signon exit.

PNODEID=YES supports the PNODEID overrides.

SNODEID=YES supports the SNODEID overrides.

TEST=YES provides a security exit trace since this is a first time installation.

AUTHFIL uses the CONNECT:Direct authorization file to determine CONNECT:Direct authority.

DEFLTID sets any userid not found in the authorization file defaults to general user authority.

Step 2—Build REXX EXEC

Build the REXX EXEC, as shown as page 3-27, on your A-disk called CDASM EXEC A. The sample REXX EXEC can be used to assemble and link-edit the Stage 1 Signon exit and Stage 2 Security exit in the RACF environment.

Step 3—Assemble and Link-edit Stage 1 Signon Exit

Enter the following commands to assemble and link-edit the Stage 1 Signon exit:

```
CDASM DMCXSIGN
```

Note: If you have comment statements in your link-edit input string, you will receive message IEW0222 for each comment statement. These can be ignored.

Step 4—Assemble and Link-edit Stage 2 Security Exit

Enter the following commands to assemble and link-edit the Stage 2 Security exit:

```
CDASM DMGRACF
```

Note: CONNECT:Direct sample exits require Assembler H or High-level Assembler for correct assembly.

Step 5—Define the DTF Userid to RACF

The CONNECT:Direct DTF userid must be defined to RACF as a surrogate controller.

To provide this capability, enter the following RACF commands:

- Ensure that the class of VMBATCH is active by issuing the following RACF command:

```
SETROPTS CLASSACT(VMBATCH)
```

- Define the CONNECT:Direct DTF userid as a member of the VMBATCH class. This is done by issuing the following RACF command, where userid is the CONNECT:Direct DTF userid:

```
RDEFINE VMBATCH (userid)
```

- Define all users or groups that will use CONNECT:Direct as alternate IDs for the CONNECT:Direct DTF. This is done by the following RACF Permit command, where userid is the CONNECT:Direct DTF userid:

```
PERMIT altid CLASS(VMBATCH) ID(userid)
ACCESS(CONTROL)
```

Note: Alternate IDs must be defined for each userid that will be using the CONNECT:Direct DTF. Refer to the IBM *RACF System Administrator's Guide* for more information on specifying alternate IDs.

Step 6—Add Initialization Parameters

Add the following parameters to your CONNECT:Direct initialization parameter file:

```
SECURITY.EXIT=(DMGRACF,ALL)
SECURITY.TYPE=RACF
```

Step 7—Add FILEDEF Statement

Add a FILEDEF statement to your PROFILE GCS file for the DDNAME SECURITY to receive trace information at the console. For example:

```
FILEDEF SECURITY DUMMY
```

Step 8—Start CONNECT:Direct

IPL your GCS machine and start CONNECT:Direct. When CONNECT:Direct is initialized, sign on to CONNECT:Direct with a valid userid and password, and test CONNECT:Direct functional authority. This can be done by signing on with different levels of authorization ids and performing tests.

Note: The security exit(s) must be made available to the CONNECT:Direct DTF GCS machine through the GLOBAL LOADLIB statement.

Exit Implementation in the VMSECURE Environment

The following steps are required to implement the VMSECURE exit.

Note: The VMSECURE RULES facility must be installed to run the Stage 2 exit with VMSECURE.

Step 1—Edit DMGVMSE2 ASSEMBLE

Edit DMGVMSE2 ASSEMBLE with the correct options for your VMSECURE environment. The following is a sample DMGVMSE2 exit.

```
DMGVMSEC        DMGSECUR   TYPE=VMSECUR,                        X
                           LINKCHK=YES,                         X
                           VMSECID=vmsecid,                     X
                           GCSAPI=cmsid,                        X
                           ADMVAL=ccuu,                         X
                           OPRVAL=ccuu,                         X
                           GENVAL=ccuu,                         X
                           STAGE1=YES,                          X
                           PNODEID=YES,                         X
                           SNODEID=YES,                         X
                           TEST=YES,                            X
                           LIDREC=NO,                           X
                           AUTHFIL=NO
                         END
```

The DMGVMSE2 exit performs the following:

LINKCHK=YES sets functional authorization.

Note: If LINKCHK=NO is specified, the CONNECT:Direct AUTH file will be used to grant authority. If LINKCHK=YES is specified, then AUTHFIL must be coded as NO. The CONNECT:Direct DTF machine must be given the authority to issue the VMSECURE CAN command.

For example, add the GRANT command in the AUTORIZ CONFIG file in the VMSECURE service machine:

```
GRANT CAN TO *ALL
GRANT NOPASS CAN TO userid
```

In the previous example, userid is the CMS id for the CONNECT:Direct DTF machine.

VMSECID=vmsecure id specifies the CMS machine ID of the VMSECURE service machine.

GCSAPI=dtf id specifies the CMS ID of the CONNECT:Direct DTF machine.

ADMVAL=ccuu specifies the virtual minidisk address for ADMIN.

OPRVAL=ccuu specifies the virtual minidisk address for OPERATOR.

GENVAL=ccuu specifies the virtual minidisk address for GENERAL USER. A value of DEFAULT causes GENUSR authority to default if the first two checks fail.

STAGE1=YES supports the Stage 1 Signon exit.

PNODEID=YES supports the PNODEID overrides.

SNODEID=YES supports the SNODEID overrides.

TEST=YES provides a security exit trace since this is a first time installation.

LIDREC=NO specifies a user-defined procedure to determine CONNECT:Direct functional authority.

AUTHFIL=NO specifies a user-defined procedure to determine CONNECT:Direct functional authority.

Step 2—Build REXX EXEC

Build the REXX EXEC, as shown on page 3-27, on your A-disk called CDASM EXEC A. The sample REXX EXEC can be used to assemble and link-edit the Stage 1 Signon exit and Stage 2 Security exit in the VMSECURE environment.

Step 3—Assemble and Link-edit Stage 1 Signon

Enter the following commands to assemble and link-edit the Stage 1 Signon exit:

```
CDASM DMCXSIGN
```

Note: If you have comment statements in your link-edit input string, you will receive message IEW0222 for each comment statement. These can be ignored.

Step 4—Assemble and Link-edit Stage 2 Security Exit

Enter the following commands to assemble and link-edit the Stage 2 Security exit:

```
CDASM DMGVMSE2
```

Note: CONNECT:Direct sample exits require either Assembler H or High-level Assembler for correct assembly.

Step 5—Define DTF Userid to VMSECURE

The CONNECT:Direct DTF userid must be defined to VMSECURE as a surrogate controller. To provide this capability, add the following GRANT records to your VMSECURE configuration file where userid is the CONNECT:Direct DTF userid:

```
GRANT SURROGAT TO userid
GRANT DIAGPCHK TO userid
```

Step 6—Add Initialization Parameters

Add the following parameters to your CONNECT:Direct initialization parameter file:

```
SECURITY.EXIT=(DMGVMSE2,ALL)
SECURITY.TYPE=VMSECURE
```

Step 7—Add FILEDEF Statement

Add a FILEDEF statement to your PROFILE GCS file for the DDNAME SECURITY to receive trace information at the console. For example:

```
FILEDEF  SECURITY  DUMMY
```

Step 8—Start CONNECT:Direct

IPL your GCS machine and start CONNECT:Direct. When CONNECT:Direct is initialized, sign on to CONNECT:Direct with a valid userid and password, and test CONNECT:Direct functional authority. This can be done by signing on with different levels of authorization IDs and performing tests.

Note: The security exit(s) must be made available to the CONNECT:Direct DTF GCS machine via the GLOBAL LOADLIB statement.

Exit Implementation in the VMSECURE Environment – Alternative Method

The following steps provide you with an alternative method to implement the VMSECURE exit.

Step 1–Edit DMGVMSEC

Edit DMGVMSEC ASSEMBLE with the correct options for your VMSECURE environment. This sample illustrates the exit.

```
DMGVMSEC DMGSECUR TYPE=VMSECUR,
                  STAGE1=YES,
                  PNODEID=YES,
                  SNODEID=YES,
                  TEST=NO,
                  AUTHFIL=YES,
                  DEFLTID=DEFAULT
```

This sample exit shows a user configured the CONNECT:Direct AUTH file to assign CONNECT:Direct functional authority within the CONNECT:Direct API.

Step 2—Build REXX EXEC

Build the REXX EXEC, as shown on page 3-27, on your A-disk called CDASM EXEC A. The sample REXX EXEC can be used to assemble and link-edit the Stage 1 Signon exit and Stage 2 Security exit in the VMSECURE environment.

Step 3—Assemble and Link-edit Stage 1 Signon

Enter the following commands to assemble and link-edit the Stage 1 Signon exit:

```
CDASM DMCXSIGN
```

Note: If you have comment statements in your link-edit input string, you will receive message IEW0222 for each comment statement. These may be ignored.

Step 4—Assemble and Link-edit Stage 2 Security Exit

Enter the following commands to assemble and link-edit the Stage 2 Security exit:

```
CDASM DMGVMSEC
```

Note: CONNECT:Direct sample exits require either Assembler H or High-level Assembler for correct assembly.

Step 5—Define DTF Userid to VMSECURE

The CONNECT:Direct DTF userid must be defined to VMSECURE as a surrogate controller. To provide this capability, add the following GRANT records to your VMSECURE configuration file where userid is the CONNECT:Direct DTF userid:

```
GRANT SURROGAT TO userid
GRANT DIAGPCHK TO userid
```

Step 6—Add Initialization Parameters

Add the following parameters to your CONNECT:Direct initialization parameter file:

```
SECURITY.EXIT=(DMGVMSEC,ALL)
SECURITY.TYPE=VMSECURE
```

Step 7—Add FILEDEF Statement

Add a FILEDEF statement to your PROFILE GCS file for the DDNAME SECURITY to receive trace information at the console. For example:

```
FILEDEF  SECURITY  DUMMY
```

Step 8—Start CONNECT:Direct

IPL your GCS machine and start CONNECT:Direct. When CONNECT:Direct is initialized, sign on to CONNECT:Direct with a valid userid and password, and test CONNECT:Direct functional authority. This can be done by signing on with different levels of authorization IDs and performing tests.

Note: The security exit(s) must be made available to the CONNECT:Direct DTF GCS machine using the GLOBAL LOADLIB statement.

Chapter 4

Maintaining User Authorization

This chapter includes the following information:

- Description of the Authorization Facility
- Maintaining the User Authorization File

Read the *Controlling Security* chapter beginning on page 3-1 for more information about the Authorization Facility.

Authorization Facility

The CONNECT:Direct Authorization Facility controls access to CONNECT:Direct functions and serves as the source of security information as an alternative to the Stage 1 Signon exit and Stage 2 Security exit described in the *Controlling Security* chapter beginning on page 3-1. If the CONNECT:Direct Authorization Facility is used, all CONNECT:Direct users must be identified in all nodes that will participate in Process execution.

Understanding the Authorization File

The Authorization file contains records of user attribute defaults. Each of these records defines which features of CONNECT:Direct the user is allowed to access for each node.

Accessing the Authorization File

Individual users can access the User Authorization screen to display information about their own authorization record. The following procedure describes how to access the User Authorization File:

- Select option **AUTH** from the Primary OPTIONS Menu

Refer to the *CONNECT:Direct for VM/ESA User's Guide* for a display of the screen and a description of each field.

Maintaining the User Authorization File

The User Authorization file maintenance commands can be entered through:

- Batch Interface
- ISPF Interactive User Interface
- Operator Interface

Refer to the *CONNECT:Direct Console Operator's Guide* for information on using the Operator Interface.

There are four User Authorization file maintenance commands:

- INSERT USER
- UPDATE USER
- DELETE USER
- SELECT USER

INSERT USER and UPDATE USER Commands

The INSERT USER and UPDATE USER commands enable you to add or update a user in the CONNECT:Direct Authorization file. The CONNECT:Direct functions and resources available to the user are defined through this command.

The commands have the following format and associated parameters. The required parameters and keywords appear in bold print. Default values for parameters and subparameters are underlined.

Note: The NAME parameter is required only for INSERT USER.

Label	Command	Parameters
(optional)	**INSert USER** **UPDate USER**	**USERID** = (nodeid, userid)
		NAME ='username'
		ADD TYPE = Y | N
		ALTER TYPE = Y | N
		READ TYPE = Y | N
		REMOVE TYPE = Y | N
		ADD USER = Y | N
		ALTER USER = Y | N
		READ USER = Y | N
		REMOVE USER = Y | N
		CHANGE = Y | N
		COPY = Y | N
		DELPR = Y | N
		EVENTCMD = Y | N
		FLUSH = Y | N
		† GEN.CHG.PROCESS = Y | N
		† GEN.DEL.PROCESS = Y | N
		† GEN.FLS.PROCESS = Y | N
		† GEN.SEL.PROCESS = Y | N
		† GEN.SEL.STATISTICS = Y | N
		MAXSA = max signon attempts
		MODALS = Y | N
		MODIFY = Y | N
		PASSword = initial password
		PHone = 'phone number'
		PTICDATA = (APPL profile name, secured signon key)
		RESETSA
		RUNJOB = Y | N
		RUNTASK = Y | N
		SECURITY = (security id, security pswd)
		SELNET = Y | N
		SELPR = Y | N
		SELSTAT = Y | N
		STAT COMMAND = Y | N

(continued)

Label	Command	Parameters	
		STOPCD	= Y | N
		SUBMIT	= Y | N
		†† SUBMITTER.CMDS	= (Y | N , Y | N , Y | N , Y | N , Y | N , Y | N , Y | N , Y | N)
		UPDNET	= Y | N
		CASE	= Y | N

† Valid only in the Interactive User Interface.
†† Valid only in the Batch Interface.

Required Parameters

The following are the required parameters for the INSERT USER command. The USERID parameter is a required parameter for the UPDATE USER command, but the the NAME parameter is not.

USERID = (nodeid, userid)
specifies the user node and userid of the record being added or updated. The user node is a 1–72 character alphanumeric string. The userid can contain 1–64 characters of any kind.

nodeid specifies the user node of the User record.

userid specifies the userid of the User record.

NAME = 'username'
specifies the full name of the user. The NAME is a string of 1–20 characters. If blanks are embedded in the NAME parameter, the NAME must be enclosed in single quotation marks. This parameter is not required by the UPDATE USER command.

Optional Parameters

The following optional parameters for the INSERT USER and UPDATE USER commands are separated into two categories: authorization record parameters and functional authorization parameters.

Authorization Record Parameters

The following are the authorization record parameters for the INSERT USER and UPDATE USER commands. Each user can be authorized to add, alter, read, or remove a record. The authorization is specified by

indicating the action (ADD, ALTER, READ, REMOVE) followed by the record type. *If no action is specified for a Type or User record, the default value is No.*

ADD TYPE = Y | N
specifies whether the user is allowed to insert new records into the Type Defaults file. The default value is No.

ALTER TYPE = Y | N
specifies whether the user is allowed to update records in the Type Defaults file. The default value is No.

READ TYPE = Y | N
specifies whether the user is allowed to read records from the Type Defaults file. The default value is No.

REMOVE TYPE= Y | N
specifies whether the user is allowed to delete records from the Type Defaults file. The default value is No.

ADD USER = Y | N
specifies whether the user is allowed to insert new records into the Authorization file. The default value is No.

ALTER USER = Y | N
specifies whether the user is allowed to update records in the Authorization file. The default value is No.

READ USER = Y | N
specifies whether the user is allowed to read records from the Authorization file. The default value is No.

REMOVE USER = Y | N
specifies whether the user is allowed to delete records from the Authorization file. The default value is No.

Functional Authorization Parameters

The following are the functional authorization parameters for the INSERT USER and UPDATE USER commands.

CHANGE = Y | N
specifies whether the user is allowed to use the CHANGE PROCESS command. The default value is No.

COPY = Y | N
specifies whether the user is allowed to use the COPY statement. The default value is No.

DELPR = Y | N

specifies whether the user is allowed to use the DELETE PROCESS command. The default value is No.

EVENTCMD = Y | N

specifies whether the user is allowed to use the EVENT SERVICES SUPPORT commands. No is the default.

FLUSH = Y | N

specifies whether the user is allowed to use the FLUSH PROCESS and SUSPEND PROCESS commands. The default value is No.

GEN.CHG.PROCESS = Y | N

(valid only in the IUI) determines whether the user can change any Processes or only Processes that are submitted. If you specify GEN.CHG.PROCESS=Y, the user can only change Processes that the user submitted. The default value is No.

GEN.DEL.PROCESS = Y | N

(valid only in the IUI) determines whether the user can delete any Processes or only Processes that are submitted. If you specify GEN.DEL.PROCESS=Y, the user can only delete Processes that the user submitted. The default value is No.

GEN.FLS.PROCESS = Y | N

(valid only in the IUI) determines whether the user can flush any Processes or only Processes that the user submitted. If you specify GEN.FLS.PROCESS=Y, the user can only flush Processes that the user submitted. The default value is No.

GEN.SEL.PROCESS = Y | N

(valid only in the IUI) determines whether the user can select any Processes or only Processes that the user submitted. If you specify GEN.SEL.PROCESS=Y, the user can only select Processes that the user submitted. The default value is No.

GEN.SEL.STATISTICS = Y | N

(valid only in the IUI) determines whether the user can select any statistics or only statistics for Processes that the user submitted. If you specify GEN.SEL.STATISTICS=Y, the user can only select statistics for Processes that the user submitted. The default value is No.

MAXSA = max signon attempts

(valid only in the IUI) specifies the maximum number of sign-on attempts the user is allowed per hour. The range is 0–99. The default is 60. There is no maximum number if **0** is specified. (See the RESETSA parameter on page 4-7 for how to temporarily reset this value.)

MODALS = Y | N

specifies whether the user is allowed to use the modal statements IF, ELSE, EIF, GOTO, and EXIT. The default value is No.

MODIFY = Y | N

specifies whether the user is allowed to request traces and modify certain CONNECT:Direct operational functions. The default value is No.

PASSword = initial password

defines the initial password for the userid. The password is a 4–64 character alphanumeric string.

Note: Tandem users cannot use the escape or control keys because CONNECT:Direct does not recognize them.

PHONE = 'phone number'

specifies the phone number of the user. The phone number must be enclosed in single quotation marks. The quotation marks allow for a space after the area code.

PTICDATA=(APPL prof name, secured signon key)

specifies the values required for the Stage 2 security exit to rewrite a RACF PassTicket password. APPL prof name is the value specified when the profile is defined for the PTICDATA class. The secured signon key is the value associated with the PTICDATA class and the name specified in the APPL Prof name.

RESETSA

specifies that the signon attempt count is to be reset to 0. (See the MAXSA parameter on page 4-7 for how to set the signon attempt count.) This allows the user to try to sign on, even if he or she has previously exceeded the maximum number of signon attempts. *This parameter is used in the UPDATE USER command only.*

RUNJOB = Y | N

specifies whether the user is allowed to use the RUN JOB statement. The default value is No.

RUNTASK = Y | N
specifies whether the user is allowed to use the RUN TASK statement. The default value is No.

SECURITY = (security ID, security pswd)
specifies the security ID and security password used to identify the file authorization of the user. Security support includes CA-ACF2, VMSECURE, and RACF.

Security ID specifies the 1–64 character security system ID for the user. This ID must meet the standards of the security subsystem at the location of the user. The security ID is required if this parameter is specified.

Security pswd specifies the 1–64 character security system password for the user. This password must meet the standards of the security subsystem at the location of the user.

SELNET = Y | N
specifies whether the user is allowed to use the SELECT NETMAP command. The default value is No.

SELPR = Y | N
specifies whether the user is allowed to use the SELECT PROCESS command. The default value is No.

SELSTAT = Y | N
specifies whether the user is allowed to use the SELECT STATISTICS command. The default value is No.

STAT COMMAND = Y | N
specifies whether the user is allowed to use the STATISTICS COMMAND command. The default value is No.

STOP CD = Y | N
specifies whether the user is allowed to use the STOP CD command. The default value is No.

SUBMIT = Y | N
specifies whether the user is allowed to use the SUBMIT statement within a Process. The default value is No.

SUBMITTER.CMDS = (Y | N Y | N Y | N Y | N Y | N Y | N)
(valid only in the batch interface) specifies whether the user is allowed to issue certain commands concerning the Processes that the user submitted.

These commands in order are:

- SELECT PROCESS
- DELETE PROCESS
- FLUSH PROCESS
- SUSPEND PROCESS
- CHANGE PROCESS
- SELECT STATISTICS

For more information on each of these commands, refer to the IUI definitions for these commands beginning on page 4-6. The IUI definitions begin with GEN and the command name is abbreviated.

The default value for each is No.

UPDNET = Y | N
specifies whether the user is allowed to use the UPDATE NETMAP command. The default value is No.

CASE = Y | N
specifies whether parameters associated with accounting data, userid, password, and data set name are to be case sensitive. This choice overrides the case sensitivity designation selected for the session at signon and is in effect only for this command. The default is the designation made at session signon.

Batch Interface Use of INSERT USER or UPDATE USER

To use the INSERT USER or UPDATE USER commands from the Batch Interface, perform the following steps:

1. Place the INSERT USER or UPDATE USER commands in the DMBATCH job stream as described in the *CONNECT:Direct for VM/ESA User's Guide.*

2. Make sure that CONNECT:Direct is running.

3. Submit the job.

4. Verify your changes.

Batch Interface Example of INSERT USER

The following example shows how to add a User record for userid SMITH to the Authorization file.

```
SIGNON  USERID=(userid, password)
INSERT USER    USERID=(DALLAS, SMITH) -
               NAME='RB SMITH'  PASS=XYZZY -
               PH='214 555-5555'  -

               ADD USER=Y  ALTER USER=Y -
               READ USER=Y  REMOVE USER=Y -
               SUBMIT=Y -
               SUBMITTER.CMDS=(Y Y Y Y N)
SIGNOFF
```

With the preceding definition, Smith can perform the following functions:

- Add users to the CONNECT:Direct Authorization file
- Update and read User records
- Delete User records
- Define and submit Processes for execution
- Select, delete, flush/suspend, and change Processes that have been submitted

Smith cannot perform the Select Statistics command on any Processes, however, regardless of who submitted them. Smith's initial password is XYZZY, and Smith's phone number is (214) 555–5555.

Batch Interface Example of UPDATE USER

The following commands update the record of a user named Smith in the Authorization file.

```
UPDATE USER    USERID=(DALLAS, SMITH) -
               NAME='RB SMITH' -
               PASS=XYZZY -
               PH='214 555-5555' -
               ADD USER=Y ALTER USER=Y -
               READ USER=Y  REMOVE USER=Y -
               CH=Y  FLUSH=Y  DELPR=Y
```

With the previous updates, Smith is able to perform the following functions:

- Add Users to the CONNECT:Direct Authorization file
- Update and read User records
- Delete User records
- Change a process in the TCQ

- Delete an executing process from the TCQ
- Delete an inactive process from the TCQ

INSERT USER and UPDATE USER Through the IUI

To issue INSERT USER and UPDATE through the CONNECT:Direct IUI, perform the following steps:

1. Access the Insert/Update/Select/Delete User Record screen by selecting option **IU** from the Administrative Options Menu.

```
node.name          INSERT/UPDATE/SELECT/DELETE USER RECORD          hh:mm
CMD ==>

FUNCTION    ==> SEL ('I'=INS, 'U'=UPD, 'S'=SEL , 'D'=DEL)
ENTER USER INFORMATION:                  NAME      ==> ________________
  USER ID ==> ____________________________________________________________
  USERNODE==> ________________             PHONE     ==> ____________
  PASSWORD==>
  SEC ID  ==> ____________________________________________________________
  SEC PASS==>
  MAX SIGNON ATTEMPTS===> ___       PASSTICKET DATA ==> ( ________ ,         )
DEFINE USER FUNCTIONS: ('Y'=YES OR 'N'=NO, EXTEND WITH BLANKS IF NECESSARY)
  INSERT USER     => _     FLUSH PROCESS       => _   SELECT NETMAP     => _
  DELETE USER     => _     SELECT PROCESS      => _   UPDATE NETMAP     => _
  SELECT USER     => _     SUBMIT FUNCTION     => _   MODALS FUNCTION   => _
  UPDATE USER     => _     RUNJOB FUNCTION     => _   RUNTASK FUNCTION  => _
  COPY FUNCTION   => _     CONTROL TRACING     => _   INSERT TYPE       => _
  CHANGE PROCESS  => _     STOP CONNECT:Direct => _   DELETE TYPE       => _
  DELETE PROCESS  => _     SELECT STATISTICS   => _   SELECT TYPE       => _
  SQL SELECT      => _     SQL INSERT          => _   UPDATE TYPE       => _
  SQL CREATE      => _     SQL DROP            => _   SQL GRANT         => _
  STAT COMMAND    => _     GEN.DEL.PROCESS     => _   GEN.FLS.PROCESS   => _
  GEN.SEL.PROCESS => _     GEN.SEL.STATISTICS  => _   RESET SIGNON      => N
  GEN.CHG.PROCESS => _     EVENT COMMAND       => _   MIXED CASE?        => N
```

2. Enter the information in the appropriate fields.

Note: See the description of the INSERT and UPDATE USER command parameters beginning on page 4-4 for the valid values of the fields, or press the **PF1** key for online help.

---

3. Press **ENTER** to execute the command after you have completed your entries.

DELETE USER Command

The DELETE USER command removes a User record from the CONNECT:Direct Authorization file. The command has the following

format and associated parameters. The required parameters and keywords appear in bold print.

Label	Command	Parameters
(optional)	**DELete USER**	**WHERE (**
		USERID = (nodeid, userid) | (list)
		)
		CASE = YES | NO

Required Parameter

WHERE is the only required parameter.

WHERE (USERID = (nodeid, userid) | (list))
specifies which User record(s) to delete.

USERID = (nodeid, userid) | (list)
specifies the record to be deleted from the CONNECT:Direct Authorization file.

nodeid specifies the node ID of the User record that will be searched. The nodeid is a 1–16 character alphanumeric string.

userid specifies the userid of the User record. The complete userid consists of the nodeid and the userid enclosed in parentheses and separated by a comma.

list specifies a list of userids.

Optional Parameter

CASE is the only optional parameter.

CASE = YES | NO
specifies whether parameters associated with nodeid and userid are to be case sensitive.

This choice overrides the case sensitivity designation selected for the session at signon and is in effect only for this command. The default is the designation made at session signon.

Batch Interface Use of DELETE USER

To use the DELETE USER command from the Batch Interface, perform the following steps:

1. Place the DELETE USER commands in the DMBATCH job stream as described in the *CONNECT:Direct for VM/ESA User's Guide.*

2. Ensure that CONNECT:Direct is running.

3. Submit the job.

4. Verify your changes.

The following example shows how to delete single and multiple User records.

```
* DELETES A SINGLE User record
        DELETE USER WHERE (USERID=(MPLS, SMITH))

* DELETES MULTIPLE User records
        DELETE USER WHERE (USERID=(DALLAS, JONES), -
                (MPLS, SMITH), (CHICAGO, BROWN)))
```

Deleting Users Through the IUI

This section provides instructions on how to delete a user using the following IUI screens:

- Delete A User Record screen
- Insert/ Update/ Select/ Delete User Record screen

Using the Delete A User Record Screen

The Delete A User Record screen allows you to simultaneously delete up to four User records. To use the Delete a User Record screen, perform the following steps:

1. Access the Delete A User Record screen by selecting option **DU** from the Administrative Options Menu.

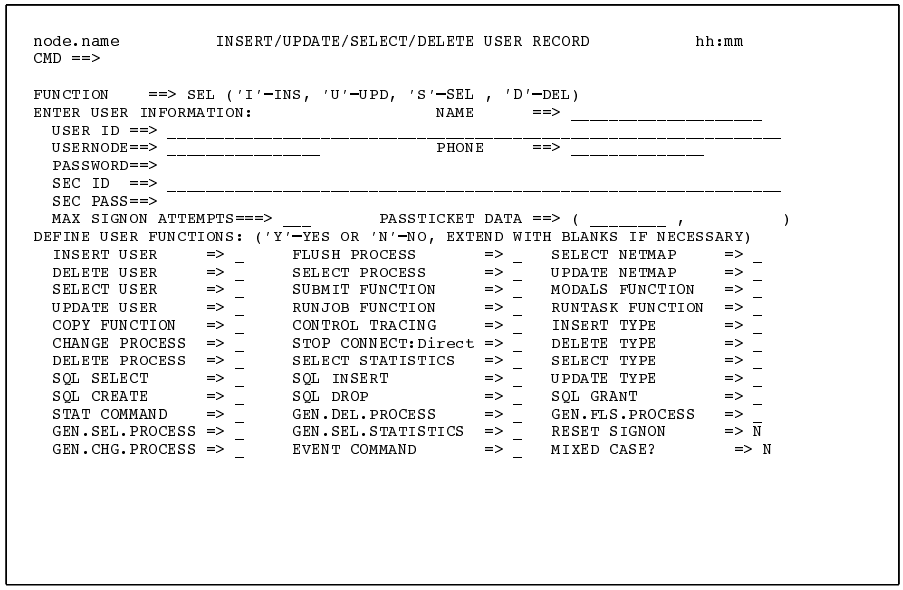
2. Enter the userid and user node of the records to be deleted and press **ENTER**.

3. Verify your results.

Using the Insert/Update/Select/Delete User Record Screen to Delete a User

To delete a user from the Insert/Update/Select/Delete User Record screen, perform the following steps:

1. Access the Insert/Update/Select/Delete User Record screen by selecting option **IU** from the Administrative Options Menu.



```
node.name          INSERT/UPDATE/SELECT/DELETE USER RECORD          hh:mm
CMD ==>

FUNCTION    ==> SEL ('I'=INS, 'U'=UPD, 'S'=SEL , 'D'=DEL)
ENTER USER INFORMATION:                  NAME      ==> ________________
  USER ID ==> ____________________________________________________________
  USERNODE==> _____________              PHONE     ==> ____________
  PASSWORD==>
  SEC ID  ==> ____________________________________________________________
  SEC PASS==>
  MAX SIGNON ATTEMPTS===> ___        PASSTICKET DATA ==> ( ________ ,          )
DEFINE USER FUNCTIONS: ('Y'=YES OR 'N'=NO, EXTEND WITH BLANKS IF NECESSARY)
  INSERT USER     => _     FLUSH PROCESS       => _   SELECT NETMAP     => _
  DELETE USER     => _     SELECT PROCESS      => _   UPDATE NETMAP     => _
  SELECT USER     => _     SUBMIT FUNCTION     => _   MODALS FUNCTION   => _
  UPDATE USER     => _     RUNJOB FUNCTION     => _   RUNTASK FUNCTION  => _
  COPY FUNCTION   => _     CONTROL TRACING     => _   INSERT TYPE       => _
  CHANGE PROCESS  => _     STOP CONNECT:Direct => _   DELETE TYPE       => _
  DELETE PROCESS  => _     SELECT STATISTICS   => _   SELECT TYPE       => _
  SQL SELECT      => _     SQL INSERT          => _   UPDATE TYPE       => _
  SQL CREATE      => _     SQL DROP            => _   SQL GRANT         => _
  STAT COMMAND    => _     GEN.DEL.PROCESS     => _   GEN.FLS.PROCESS   => _
  GEN.SEL.PROCESS => _     GEN.SEL.STATISTICS  => _   RESET SIGNON      => N
  GEN.CHG.PROCESS => _     EVENT COMMAND       => _   MIXED CASE?        => N
```

2. Enter the information in the appropriate fields.

Note: Refer to the description of the DELETE USER command parameters beginning on page 4-12. Press the **PF1** key for online help descriptions of each field.

---

3. Press **ENTER** to execute the command after you have completed your entries.

SELECT USER Command

The SELECT USER command displays a User record in the Authorization file. You can specify the search criteria and the form in which the information is presented.

The command has the following format and associated parameters. Required parameters and keywords appear in bold print. Default values are underlined.

Label	Command	Parameters
(optional)	**SELect USER**	**WHERE (**
		USERID = (nodeid, userid) | (generic | (list))
		EXCLUDE = (AUTH)
		)
		PRint | TABle
		CASE = YES | NO

Required Parameter

WHERE is the only required parameter for the SELECT USER command.

WHERE (USERID = (nodeid, userid) | (generic | list))
EXCLUDE = (AUTH)
specifies which User record(s) you want to examine.

USERID = (nodeid, userid) | (generic | list)
specifies the record to be searched for in the CONNECT:Direct Authorization file. This subparameter of the WHERE parameter is required. The complete userid consists of the nodeid and the userid enclosed in parentheses and separated by a comma.

nodeid specifies the node ID of the User record that will be searched. Enter a 1–16 character alphanumeric string. If the user node is not specified, nodeid defaults to the CONNECT:Direct system that receives the command.

userid specifies the userid of the User record.

generic specifies generic selection of userids. To specify user nodes and userids generically, type a 1–7 character alphanumeric string with the first character alphabetic, plus an asterisk (*). For instance, if you specify a userid of B*, you might examine records for BLACK, BRADFORD, and BROWN.

list specifies a list of userids.

EXCLUDE = (AUTH)
specifies that the function-by-function authorization description is not included in the output. This subparameter of the WHERE parameter is not required.

Optional Parameters

The following are the optional parameters for the SELECT USER command.

PRint | TABle
specifies the output destination.

PRINT specifies that the output of the SELECT USER command to a printer rather than a display. Printed output is in tabular format, the same as that produced by the TABle parameter. Output is routed to the destination specified in the PRINT keyword of the CONNECT:Direct SIGNON command.

TABLE specifies that the output of the SELECT USER command is stored in a temporary file in tabular format and is displayed upon successful completion of the command. The default for the output is TABLE.

CASE = Yes | No
specifies whether parameters associated with nodeid and userid are to be case sensitive. This choice overrides the case sensitivity designation selected for the session at signon and is in effect only for this command. The default is the designation made at session signon.

Batch Interface Use of the SELECT USER Command

To use this command from the Batch Interface, perform these steps:

1. Place your SELECT USER commands in the DMBATCH job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide.*

2. Ensure that CONNECT:Direct is running.

3. Submit the job.

4. Verify the results.

Batch Interface Example of Select User

This example illustrates the use of SELECT USER in the batch interface.

```
SELECT USER WHERE (USERID=(, BILL))
```

In this example, the command searches for user BILL at the local (default) node.

Selecting a User Through the IUI

This section provides instructions on how to select a user using the following IUI screens:

- Select A User Record screen
- Insert/ Update/ Select/ Delete User Record screen

Using the Select a User Record Screen

The Select a User Record screen allows you to simultaneously display user information and authorized functions for up to four users. To select a user using the Select A User screen, perform the following steps:

1. Access the Select a User Record screen by selecting option **SU** from the Administrative Options Menu.
2. Enter the userid and user node for the records to be displayed.
3. Enter the appropriate fields.
4. Press **ENTER** to execute the SELECT USER command after you have completed your entries.

See the description of the SELECT USER command parameters on page 4-15 for a description of the valid values for the fields or press **PF1** for online help.

To select a user through the Insert/Update/ Select/Delete User Record screen, perform the following steps:

1. Access the Insert/Update/Select/Delete User Record screen by selecting option **IU** from the Administrative Options Menu.

```
node.name          INSERT/UPDATE/SELECT/DELETE USER RECORD          hh;mm CMD
CMD ==>

FUNCTION    ==> SEL ('I'-INS, 'U'-UPD, 'S'-SEL , 'D'-DEL)
ENTER USER INFORMATION:                  NAME      ==> ________________
  USER ID ==> ____________________________________________________________
  USERNODE==> _____________              PHONE     ==> ____________
  PASSWORD==>
  SEC ID  ==> ____________________________________________________________
  SEC PASS==>
  MAX SIGNON ATTEMPTS===> ___        PASSTICKET DATA ==> ( _______ ,          )
DEFINE USER FUNCTIONS: (RESPOND WITH 'Y'-YES OR 'N'-NO)
  INSERT USER     => _     FLUSH PROCESS       => _   SELECT NETMAP     => _
  DELETE USER     => _     SELECT PROCESS      => _   UPDATE NETMAP     => _
  SELECT USER     => _     SUBMIT FUNCTION     => _   MODALS FUNCTION   => _
  UPDATE USER     => _     RUNJOB FUNCTION     => _   RUNTASK FUNCTION  => _
  COPY FUNCTION   => _     CONTROL TRACING     => _   INSERT TYPE       => _
  CHANGE PROCESS  => _     STOP CONNECT:Direct => _   DELETE TYPE       => _
  DELETE PROCESS  => _     SELECT STATISTICS   => _   SELECT TYPE       => _
  SQL SELECT      => _     SQL INSERT          => _   UPDATE TYPE       => _
  SQL CREATE      => _     SQL DROP            => _   SQL GRANT         => _
  STAT COMMAND    => _     GEN.DEL.PROCESS     => _   GEN.FLS.PROCESS   => _
  GEN.SEL.PROCESS => _     GEN.SEL.STATISTICS  => _   RESET SIGNON      => NO_
  GEN.CHG.PROCESS => _     EVENT COMMAND       => _   MIXED CASE? ==> NO
```

2. Enter the information in the appropriate fields.

Note: Refer to the description of the SELECT USER command parameters beginning on page 4-15 for the valid values of the fields, or press the **PF1** key for online help.

---

3. Press ENTER to execute the SELECT USER command after you have completed your entries.

Chapter 5

Maintaining the Type File

This chapter includes discussions of the purpose of the Type file and the format and parameters of the file maintenance commands.

Description of the Type File

The Type file contains records of file attribute information. CONNECT:Direct uses this information when you submit Processes with the TYPE parameter specified on the COPY statement and when CONNECT:Direct creates new files or accesses existing files.

```
COPY1 COPY  FROM  (DSN=MYFILE) -
            TO (DSN=YOURFILE TYPE=ALLOCATE)
```

The purpose of the Type file is twofold.

- It saves retyping parameters such as DCB, DISP, and SPACE within Processes for files with common attributes.
- It facilitates the use of previously defined attribute specifications of different systems (for example, MVS, VMS, PC, and Tandem). This is especially useful for remote CONNECT:Direct users who may not be familiar with VM data set organizations and allocation parameters.

Note: The Type record referenced must reside in the Type file on the **destination system**, which is the system responsible for allocating the new file.

Overriding File Attributes

If you specify file attributes in conjunction with the TYPE parameter on the COPY statement, the parameters coded on the COPY statement take precedence over like parameters on the associated Type record. This is particularly useful when you need to override a specific subparameter of a Type key record that is similar to what you want.

MS-DOS and OS/2 Type Keys

There are two predefined Type keys provided with CONNECT:Direct for VM/ESA: TEXT and BINARY. Use these for communication with CONNECT:Direct for MS-DOS and OS/2 nodes. The two Type keys contain file allocation information as defined below. These Type keys also determine whether to perform EBCDIC to ASCII translation initiated by CONNECT:Direct for VM/ESA.

Note: The TEXT Type Key must be installed on the VM node in order to receive data from an MS-DOS or OS/2 node.

TEXT Type Key

The following is the predefined TEXT Type key:

```
TYPE KEY => TEXT
    DCB=(DSORG=PS,LRECL=255,BLKSIZE=259,RECFM=V)
```

BINARY Type Key

The following is the predefined BINARY Type key:

```
TYPE KEY => BINARY
    DCB=(DSORG=PS,BLKSIZE=6144,RECFM=U)
```

DF Type Key

The following is the predefined DF Type key:

```
TYPE KEY => DF
    DCB=(DSORG=PS,LRECL=255,BLKSIZE=259,RECFM=V)
```

DF2 Type Key

The following is the predefined DF2 Type key:

```
TYPE KEY => DF2
    DCB=(DSORG=PS,LRECL=255,BLKSIZE=259,RECFM=V)
```

Maintaining the Type File

The Type file is maintained through the commands INSERT/UPDATE TYPE, DELETE TYPE, and SELECT TYPE. These commands can be entered through Interactive User Interface (IUI), Batch Interface, or Operator Interface. Refer to the *CONNECT:Direct for VM/ESA Operator Interface* chapter of the *CONNECT:Direct Console Operator's Guide* for information on using the Operator Interface.

INSERT and UPDATE TYPE Command

The INSERT TYPE command enables you to insert a record in the Type file. The UPDATE TYPE command enables you to update a record in the Type file. The commands use the following format and associated parameters. The required parameters appear in bold print. Default values for parameters are underlined.

Label	Command	Parameters
(optional)	**INSert TYPE** | **UPDate TYPE**	**TYPEKEY**=typekey
		LINK=(vmid,pwd,accmode,ccuu)
		VSAMCAT=(dsn,vmid,pwd,accmode,ccuu)
		DCB=(BLKSIZE = no. bytes ,DSORG = (DA | IS | PS | PO | VSAM) ,LRECL = no. bytes ,RECFM = record format)

Label	Command	Parameters
		DISP=((NEW | OLD | MOD | RPL | SHR) (,KEEP , CATLG , DELETE) (,KEEP , CATLG , DELETE))
		UNIT=unit type
		VOL=SER = volume serial number

Required Parameter

The following is the required parameter for the INSERT TYPE or UPDATE TYPE command.

TYPEKEY=typekey
specifies the name associated with the entry being added or updated. The Type key is a 1–8 character alphanumeric string, with the first character alphabetic.

Optional Parameters

The following are the optional parameters for the INSERT TYPE or UPDATE TYPE command.

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

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

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

mode specifies the link access mode.

When used with the FROM parameter, the access modes are:

- W (primary read/write access)
- M (primary multiple access)
- R (primary read only)
- RR (primary and secondary read only access)

- WR (primary read/write access; alternate read only access)
- MR (primary multiple access; alternate read only access)
- MW (primary multiple access; alternate read/write only access)

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

WARNING: MW access to CMS format disks can be destructive. You must be able to guarantee that no other VM user, or CONNECT:Direct Process, has MW, M, or W access to the minidisk. If multiple users or Processes get write access to the disk at the same time, there is a high probability that the CMS directory on the disk will be destroyed. The most likely result is the following message from Group Control System (GCS) or an equivalent message from CMS:

```
GCSFNS420T file system error
detected. Virtual address 'vaddr'.
Reason code 'nn'
```

When GCS issues the CSIFNS420T message, a GCS dump occurs and all processing terminates.

---

ccuu specifies the virtual address of the disk where the CMS file is located. Either a three-digit or four-digit hexadecimal value may be specified.

VSAMCAT=(dsn,userid,password,mode,ccuu)
specifies the catalog for the VSAM file to be copied. This parameter is required only if using a catalog other than the master catalog.

dsn specifies the filename of the VSAM catalog containing the file to be copied. The maximum length is 44 characters.

userid specifies the owner ID for the VSAM minidisk where the file is located. The maximum length is 8 characters.

password specifies the appropriate password for the VSAM minidisk where the file is located. The maximum length is 8 characters.

mode specifies the link access mode. Valid access modes are NULL, W (primary read/write access), M (primary multiple access), and MW (primary multiple access; alternate read/write only access).

ccuu specifies the virtual address of the disk where the VSAM file is located. Either a three-digit or four-digit hexadecimal value may be specified.

DCB=([BLKSIZE=no.bytes,DSORG=[PS|VSAM], LRECL=no.bytes, RECFM=record format])

specifies DCB information associated with the DSN on the COPY statement.

BLKSIZE specifies the length of the block in bytes.

DSORG specifies the file organization. File organizations supported are PS and VSAM.

LRECL specifies the length of the records in bytes .

RECFM specifies the format of the records in the file. Specify any valid format, such as F, FA, FB, FBA, FBM, FM, U, V, VB, VBA, VBM, and VBS.

The file attributes on the COPY statement take precedence over the Type file. If you do not code attributes on the COPY statement, the attributes in the Type file take precedence. If the attributes are coded neither on the COPY TO statement nor in the Type file, the attributes are taken from the FROM side (source).

DISP=({NEW|OLD|MOD|RPL|SHR} {,KEEP|,CATLG|,DELETE} {,KEEP|,CATLG|,DELETE})

specifies the default destination file status on the receiving node.

first subparameter of DISP specifies the status of the file. Only the OLD and RPL dispositions apply to VSAM files. There is no default.

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

OLD specifies that the destination file existed before the Process began executing

and that the Process is given exclusive control of the file. The destination file may be a VSAM file, a SAM file, or a PDS (CONNECT:Direct for MVS).

MOD specifies that the Process step modifies the SAM file by adding data to the end of the file. If a system failure occurs when MOD is specified, the system is designed not to restart because data loss or duplication would be difficult to detect.

RPL specifies that the destination file replaces any existing file or allocates a new file. DISP=RPL may be specified for SAM or VSAM files. If the file is VSAM, it must have been defined with the REUS attribute. RPL cannot be specified if VOL=SER is specified.

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.

second subparameter of DISP specifies the normal termination disposition. This subparameter does not apply to VSAM files. There is no default.

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 is placed in the catalog.

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

third subparameter of DISP specifies abnormal termination disposition. This subparameter does not apply to VSAM files. There is no default.

KEEP specifies that the system keeps the file after the Process step terminates abnormally.

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

DELETE specifies that the system deletes the file if CONNECT:Direct terminates abnormally.

UNIT = unit type
specifies the unit address, device type, or user-assigned group name that contains the data. For SAM-to-SAM copies where the destination file is new and the UNIT parameter is not coded with the TO parameter, the device type from the source file is used.

VOL=SER=volume serial number
specifies the volume serial number containing the file. If VOL=SER is not specified with the FROM parameter, the file must be cataloged.

Using the INSERT TYPE or UPDATE TYPE Command Through the Batch Interface

To use the INSERT TYPE or UPDATE TYPE command from the Batch Interface, perform the following steps:

1. Place the INSERT TYPE or UPDATE TYPE command in the DMBATCH job stream as described in the *CONNECT:Direct for VM/ESA User's Guide.*

2. Submit the job while CONNECT:Direct is running.

3. Verify your results.

Batch Interface INSERT TYPE Example

The following example adds a Type record named NEWALLOC to the Type file.

```
INSERT TYPE  TYPEKEY=NEWALLOC -
             DCB=(DSORG=PS) -
             DISP=(NEW)
```

CONNECT:Direct users can then reference the NEWALLOC Type record in a COPY request to allocate a new physical sequential file that will reside on a 3380 unit device and be cataloged upon normal termination.

Batch Interface UPDATE TYPE Example

The following example updates a record in the Type file. When referring to the Type record, NEWALLOC, the destination file will be an existing PS file to be replaced.

```
UPDATE TYPE   TYPEKEY=NEWALLOC -
              DISP=(RPL)
```

IUI Use of INSERT TYPE or UPDATE TYPE Command

The CONNECT:Direct IUI uses the same screen to insert or update a Type record. To issue the INSERT TYPE or UPDATE TYPE command from the CONNECT:Direct IUI, perform the following steps:

1. Access the Insert/Update Type Record screen by selecting option **IT** from the Administrative Options Menu. A series of screens appear, beginning with the Insert/Update Type Record screen.

```
node.name             INSERT/UPDATE TYPE RECORD                        hh:mm
CMD ==>

FUNCTION ==>  I ('I'-INS, 'U'-UPD, 'S'-SEL, 'D'-DEL)
TYPE KEY ==>

              INITIAL  NORMAL   ABNORMAL
DISP      ==> ___      _______

              DSORG  LRECL   BLKSIZE    RECFM
DCB       ==> ____   ______  __________

              ALLOCATION  PRIMARY   SECONDARY   DIRECTORY     RLSE     CONTIG  ROUND
SPACE     ==> __________  ________  __________  __________    _        _       _
UNIT      ==>
VOL=SER   ==>
              VMID         PASSWORD    MODE   CUU
LINK      ==> __________               __
              DSN
VSAMCAT   ==>
              VMID         PASSWORD    MODE   CUU
          ==> __________               __
```

2. Select either Insert or Update and press **ENTER**.

Note: CONNECT:Direct displays an error message if you attempt to insert an existing record or update a non-existent record.

The next display is the Type Record Selection List screen.

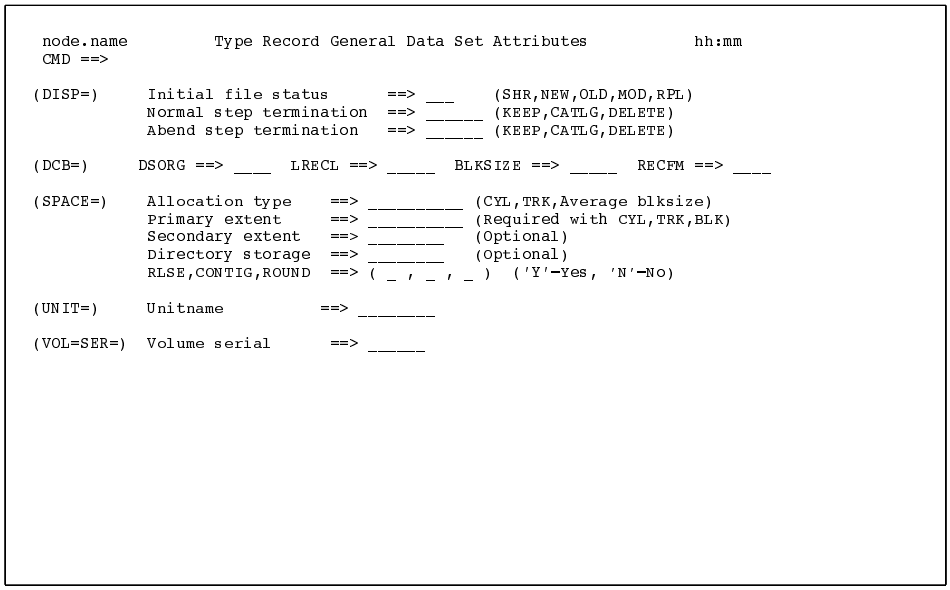
3. Select the General Data Set Attributes screen by entering an **S** next to the desired entry.

The following sections describe the purpose of each option. You can complete the insertion or update of the Type file record by entering the END command from the Type Record Selection List or selecting another option to add or update additional parameters.

To terminate the insert or update request, enter the CANcel command from the Type Record Selection List screen.

Type Record General Data Set Attributes Screen

From the Type Record General Data Set Attributes screen you can define or update the file attributes for the Type record. Refer to the INSERT TYPE and UPDATE TYPE command parameter descriptions beginning on page 5-4 for valid values.



After defining or updating the file attributes, enter the **END** command to return to the Type Record Selection List.

Using the DELETE TYPE Command

The DELETE TYPE command enables you to delete a record from the Type file. It uses the following format and associated parameters. The required parameters and keywords appear in bold print.

Label	Command	Parameters
(optional)	**DELete TYPE**	**WHERE** (
		TYPEKEY = typekey | (list)
		)

Required Parameters

WHERE is the required parameter for the DELETE TYPE command. There are no optional parameters.

WHERE (TYPEKEY=typekey | (list))
specifies which records(s) in the Type file to delete.

TYPEKEY=typekey | (list)
specifies the Type key or a list of Type keys.

typekey specifies the name associated with the record being deleted. The Type key is a 1–8 character alphanumeric string, with the first character alphabetic.

list specifies multiple Type keys. A list of Type keys can be specified by enclosing them in parentheses.

Optional Parameters

There are no optional parameters for the DELETE TYPE command.

Using the DELETE TYPE Command Through the Batch Interface

To use the DELETE TYPE command from the batch interface, perform the following steps:

1. Place your commands in the DMBATCH job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.
2. Submit the job while CONNECT:Direct is running.
3. Verify the results.

Batch Interface Example of DELETE TYPE

The following example illustrates using the DELETE TYPE command through the batch interface. In this example, the following commands delete Type records keyed by the names MYALLOC, NEWALLOC, and RPLALLOC from the Type file.

```
* DELETES A SINGLE TYPE RECORD                                    *
   DELETE TYPE WHERE (TYPEKEY=MYALLOC)
* DELETES MULTIPLE TYPE RECORDS                                   *
   DELETE TYPE WHERE (TYPEKEY=(NEWALLOC RPLALLOC))
```

Using the DELETE TYPE Command Through the IUI

To issue the Delete Data Set Type command through the CONNECT:Direct IUI, perform the following steps:

1. Access the Delete Data Set Type Defaults screen by selecting option **DT** from the Administrative Options Menu.

2. Type in the names of the TYPE KEYs that you want to delete and press **ENTER**.

   A list of the deleted records is displayed. If the delete is unsuccessful, CONNECT:Direct displays a list of the records not deleted.

3. Press **PF3/END** to return to the Delete Data Set Type Defaults screen.

Note: This screen will display a message indicating if the delete request was successful.

Using the SELECT TYPE Command

The SELECT TYPE command enables you to examine a record in the Type file. You can specify the search criteria and the form in which the information is presented.

The SELECT TYPE command uses the following format and associated parameters. The required parameters and keywords appear in bold print. Default values for parameters and subparameters are underlined.

Label	Command	Parameters
(optional)	**SELect TYPE**	**WHERE** (
		TYPEKEY = typekey | generic |(list)
		)
		PRint | TABle

Required Parameter

WHERE is the required parameters for the SELECT TYPE command.

WHERE (TYPEKEY=typekey | generic | (list))
specifies which records(s) in the Type file to select.

TYPEKEY=typekey | generic | (list)
specifies the key or list of keys of the records to select.

typekey specifies the name associated with the record being selected. You created the typekey name when the entry was originally added to the Type file. The typekey is a 1–8 character alphanumeric string, with the first character alphabetic.

generic specifies generic selection of type key(s). To specify type keys generically, type a 1–7 character alphanumeric string, with the first character alphabetic, plus an asterisk (*). For instance, if your network includes the type keys SENDDAY, SENDMO, SENDWK, a specification of SEND* will provide information about those keys.

list specifies multiple type keys. A list of type keys can be specified by enclosing them in parentheses.

Optional Parameters

The following parameters are optional:

PRint | TABle
specifies the method of display for the output of the select.

PRint specifies that the output of the SELECT TYPE command is printed rather than displayed. Printed output is in tabular format, the same as that produced by the TABLE parameter. Output is routed to the destination specified in the PRINT keyword of the CONNECT:Direct Signon command.

TABle specifies that the output of the SELECT TYPE command is stored in a temporary file in tabular format and is displayed upon successful completion of the command. The default for the output is TABLE.

Using the SELECT TYPE Command Through the Batch Interface

To use the SELECT TYPE command from the Batch Interface, perform the following steps:

1. Place the SELECT TYPE commands in the DMBATCH job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide.*
2. Submit the job while CONNECT:Direct is running.
3. Verify the results.

Batch Interface Example of SELECT TYPE

The following command selects a record in the TYPE file.

```
SELREC    SELECT TYPE   WHERE (TYPEKEY=DF*)
```

The output follows.

```
============================================================
                    SELECT TYPE DEFAULTS
============================================================
 Type Key  => DF           Date Created => 01/19/1997

       DCB=(DSORG=PS,LRECL=255,BLKSIZE=259,RECFM=V)
```

Using the SELECT TYPE Command Through the IUI

To issue the SELECT TYPE command through the CONNECT:Direct IUI, perform the following steps:

1. Access the Select Data Set Type Defaults screen by selecting option **ST** from the Administrative Options Menu.

2. Enter the type key for each member you want to select.

Note: Refer to the description of parameters for the SELECT TYPE command beginning on page 5-13, or press **PF1** to view the online help.

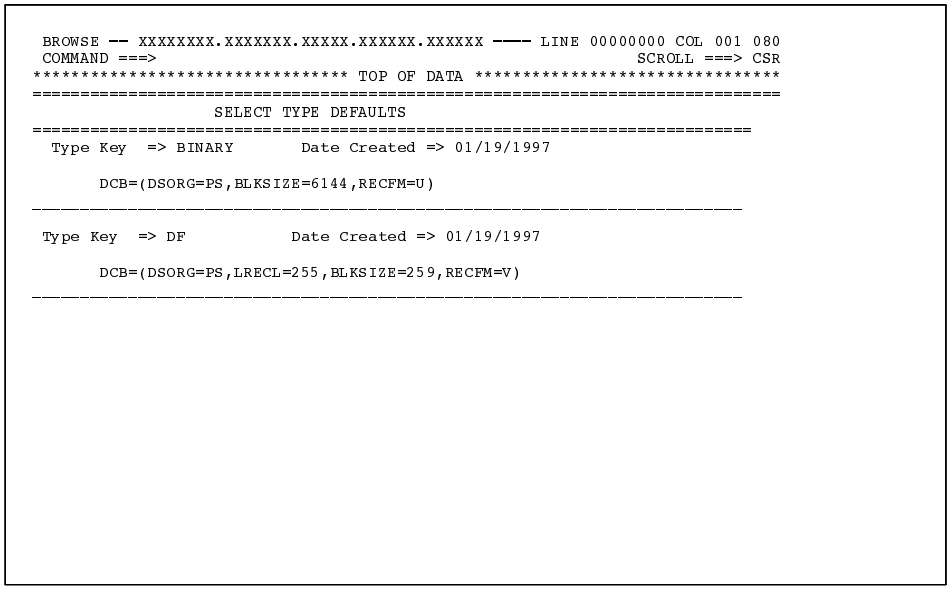
---

3. Indicate the output destination.

4. Provide the requested information and press **ENTER**.

IUI Example of SELECT TYPE

An output destination of **DIS** produces a display like this example.



Chapter 6

Maintaining the Network Map

This chapter contains the following information:

- Contents of the Network Map and examples of node records
- Methods of maintaining the Network Map

Contents of the Network Map

The Network Map identifies the local CONNECT:Direct node and the nodes with which it can communicate. It consists of a local node entry and one or more adjacent node entries. Each entry identifies the communications name and protocol associated with a CONNECT:Direct node.

The source form of the Network Map is generated during the install process and is input to the Network Map load utility, DMCNTMPL. This utility creates the VSAM form of the Network Map that CONNECT:Direct uses.

See the *CONNECT:Direct for VM/ESA Installation Guide* for rules governing the Network Map for cross-domain VTAM definitions.

Local Node Definition

The local node entry defines the logical CONNECT:Direct name of the local CONNECT:Direct DTF and its associated communications name.

The local node entry also contains the name of the transmission queue and the SUPERUSR ID password, if specified. The syntax is shown in the following figure.

```
LOCAL.NODE=(                                                    -
           (node name,communications name,,superuser password)  -
           TCQ=(tcxdsn, tcqdsn)                                 -
           )
```

Types of Parameters

The Network Map local node entry contains two types of parameters; positional and keyword.

Positional Parameters

The following are the positional parameters for the Network Map local node entry.

node name

(first positional parameter) specifies the node name. It specifies the logical name of the local CONNECT:Direct DTF. The name must be from 1–16 alphanumeric characters long.

communications name

(second positional parameter) specifies the communications name. It specifies the name that CONNECT:Direct uses to communicate over the network. The name must be 1–8 characters long.

For **SNA** this field must contain the VTAM APPLID that the local DTF uses for DTF-DTF communications.

null

(third positional parameter) is not used.

superuser password

(fourth positional parameter) specifies the 1–8 character SUPERUSR ID password. The initial value for this field is specified during installation.

The SUPERUSR ID is provided to bypass your usual security system at signon. This bypass may be necessary if CONNECT:Direct is configured improperly, resulting in the inability to signon. SUPERUSR still goes through usual data set verification done by the Stage 2 security exit.

The following is the keyword parameter for the Network Map local node entry.

TCQ=(tcxdsn,tcqdsn)

(required) specifies the names of the two files that together make up the TCQ.

tcxdsn (first subparameter) identifies the data set name of the TCQ index (TCX).

tcqdsn (second subparameter) identifies the data set name of the TCQ.

Defining Local Node as Adjacent Node

The local node must also be defined as an adjacent node for the following reasons:

- To specify the VTAM application IDs to be used for IUI and batch sessions. For more information on how to specify the VTAM application IDs, refer to the *Define APPLID for IUI and Batch Sessions* section of the *CONNECT:Direct for VM/ESA Installation Guide*.
- To provide the ability to run Processes where the local node is both the initiating and target node (PNODE and SNODE). The communications name matches the APPLID defined during installation preparation as described in the *CONNECT:Direct for VM/ESA Installation Guide*. Observe the rules as described in the section *PNODE=SNODE Processing* beginning on page 6-9.

Adjacent Node Definition

Adjacent node entries define nodes in the network with which the local CONNECT:Direct may communicate. Each entry specifies a locally used CONNECT:Direct name, its associated network communications name, and session control parameters for these nodes. The syntax is shown in the following figure.

```
ADJACENT.NODE=(                                                    -
     (nodename,                                                    -
      communications name,                                         -
      session type,                                                -
      security node type,                                          -
      data direction restriction )                                 -
     PARSESS=(max default)                                         -
     ENVIRONMENT=operating environment                             -
     LOGMODE=logmode entry name                                    -
     APPLIDS=(vtam applid1 [,vtam applid2,...] )                   -
     NETID=networkid                                               -
     PNODE.LUS=(luname1 [,luname2,...] )                           -
     SNODE.LUS=(luname1 [,luname2,...] )                           -
     NDMPACE = (#sends,time)                                       -
                )
```

Please read the section *TCP/IP Considerations* beginning on page 6-9 for Network Map entry requirements for TCP/IP nodes.

Types of Parameters

The Network Map adjacent node entry contains two types of parameters: positional and keyword.

Positional Parameters

The following are the positional parameters for the Network Map adjacent node entry.

nodename

(first positional parameter) specifies the node name. It specifies the name the local CONNECT:Direct will use to refer to the partner CONNECT:Direct represented by this entry. This name must be 1–16 characters long. This is a required parameter.

communications name

(second positional parameter) specifies the communications name. It specifies the network name of the partner CONNECT:Direct. It can be either an SNA VTAM APPLID or a TCP/IP port number. The name must be 1–8 characters long. This parameter is optional.

For **SNA** this field must contain the VTAM APPLID the local DTF uses for DTF-DTF communications with the partner CONNECT:Direct.

The communications name is the VTAM APPLID of the remote CONNECT:Direct node (the same name as defined for communications name in Network Map of the remote CONNECT:Direct node).

For **SNA**, this APPLID must be defined as a CDRSC locally and as an APPL to VTAM on the remote system.

For **OS/2 Warp**, if the partner system is a remote node, the communications name is the LU name used to contact the remote CONNECT:Direct.

For **NetWare**, if you are using LU6.2 independent LUs (parallel-session-capable LUs), you can specify a single LU name. In this case, only the named LU is used to communicate between the two CONNECT:Direct nodes, and any session limits placed on that LU impact the number of parallel sessions that can be active between the two nodes. Alternatively, you can leave this field blank and specify SNODE.LUS and PNODE.LUS.

For **TCP/IP**, this field contains the TCP/IP port number of the remote partner CONNECT:Direct. The field need not be used if the partner CONNECT:Direct is initialized using the default TCP/IP port number. This port number refers only to the partner CONNECT:Direct and does not change the port number for the host CONNECT:Direct as defined at initialization. See page 6-10 for more information about TCP/IP port number.

session type

(fourth positional parameter) specifies the session type. It specifies the type of session communications protocol to use for communications with this adjacent node. This parameter is required for OS/400 adjacent nodes and any node using a protocol other than LU0. Valid values are described in the following table.

Session Type	Session Protocol
SNA	LU0
SNUF	LU0 for the OS/400
LU62	LU6.2
TCP	TCP/IP

security node type

(fifth positional parameter) specifies the security node type. It classifies the node as an internal or external node. This optional parameter is provided with the Trusted Node security.

For further information on Trusted Node security refer to page 3-18.

EXTERNAL|EXT specifies an external security classification for this node.

INTERNAL|INT specifies an internal security classification for this node.

data direction restriction
(sixth positional parameter) specifies the data direction restriction. It identifies the copy initiation abilities of this adjacent node with the local node. For further information on data direction restriction, refer to page 3-18. This parameter is optional. Valid data direction values are:

RECEIVE|RECV indicates this node can receive data from the local node.

SEND indicates this node can send data to the local node.

BOTH indicates this node can send data to and receive data from the local node. This is the default.

NONE indicates this node can neither send data to nor receive data from the local node.

Keyword Parameters

The following are the keyword parameters for the Network Map adjacent node entry.

PARSESS=(max, default)
(optional) defines two session control values for parallel session capable nodes. If this parameter is not coded, the node is not considered parallel-session capable; the max and default values are set to 1.

This parameter is required if you do PNODE=SNODE processing. It governs the number of simultaneous connections.

max specifies the maximum number of simultaneous DTF-DTF sessions the local CONNECT:Direct node can have with this adjacent node. The valid range for this subparameter is 1–255. Each session is represented by a corresponding class value. This class value determines

the execution eligibility of a Process. Leave this field blank if parallel sessions are not available.

default specifies the class to be assigned to a Process if one is not specified on the PROCESS statement or at submit time.

Refer to the *CONNECT:Direct Process Guide* for additional details on Process class. In addition, refer to the *Selecting, Queuing, and Recovering Processes* chapter on beginning on page 7-1 for more information about parallel sessions.

ENVIRONMENT=operating environment

identifies the operating system environment of the adjacent node.

Note: This parameter is required only when the session type positional parameter specifies LU6.2 protocol for OS/400 systems.

It can be used for documentation purposes for other protocols. The operating environment variable is limited to six characters.

Valid values are: OS400, MVS, VM, VSE, TANDEM, VMS, VAX, OS2, MSDOS, MSP, NW, NT, PC, ST–VOS, UNIX, and WIN95.

LOGMODE=logmode entry name

identifies the VTAM logmode entry defining the session protocol to be used when communicating with this node. This parameter is required for LU62 nodes only. This parameter is not valid for TCP/IP nodes.

This parameter is optional for LU0 connections. If you specify the LOGMODE parameter for LU0 connections, it indicates that the RUSIZE defined within this LOGMODE is used for any transfer with this node. For a host-to-host transfer, the LOGMODE entry in the VTAM MODETAB of the SNODE is used to determine the RUSIZE. For a host-to-PC transfer, the LOGMODE entry in the host VTAM MODETAB is used.

Refer to the *Preparing VTAM Definitions* chapter and the *VTAM Definitions* appendix in the *CONNECT:Direct for VM/ESA Installation Guide* for more VTAM information.

APPLIDS=(vtam applid1 [,vtam applid2,...])

specifies the VTAM APPLID(s) to be used to establish a session between the local CONNECT:Direct API and the remote CONNECT:Direct DTF. APPLIDs are defined on the local node. This parameter is not valid for TCP/IP nodes; it is valid for MVS, MSP, VSE, and VM nodes only.

NETID=networkid

specifies the 1–8 character network ID for this node. At Process initiation, the network ID provided at session establishment is verified against the network ID specified on the Network Map entry for the adjacent node. If they do not match, the Process is terminated with an error.

For multiple session signons, the network ID for the node signing on is verified against the network ID coded in the Network Map for the node being signed on to. If they do not match, the signon fails.

If this keyword is not coded or the CONNECT:Direct initialization parameter NETMAP.CHECK is set to NO, the network ID will not be checked at Process initiation or multiple session signon.

PNODE.LUS=(luname1 [,luname2,...]))

specifies the logical units used by a remote node to initiate a session with this local node. Do not specify the communications name when you use this parameter.

This parameter applies to OpenVMS nodes and OS/2 or NetWare LU6.2 dependent LUs.

SNODE.LUS=(luname1 [,luname2,...]))

specifies the logical units used for all communications with the remote node.

Communications to nodes not capable of parallel sessions may require the use of a pool of logical units. If an adjacent node is defined in its host environment to use more than one logical unit for communications, then each of the logical unit names that can be used to communicate with the local node must be defined to the local node on the corresponding adjacent node Network Map entry.

This parameter also applies to OS/2 and NetWare LU6.2 dependent LUs if you do not specify the LU name as the communications name.

NDMPACE=(#sends, time)

controls pacing when sending data. Use this parameter for TCP/IP nodes only.

#sends indicates how many sends to issue before a response is required from the receiver. It is a numeric value between 0–1024. If **0** is coded, no response is required. The default is **0**.

time indicates the delay in milliseconds between sending of data blocks. It is a numeric value between 0–1024. If **0** is coded, no delay is used. The default is **0**.

PNODE=SNODE Processing

CONNECT:Direct has the ability to initiate Processes in which the local node is defined as both the initiating and target nodes (PNODE and SNODE). PNODE=SNODE processing is enabled by the insertion of an adjacent node entry in the Network Map where the node name is the same as that defined for the local node entry. Observe the following rules when setting up the adjacent node:

- Define an LU0 communications name for the PNODE=SNODE Netmap entry. The communications name of the adjacent node entry must not specify an LU6.2 logmode entry name.
- The communications name for the adjacent node must be different than the communications name of the local node. If the names are the same, PNODE=SNODE processing is disabled at installation.
- Code the PARSESS=(max, default) parameter in the Network Map adjacent node entry to govern the number of simultaneous PNODE=SNODE connections. The PARSESS parameter is described on page 6-6.

TCP/IP Considerations

A Network Map entry is *not* required for every TCP/IP node. A default entry is used to provide a standard set of parameters. If the standard set of parameters is not adequate, you can code a Network Map entry for that node to override the default entry. If Network Map entries are not used for TCP/IP, you must submit Processes by using the SNODE=TCPNAME=keyword.

Note: If you code the NETMAP.CHECK=TCP initialization parameter, you must define the TCP/IP node in the local Network Map.

The APPLIDS and LOGMODE keywords is not used on any TCP/IP node and should not be coded on the Network Map. A warning is generated for any unneeded keyword or subparameter, and the coded value is ignored when the Network Map is loaded.

TCP/IP Addressing

Each host on a TCP/IP network is assigned a unique 32-bit numeric address known as an IP address. Applications running on a TCP/IP host that connect to the network are assigned one or more port numbers of the IP address.

TCP/IP Port Number

CONNECT:Direct implements a standard TCP client/server architecture. Each CONNECT:Direct in the TCP/IP network can function as both the client and the server simultaneously.

The server establishes a connection with the TCP/IP network and waits for session requests from other CONNECT:Direct clients. When a session request is received and validated, the server accepts the connection to the client.

The CONNECT:Direct client requests a session to be established through the TCP/IP network to a CONNECT:Direct server. Once the session is accepted by the server, a dynamic port is assigned by the network. This dynamically assigned port is used for the actual data transfer. When the data transfer is complete, the port is released back to the network and the session is terminated.

Unless the client override port number is coded on the adjacent node record in the Network Map and the server override port number is coded on the TCP.PORTNUM initialization parameter, CONNECT:Direct will use a predefined TCP/IP port number for both client and server. This port is defined in TCP/IP as a TCP service and should be the same for all CONNECT:Direct TCP/IP nodes in your network. Refer to your TCP/IP implementation documentation for how to define TCP servers. If the predefined TCP/IP port cannot be used it can be overridden for both the client and server.

Client Override Port Number

The client override port number is coded on the adjacent node record in the second positional parameter. The CONNECT:Direct client requests sessions to the CONNECT:Direct server through this port. The port number coded on the adjacent node record is used by client functions only and will not change the port number used by the server functions. A Network Map entry must be coded in your Network Map to use the override port number.

Server Override Port Number

The server override port number is coded on the TCP.PORTNUM initialization parameter. The CONNECT:Direct server waits for a CONNECT:Direct client request on this port. The initialization port number is used by server functions only and will not change the port number used by the client functions.

Defining a TCP/IP Default Entry

The following example defines a TCP/IP default record using ADJACENT.NODE definitions:

```
/*                                                                  */
/* The following entry is for the TCP/IP default entry              */
/*                                                                  */
 ADJACENT.NODE=(PARSESS=(8 1)                -
  ENVIRONMENT=MVS                            -
  (TCP.IP.DEFAULT, 2048,, TCP))
```

Note: For a TCP/IP default definition, the nodename must be TCP.IP.DEFAULT and the session type must be TCP.

In the previous example, nodename=TCP.IP.DEFAULT, communications name=2048, and session type=TCP. For additional information about adjacent node definitions, see the *Adjacent Node Definition* section on page 6-3.

When a process is submitted, the port number value is determined in the following order:

1. If the nodename within a process is defined in the netmap, the port number associated with the nodename entry is used.

2. If the nodename within a process is not defined in the netmap, the port number associated with the default entry is used.

3. If there is no port number in the communications name field of the default entry, the TCP.PORTNUM initialization parameter is used.

4. If the TCP.PORTNUM initialization parameter is not defined, then the port number defaults to 1364.

The PARSESS value for the SNODE is determined in the following order:

1. If the nodename within a process is defined in the netmap, the PARSESS value associated with the nodename entry is used.

2. If the nodename within a process is not defined in the netmap, the PARSESS value associated with the default entry is used.

3. If there is no PARSESS value in the default entry, the PARSESS value defaults to (1 0). A PARSESS value of (1 0) means that processes to the nodes for which the default entry is used will be single-threaded.

Examples of Local and Adjacent Node Records

This section contains examples of local and adjacent node records for various platforms. See the *Compatibility Chart* of the *CONNECT:Direct for VM/ESA Release Notes* for connectivity protocols supported for the various platforms.

Local Node and Corresponding Adjacent Node Record

The following is an example of a local node definition and its corresponding definition as an adjacent node. The local and adjacent node names are the same (CD.DALLAS) but the local and adjacent

communications names (NDMAPP1 and NDMAPP2) are different. The local node shows a superuser password of XYZZY.

```
LOCAL.NODE=((CD.DALLAS,NDMAPP1,,XYZZY) -
          TCQ=(DSC.DALLAS.TCX DSC.DALAS.TCQ))
/*                                                                */
/*  THE FOLLOWING ENTRY IS FOR THE LOCAL NODE                     */
/*                                                                */
ADJACENT.NODE=(PARSESS=(12 2) (CD.DALLAS,NDMAPP2) -
          APPLIDS=(NAI01   NAI02  NAI03  CDDD12 -
                   CDDD17  CDDD18 CDDD32   CDDD41  CDDD42))
```

VM Adjacent Node Examples

The following examples show how to define adjacent VM nodes.

TCP/IP

The following example shows three adjacent node definitions with session protocol types of TCP/IP.

- The first, with a TCP net name of VM.CD.CHICAGO, specifies the default TCP/IP port number by leaving the communications name positional parameter null.
- The second, with a TCP/IP net name of MVS.CD.DALLAS, specifies a TCP/IP port number of 4444.
- The third, with a TCP/IP net name of VM.CD.AUSTIN, specifies a TCP/IP port number of 4443 and an IP address of 199.8.8.8.

```
ADJACENT.NODE=(PARSESS=(4, 2) -
       (VM.CD.CHICAGO,,199.1.4.51,TCP) -
       ENVIRONMENT=MVS)

ADJACENT.NODE=(PARSESS=(4, 2) -
       (VM.CD.DALLAS,4444,,TCP) -
       ENVIRONMENT=MVS)

ADJACENT.NODE=(PARSESS=(4, 2) -
       (VM.CD.AUSTIN,4443,199.8.8.8,TCP) -
       ENVIRONMENT=MVS)
```

SNA LU6.2

The following example shows an adjacent node definition for a node named CD.LAMVS with a communications name (applid) of APPLLAI and a session protocol type of LU6.2. The operating environment of this adjacent node is MVS, and the VTAM logmode entry which defines the

session protocol to be used when communicating with this node is LU62M0D4. The LOGMODE parameter is required for LU6.2.

```
ADJACENT.NODE=(PARSESS=(4,  2) -
          (CD.LAMVS,APPLLAI,,LU62) -
          ENVIRONMENT=MVS LOGMODE=LU62MOD4 -
          APPLIDS=(CDDD2,CDDD3,CDDD4))
```

VM SNA LU0 Adjacent Node Example

The following example shows an adjacent node named CD.BOSTONVM, with a communications name (applid) of CDDD16.

```
ADJACENT.NODE=(PARSESS=(4 2)  (CD.BOSTONVM,CDDD16) -
               APPLIDS=(CDAPI01  CDAPI02   CDAPI03  CDAPI04 -
               CDAPI05  CDAPI06   CDAPI07  CDAPI08  CDAPI09))
```

VSE SNA LU0 Adjacent Node Example

The following example shows an adjacent node named CD.DALLASVSE, with a communications name (applid) of CDDD22.

```
ADJACENT.NODE=(PARSESS=( 6   2) -
               (CD.DALLASVSE,CDDD22) -
       APPLIDS=(CDAPI01  CDAPI02  CDAPI03  CDAPI04  CDAPI05))
```

OpenVMS Adjacent Node Example

The following example shows an adjacent node named CD.DALLAS.VMS. The SNODE.LUS parameter specifies the logical unit names used by the local node to initiate a session with this remote node.

```
ADJACENT.NODE=((CD.DALLAS.VMS) -
PARSESS=(8 1) -
PNODE.LUS=(N91LU09  N91LU0A  N91LU0B  N91LU0C -
          N91LU0D  N91LU0E  N91LU0F  N91LU10) -
SNODE.LUS=(N91LU07  N91LU08))
```

OS/2 WARP Adjacent Node Examples

The following example shows how to define adjacent OS/2 nodes.

LU6.2

The following example shows an adjacent OS/2 node with a session protocol type of LU6.2 and logmode entry name of SA62D1K, second position is the APPLID. The LOGMODE parameter is required for LU6.2. Notice that the third positional parameter is required to be CDOS2.

```
ADJACENT.NODE=((M92.05,M592LU05,CDOS2,LU62) -
      LOGMODE=SA62D1K)
```

UNIX Adjacent Node Examples

The following examples show how to define adjacent UNIX nodes.

TCP/IP

The following example shows two adjacent node definitions with session protocol types of TCP/IP.

- The first, with the TCP net name of UNIX.CD.CHICAGO, specifies the default TCP/IP port number by leaving the communications name positional parameter null.
- The second, with the TCP net name of UNIX.CD.DALLAS, specifies a TCP/IP port number of 5555 and an IP address of 199.5.5.5.

```
ADJACENT.NODE=(PARSESS=(6, 2) -
       (UNIX.CD.CHICAGO,,,TCP) -
       ENVIRONMENT=UNIX)

ADJACENT.NODE=(PARSESS=(6, 2) -
       (UNIX.CD.DALLAS,5555,199.5.5.5,TCP) -
       ENVIRONMENT=UNIX)
```

Notice that there are no APPLID or LOGMODE keywords used for any TCP/IP node. A warning is generated for any unneeded keyword or subparameter, and the coded value is ignored.

LU6.2

The following example shows an adjacent UNIX node with a communications name (applid) of D1UNIX and a session protocol type

of LU6.2. The logmode entry name is LU62MODE. The LOGMODE parameter is required for LU6.2.

```
ADJACENT.NODE=(PARSESS=(6, 2) -
        (UNIX.LU62.DALLAS,D1UNIX,,LU62) -
        LOGMODE=LU62MODE  -
        ENVIRONMENT=UNIX)
```

OS/400 Adjacent Node Examples

The following examples show how to define adjacent OS/400 nodes.

OS/400 SNUF

The following example shows an adjacent node named AS400.CD.TX with a remote library name of LBNAME and session protocol type of LU0 (SNUF). The SNODE.LUS parameter defines the dependent LU pool.

```
ADJACENT.NODE=(PARSESS=(4, 2) -
        (AS400.CD.TX,,LBNAME,SNUF) -
        SNODE.LUS=(N11LU01,N11LU02,N11LU03,N11LU04))
```

LU6.2 with Independent LU

The following example shows an adjacent node named AS400.CD.LA with an independent LU communications name of APPLLA1, a remote library name of CDLIB1, session protocol type of LU6.2, and a logmode entry name of LU62MOD2. The ENVIRONMENT=OS400 parameter is required for OS/400 nodes using the LU6.2 protocol. The LOGMODE parameter is required for LU6.2 protocol.

```
ADJACENT.NODE=(PARSESS=(6, 2) -
        (AS400.CD.LA,APPLLA1,CDLIB1,LU62) -
        ENVIRONMENT=OS400 LOGMODE=LU62MOD2)
```

LU6.2 with Dependent LU

The following example shows an adjacent OS/400 node named AS400.CD.NY with a remote library name of CDLIB1, a session protocol type of LU6.2, and a logmode entry name of LU62MOD3.

The SNODE=LUS parameter defines the dependent LU pool. The ENVIRONMENT=AS400 parameter is required for OS/400 nodes using

the LU6.2 protocol. The LOGMODE parameter is required for LU6.2 protocol.

```
ADJACENT.NODE=(PARSESS=(4,  2) -
       (AS400.CD.NY,,CDLIB1,LU62) -
       ENVIRONMENT=OS400 LOGMODE=LU62MOD3 -
       SNODE.LUS=(NYLU01,NYLU02,NYLU03,NYLU04))
```

NetWare Adjacent Node Examples

The following examples show how to define adjacent NetWare nodes.

LU6.2 with Independent LU Name

The following example shows a designated LU name. Enter the LU name (NWLU01) in the Communications Name field. You must specify LU62 as the session type. The LOGMODE parameter is required for LU 6.2 protocol.

```
ADJACENT.NODE=(PARSESS=(6, 2) -
       (NW.LU62.DALLAS, NWLU01, , LU62)  -
       ENVIRONMENT=NW,LOGMODE=NDM621K
```

LU6.2 with Dependent LUs

The following example shows the use of the SNODE.LUS parameter that allows multiple sessions between the two nodes using dependent LUs. The session type must be LU62. The LOGMODE parameter is required for LU6.2 protocol.

```
ADJACENT.NODE=(PARSESS=(4,  2) -
       (NW.CD.NY,,,LU62) -
       ENVIRONMENT=NW,LOGMODE=LU62MOD3 -
       SNODE.LUS=(NYLU01,NYLU02,NYLU03,NYLU04))
```

LU6.2 with Independent and Dependent LUs

The following example shows two adjacent node definitions with session protocol types of TCP/IP. The first example shows a TCP net name of AS400.NODE that specifies the TCP/IP port number of 5555 and an IP address of 192.5.10.20.

```
ADJACENT.NODE=((AS400.NODE,5555,192.5.10.20,TCP) -
     PARSESS=(4, 2)                              -
     ENVIRONMENT=AS400)
```

The second example shows a TCP net name of AS400.NODE2 that specifies the default TCP/IP port number by leaving the communications name positional parameter null and it specifies an IP address of 192.5.10.30.

```
ADJACENT.NODE=((AS400.NODE2,,192.5.10.30,TCP) -
     PARSESS=(4, 2)                           -
     ENVIRONMENT=AS400)
```

TCP/IP

The following example shows two adjacent node definitions with session protocol types of TCP/IP.

- The first, with the TCP net name of NW.CD.CHICAGO, specifies the default TCP/IP port number by leaving the communications name positional parameter null and specifies an IP address of 199.4.4.4.
- The second, with the TCP net name of NW.CD.DALLAS, specifies a TCP/IP port number of 5555 and an IP address of 199.5.5.5.

```
ADJACENT.NODE=(PARSESS=(6, 2) -
       (NW.CD.CHICAGO,,199.4.4.4,TCP) -
        ENVIRONMENT=NETWARE)
ADJACENT.NODE=(PARSESS=(6, 2) -
       (NW.CD.DALLAS,5555,199.5.5.5,TCP) -
        ENVIRONMENT=NETWARE)
```

There are no APPLID or LOGMODE keywords used for any TCP/IP node. A warning is generated for any unneeded keyword or subparameter, and the coded value is ignored.

Windows NT Adjacent Node Examples

This example shows how to define adjacent Windows NT nodes.

TCP/IP

The following example shows two adjacent node definitions with session protocol types of TCP/IP.

- The first, with the TCP net name of NW.CD.CHICAGO, specifies the default TCP/IP port number by leaving the communications name positional parameter null and specifies an IP address of 199.4.4.4.
- The second, with the TCP net name of NW.CD.DALLAS, specifies a TCP/IP port number of 5555 and an IP address of 199.5.5.5.

```
ADJACENT.NODE=(PARSESS=(6, 2) -
        (NT.CD.CHICAGO,,199.4.4.4,TCP) -
        ENVIRONMENT=NT)
ADJACENT.NODE=(PARSESS=(6, 2) -
        (NT.CD.DALLAS,5555,199.5.5.5,TCP) -
        ENVIRONMENT=NT)
```

Notice that there are no APPLID or LOGMODE keywords used for any TCP/IP node. A warning is generated for any unneeded keyword or subparameter, and the coded value is ignored.

Updating the Network Map

After making changes to the source form of the Network Map, perform the following steps to convert the source form of the Network Map into a form that CONNECT:Direct can process.

1. Stop CONNECT:Direct.

   Note: While you can dynamically update the Network Map source, you cannot update the Network Map as CONNECT:Direct is operating.

2. Enter the following information, where CCUU is the virtual address of the VSAM disk containing the CONNECT:Direct system files and MODE is the one-letter mode that you use to access the same disk. Replace *nn* in CDV32*nn* with the maintenance level of CONNECT:Direct for VM/ESA.

   ```
   GLOBAL LOADLIB CDV32nn
   ACCESS CCUU MODE
   DLBL IJSYSCT MODE DSN MASTCAT (PERM
   ```

3. Enter the following to delete and redefine an empty VSAM cluster for the Network Map.

   ```
   AMSERV NETMAP
   ```

4. Enter the following to run the NTMPLOAD EXEC customized during *Step 6—Create and Load the Network Map* of the installation process. Running the NTMPLOAD EXEC converts the source form of your Network Map into the VSAM file created in the last step.

```
NTMPLOAD
```

5. The following completion message displays.

```
SMLA000I THE CONNECT:DIRECT NETWORK MAP HAS BEEN SUCCESSFULLY
LOADED
```

   If other messages display, they are indicative of syntax errors in the source form of the Network Map. In that case, you need to edit the source form of the Network Map to correct the syntax errors and go back to Step 3. on page 6-19.

6. Once you have successfully loaded the Network Map, restart CONNECT:Direct.

Updating the Network Map Dynamically Using the UPDATE NETMAP Command

You can also update the Network Map without deleting and redefining it. You can update the Network Map source without stopping CONNECT:Direct, by using the UPDATE NETMAP command.

As with most commands, you can execute the command through a batch job or through the IUI. Both methods use $$ACTION verbs as part of the Network Map source.

Note: This method of updating the Network Map is only available for adjacent node entries.

The format of the UPDATE NETMAP command follows.

Label	Command	Parameters
(optional)	**UPDate NETMAP**	**WHERE** (
		NETINput = filename FILETYPE(member name)
		LINK=(userid,password,accessmode,CUU))
		NETLOG =[ddname | NONE]

(continued)

Label	Command	Parameters
		)
		DIS | PRT

Required Parameter

WHERE is the only required parameter.

WHERE (NETINput = filename filetype (member name)
LINK = (userid,password,accessmode,CUU))
NETLOG = [ddname | NONE])
specifies the Network Map source file and where the update activity is to be reported.

NETINput = filename filetype (member name) specifies the name of the Network Map source file. The Network Map source can contain multiple basic action verbs, multiple special purpose action verbs, or a combination of both. The source used to update the Network Map can be the entire Network Map source or a subset of it.

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

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

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

mode specifies the link access mode.

The access modes are W (primary read/write access), M (primary multiple access), R (primary read only), RR (primary and secondary read only access), WR (primary read/write access; alternate read only access), MR (primary multiple access; alternate read only access), and MW (primary multiple access; alternate read/write only access).

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

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

NETLOG = [ddname | NONE] specifies where the update activity is to be reported. If the field is left blank the update activity is reported to the NDMLOG data set. Regardless of which option is selected the activity is recorded in the Statistics file as WTO records. **ddname** specifies the data definition name allocated to the CONNECT:Direct DTF where the update activity is to be reported. **NONE** specifies that no update activity is reported.

Optional Parameter

DIS | PRT

specifies the output destination.

DIS indicates that the activity is to be reported in display format, either to the screen for IUI requests or to the DDNAME for batch requests.

PRT indicates that the output is to be routed to SYSOUT for batch requests and to the print destination specified by the PRINT FILE DESTINATION parameter in the signon defaults.

Using $$ACTION VERBS

Add $$ACTION verbs to the Network Map source described in the NETINPUT parameter of the UPDATE NETMAP command. Each verb defines the action to take for the node entry immediately following the

action verb. There are three basic action verbs and three special purpose action verb pairs.

$$INSERT
inserts the following node into the Network Map.

$$UPDATE
updates the following existing Network Map node entry. Node entry updates are performed as a replacement at the keyword level; therefore, updates of list-type keywords, like APPLIDS, require the entire list to be specified.

$$DELETE
deletes the following existing Network Map node entry.

$$SYNTAX and **$$ENDSYNTAX**
performs a syntax check of the Network Map control statement(s) following this verb.

$$VERIFY and **$$ENDVERIFY**
verifies that the node definition(s) following this verb match those in the Network Map.

$$BLKxxxxxx and **$$ENDxxxxxx**
performs the basic action verb defined by *xxxxxx* for the block of node entries following this verb. Replace *xxxxxx* with either INSERT, UPDATE, or DELETE.

Update Using the CONNECT:Direct Batch Utility

To issue the UPDATE NETMAP command through the CONNECT:Direct batch utility, perform the following steps:

1. Change the Network Map source using the $$ACTION verbs.
2. Place the UPDATE NETMAP commands in the DMBATCH job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.
3. Submit the job while CONNECT:Direct is running.
4. Correct any errors identified on the activity report and resubmit if necessary.

Update Through the CONNECT:Direct IUI

To issue the UPDATE NETMAP command through the CONNECT:Direct IUI, perform the following steps:

Note: Updating the Network Map through the IUI can take significant time. For mass updates, you should consider batch processing.

1. Change the Network Map source using the $$ACTION verbs.
2. Ensure that CONNECT:Direct is running.
3. Access the Update Network Map screen by selecting option **UNM** from the Administrative Options Menu.

```
node.name           UPDATE NETWORK MAP                    hh:mm
CMD ==>
       _


         WARNING: THIS COMMAND CAN TAKE SIGNIFICANT TIME. FOR MASS
                  UPDATES, BATCH PROCESSING SHOULD BE CONSIDERED!


 ENTER NETMAP INPUT FILE NAME:
 FILENAME: ____________     FILETYPE: ____________

 (LINK INFORMATION)
 USERID:  _________  PASSWORD:  _________
 LINK MODE:  ________
```

4. Type the Network Map source file name and the desired optional parameters and press **ENTER**. Unless you select the PRINT option on the Update NETMAP screen, the report routes to your terminal.
5. Correct any errors identified on the activity report and re-enter if necessary.
6. Verify the results.

$$ACTION Verb Examples

These examples illustrate updating the network map using action verbs.

$$INSERT Example

The following $$INSERT command inserts an adjacent node into the Network Map.

```
$$INSERT
  ADJACENT.NODE=((CD.NODE2 , APPLID2  ) -
           PARSESS=(5,2) -
          APPLIDS=(RAPPL1))
```

The output follows.

```
= = > * * *   S T A R T  N E T M A P  U P D A T E   * * *
= = > DATE:  04.20.1998  TIME=14:59:26
= = > SMUPNLGI LOGGING ACTIVE  -  LOG DDNAME=NDMLOG
============================================================SMUPSTAI *
$$INSERT
  ADJACENT.NODE=(( CD.NODE2 , APPLID2 )    -
              PARSESS=(5,2)               -
             APPLIDS=(RAPPL1 ))
= = >    SMUP032I APPLIDS RECORD INSERTED
= = >    SMUP034I ADJACENT.NODE RECORD INSERTED
= = >    SMUP008I REQUEST SUCCESSFUL FOR NODE=CD.NODE2
============================================================SMYPSTAI *
= = >    SMUP000I NETWORK MAP PROCESSING COMPLETED
= = >    DATE:  04.20.1998  TIME=14:59:26
= = > * * *   E N D  N E T M A P  U P D A T E    * * *
```

The first message shows that logging is not requested; therefore, CONNECT:Direct does not keep a record of the transaction (except in the statistics file). The last messages indicate that the information for the specified adjacent node was successfully inserted.

$$UPDATE Example

The following $$UPDATE command updates an adjacent node in the Network Map by adding the RAPPL2 applid and changing the maximum parallel sessions to four.

```
$$UPDATE
  ADJACENT.NODE=(( CD.NODE2 , APPLID2 )   -
           PARSESS=(4,2)                 -
          APPLIDS=(RAPPL1 RAPPL2 ))
```

The output follows.

```
= = > * * *   S T A R T  N E T M A P  U P D A T E   * * *
= = > DATE:  04.20.1998  TIME=14:59:26
= = > SMUPNLGI LOGGING ACTIVE  -  LOG DDNAME=NDMLOG
============================================================SMUPSTAI *
$$UPDATE
  ADJACENT.NODE=(( CD.NODE2 , APPLID2 )    -
             PARSESS=(5,2)                -
            APPLIDS=(RAPPL1 ))
= = >    SMUP032I APPLIDS RECORD UPDATED
= = >    SMUP034I ADJACENT.NODE RECORD UPDATED
= = >    SMUP008I REQUEST SUCCESSFUL FOR NODE=CD.NODE2
============================================================SMYPSTAI *
= = >    SMUP000I NETWORK MAP PROCESSING COMPLETED
= = >    DATE:  04.20.1998  TIME=14:59:26
= = > * * *   E N D  N E T M A P  U P D A T E    * * *
```

The first message shows that logging is requested and that a record of the transaction will be recorded in NDMLOG. The last messages indicate that the adjacent node information was successfully updated.

$$DELETE Example

The following $$DELETE command deletes an adjacent node from the Network Map.

```
$$DELETE
 ADJACENT.NODE=((CD.NODE2 , APPLID2) -
           PARSESS=(4,2) -
          APPLIDS=(RAPPL1 RAPPL2))
```

The output follows.

```
= = > * * *   S T A R T  N E T M A P  U P D A T E   * * *
= = > DATE:  04.20.1998  TIME=14:59:26
= = > SMUPNLGI LOGGING ACTIVE  -  LOG DDNAME=NDMLOG
============================================================SMUPSTAI *
$$DELETE
  ADJACENT.NODE=(( CD.NODE2 , APPLID2 )    -
             PARSESS=(5,2)                -
            APPLIDS=(RAPPL1 ))
= = >    SMUP032I APPLIDS RECORD DELETED
= = >    SMUP034I ADJACENT.NODE RECORD DELETED
= = >    SMUP008I REQUEST SUCCESSFUL FOR NODE=CD.NODE2
============================================================SMYPSTAI *
= = >    SMUP000I NETWORK MAP PROCESSING COMPLETED
= = >    DATE:  04.20.1998  TIME=14:59:26
= = > * * *   E N D  N E T M A P  U P D A T E    * * *
```

The first message indicates that logging is requested; therefore, CONNECT:Direct does not keep a record of the transaction. The last messages indicate that the applids and adjacent node records are successfully deleted.

The following $$SYNTAX command performs a syntax check on the specified nodes.

```
$$SYNTAX
  LOCAL.NODE=((CD.NODE1 , APPLID1 ,, SUPERUSR) -
                   TCQ=( TCQ TCX  ))
 ADJACENT.NODE=((CD.NODE1 , APPLID1) -
           PARSESS=(5,2)  -
          APPLIDS=(LAPPL1  LAPPL2  LAPPL3))
 ADJACENT.NODE=((CD.NODE2 , APPLID2)  -
          PARSESS=(5,2) -
         APPLIDS=(RAPPL1 ))
 $$ENDSYNTAX
```

The output follows.

```
     ==> * * *   S T A R T  N E T M A P  U P D A T E   * * *
     ==> DATE:  02/27/1998  TIME=13:49:16
(1)  ==> SMUPNLGI LOGGING INACTIVE  -  LOG DDNAME=NDMLOG
=======================================================================
     $$SYNTAX
(2)  ==> SMUP011I "SYNTAX   " ACTION STARTED
=======================================================================
     LOCAL.NODE=(( CD.NODE1 , APPLID1  ,,  SUPERUSR) -
                  TCQ=( TCQ TCX  ))
(3)  ==> SMUP005I  LOCAL.NODE RECORD PROCESSING NOT ALLOWED - BYPASSED
=======================================================================
     ADJACENT.NODE=(( CD.NODE1 , APPLID1  ) -
           PARSESS=(5,2)  -
          APPLIDS=(LAPPL1  LAPPL2  LAPPL3))
(4)  ==> SMUP008I REQUEST SUCCESSFUL FOR NODE=CD. NODE1
=======================================================================
     ADJACENT.NODE=(( CD.NODE2 , APPLID2 ) -
           PARSESS=(5,2)  -
          APPLIDS=(RAPPL1))
(4)  ==> SMUP008I REQUEST SUCCESSFUL FOR NODE=CD. NODE2
=======================================================================
     $$ENDSYNTAX
(5)  ==> SMUP012I "SYNTAX   " ACTION STOPPED
=======================================================================
     ==> SMUP001I WARNINGS GENERATED DURING NETMAP PROCESSING
     ==> SMUP001I NETWORK MAP PROCESSING COMPLETED
     ==> DATE: 02/27/1998  TIME=13:49:16
     ==> * * *   END NETMAP UPDATE                * * *
```

The messages are numbered in the example for clarification; they are not numbered on the actual output.

1. Logging is requested. A record is kept of the transaction.
2. Syntax check of Network Map control statements starts.
3. No processing is allowed against the local node record.
4. Requests for syntax checking on nodes are successful.
5. Syntax checking completes.

$$VERIFY Example

The following $$VERIFY command verifies the definition of the specified adjacent node record prior to updating the Network Map.

```
$$VERIFY
 ADJACENT.NODE=((CD.NODE2 , APPLID2) -
           PARSESS=(5,2)  -
          APPLIDS=(RAPPL1))
$$ENDVERIFY
```

The resulting output follows.

```
    = = > * * *   S T A R T  N E T M A P  U P D A T E   * * *
    = = >    DATE:  04.20.1998  TIME=14:59:26
(1) = = >    SMUPNLGI LOGGING ACTIVE  -  LOG DDNAME=NDMLOG
============================================================SMUPSTAI *
    $$VERIFY

(2) = = > SMUP011I "VERIFY    " ACTION STARTED
============================================================SMUPSTAI *
    ADJACENT.NODE=(( CD.NODE2 , APPLID2 )    -
               PARSESS=(5,2)                 -
               APPLIDS=(RAPPL1 ))
(3) = = >    SMUP032I APPLIDS DID NOT MATCH
    = = >    SMUP034I ADJACENT.NODE RECORD DID NOT MATCH
    = = >    SMUP008I RECORDS DO NOT MATCH VERIFICATION FAILED -
                      FOR NODE=CDNODE2
============================================================SMYPSTAI *
    $$ENDVERIFY
(4) = = > SMUP012I "VERIFY    " ACTION STOPPED
============================================================SMYPSTAI *
    = = >    SMUP000I NETWORK MAP PROCESSING COMPLETED
    = = >    DATE:  04.20.1998  TIME=14:59:26
    = = > * * *   E N D  N E T M A P  U P D A T E    * * *
```

The messages are numbered in the example for clarification; they will not be numbered on the actual output.

1. Logging was not requested. No record will be kept of the transaction.

2. Verification of the node definition to the Network Map file has started.

3. The applids and adjacent node records did not match the Network Map file definitions.

4. Verify has completed.

Chapter 7

Selecting, Queuing, and Recovering Processes

This chapter provides information about the following:

- The Transmission Control Queue (TCQ)
- Process selection
- TCQ Status and State Values for the logical queues
- Process recovery and the Copy Checkpoint/Restart procedure

Transmission Control Queue

CONNECT:Direct uses the TCQ to:

- Store submitted Processes
- Control Process execution as CONNECT:Direct operates

The TCQ consists of two VSAM Relative Record Data Sets (RRDS) and an in-memory queue, which controls access. The two data sets are the TCQ and the TCX, which is a space map for the TCQ. These files are inseparable.

TCQ Logical Queues

The TCQ is divided into four logical queues:

- Wait
- Execution

- Hold
- Timer

You can access the queues and manipulate Processes through the following CONNECT:Direct commands:

- CHANGE PROCESS
- DELETE PROCESS
- FLUSH PROCESS
- SELECT PROCESS
- SUSPEND PROCESS

Refer to the *CONNECT:Direct for VM/ESA User's Guide* for more information on these commands.

How Processes are Routed

The RETAIN, HOLD, and STARTT parameters cause Processes to be routed in the way described in the following table.

Parameters	Queue	Comments
None	Wait	Process remains on the Wait queue until CONNECT:Direct can start a session with the SNODE at which time it moves to the Execution queue.
RETAIN=INITIAL	Hold	Process automatically executes each time CONNECT:Direct is initialized with TCQ=WARM.
RETAIN=YES	Hold	A copy of the Process is kept in the Hold queue after it has executed. The Process will not execute again until it is released by a CHANGE PROCESS command.
HOLD=YES	Hold	Process remains in the Hold queue until someone releases the Process.
HOLD=CALL	Hold	Process is automatically moved from the Hold queue to the Wait queue when the SNODE contacts the node on which the Process resides.
STARTT	Timer	When the scheduled time and date arrive, the Process is put on the wait queue and is available for execution.

RETAIN=INITIAL is useful for Processes that contact other CONNECT:Direct nodes each time CONNECT:Direct completes initialization. This will cause any work queued on the remote node for the local node to begin.

RETAIN=YES when combined with STARTT can be used to run a Process at a periodic interval. For example, RETAIN=YES and

START=(Tuesday, 3pm) will start the Process every Tuesday at 3 pm; RETAIN=YES and START=(,12:00) will start the Process each day at noon.

The Timer queue also is used for session retry and file allocation retry based on CONNECT:Direct initialization parameters specified by a particular installation. Once retry limits have been exhausted, the Process is moved to the Hold queue with an HE status.

Process Selection

CONNECT:Direct puts submitted Processes in the appropriate logical queue based on the Process statement parameters that affect scheduling:

- RETAIN=Yes | No | Initial
- HOLD=Yes | No | Call
- START=((date | day)(,hh:mm:ssXM))

The parameters are explained in the *CONNECT:Direct Process Guide* and the *CONNECT:Direct for VM/ESA User's Guide*.

Processes are selected in a first-in first-out manner for execution within process class and priority as sessions are available. The following section provides information on the method that CONNECT:Direct uses to select Processes for execution.

Planning for Parallel Sessions

CONNECT:Direct uses the parallel sessions capability of VTAM so that multiple Processes can execute simultaneously between any two CONNECT:Direct nodes. Process selection for each parallel session is based on a class the user specifies on a Process.

Note: Parallel sessions support requires specification of PARSESS=YES in the VTAM application definition for both nodes. If two nodes have differing values for parallel sessions, transfers are limited by the maximum number of sessions in the ADJACENT.NODE definition on the node where the Process was submitted.

The maximum number of sessions between two nodes is defined in the Network Map. Because each session has a corresponding class value, the maximum number of sessions and maximum number of classes are equal. Selection of a Process for execution in a given node is based on Process priority (the PRTY parameter of the Process statement) within session class. User-specified class values allow a Process to execute on

the session having the matching class value or on sessions with higher class values. The default class is the value specified in the CONNECT:Direct Network Map.

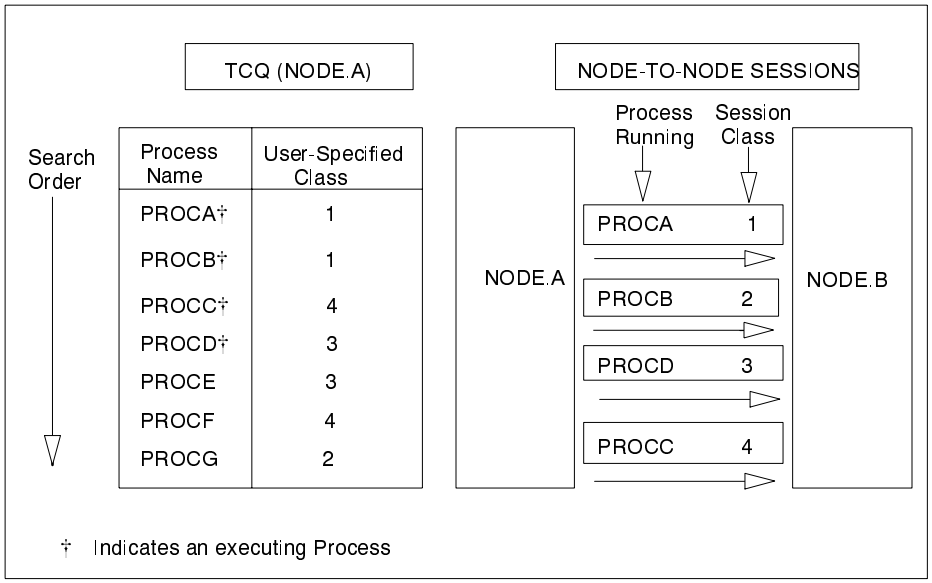
When one Process completes, CONNECT:Direct selects another Process (if any) to run on the available session. CONNECT:Direct searches until it finds the first Process with a class eligible to execute on the available session.

A typical use for classes is to define critical Processes with low class values so that more sessions are available for their execution. Higher class numbers can be specified for time-consuming Processes; this allows sessions with corresponding lower class numbers to become available more frequently. An example of selection by class follows.

Example of Parallel Processes

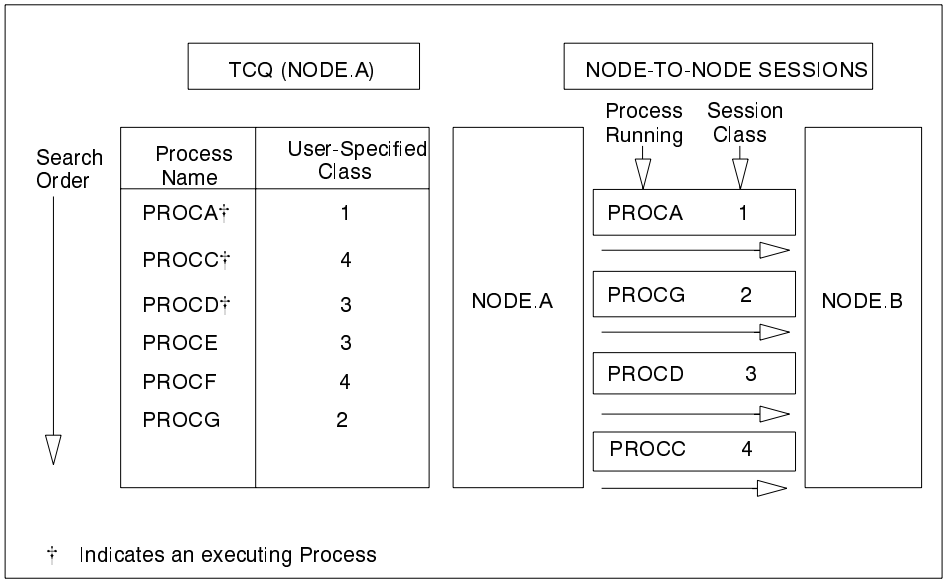
CONNECT:Direct has just been brought up. Seven Processes have been submitted for NODE.B and are ready to run. All Processes have a user-specified class value and the same priority. Class determines which session CONNECT:Direct will select. The Network Map has been defined so a maximum of four sessions can be started between NODE.A and NODE.B. Each session between NODE.A and NODE.B has its own corresponding unique class number.

1. In the following figure, NODE.A simultaneously starts four sessions. Processes are shown in the order they appear in the queue. Because PROCA and PROCB's user-specified class is 1, they can run on the class 2, 3, or 4 sessions, if needed.

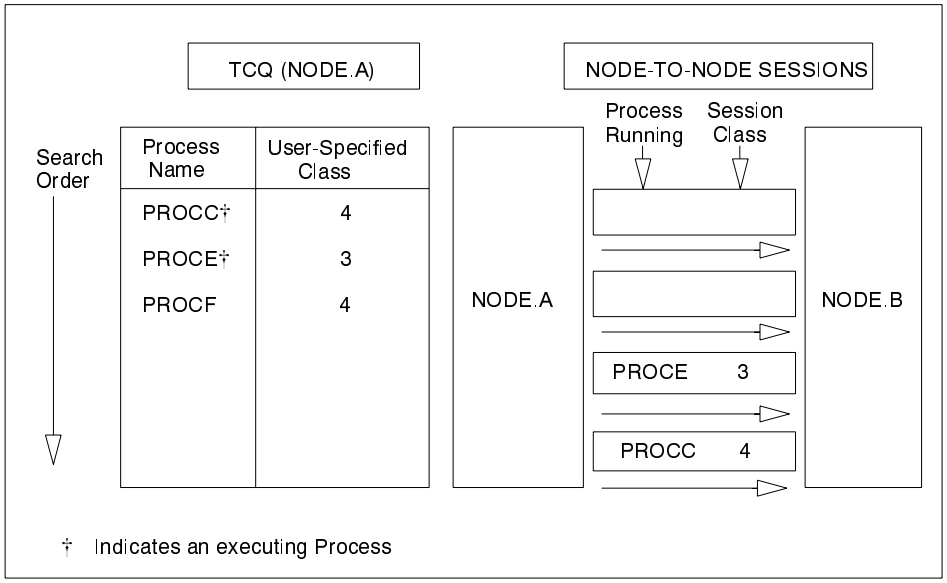


2. In the following figure, PROCB has completed execution, making a session available. CONNECT:Direct looks through the TCQ for the

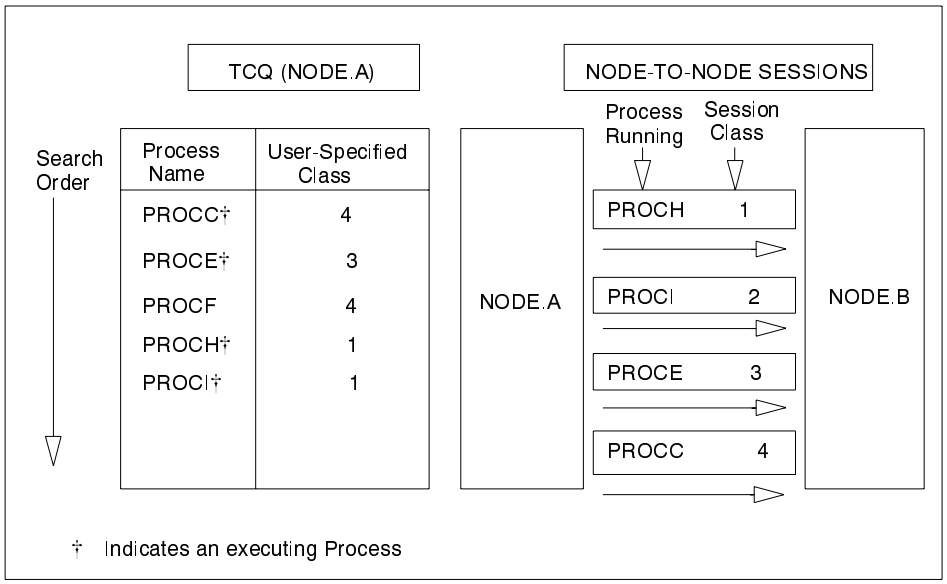
first eligible Process for that session. PROCG is the next Process available to run on the class 2 session because all other Processes have a class value higher than 2.



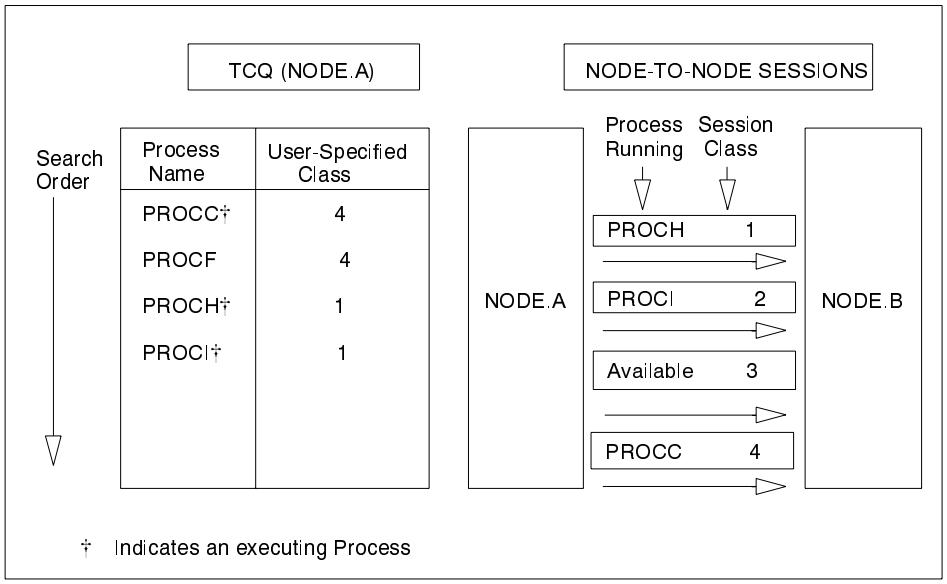
3. In the figure in step 3, PROCA, PROCG, and PROCD have completed execution. Note that sessions for class 1 and 2 cannot be used at this time because only a Process for class 4 is left in the queue.



4. In the following figure, PROCH and PROCI are submitted.



5. In the following figure, PROCE completes. Note that PROCF will not execute until PROCC has completed because PROCF can run only on the class 4 session. If at this point any other Process is submitted for class 1, 2, or 3, it can use a class 3 session.



TCQ Status and State Values

The SELECT PROCESS command displays CONNECT:Direct status values, CONNECT:Direct task state values, and VTAM state values.

CONNECT:Direct Status Value

Each Process on the TCQ has an associated CONNECT:Direct status value. This status value has a unique meaning determined by which queue the Process is in. The SELECT PROCESS command displays CONNECT:Direct status values.

CONNECT:Direct Task State Values

When a Process is in the Execution queue, the SELECT PROCESS command also displays a CONNECT:Direct task state value. The state values are provided for information purposes and cannot be modified or controlled with CONNECT:Direct commands.

The task state value shows the current state of the Process. Usually, CONNECT:Direct tasks are waiting for completion of a service such as File I/O, CONNECT:Direct locked resource, or VTAM I/O.

VTAM State Values

When a Process is on the Execution queue, the SELECT PROCESS command may also display a VTAM state value, depending on timing.

The state values are provided for information purposes and cannot be modified or controlled with CONNECT:Direct commands. If the Process is currently in a VTAM I/O state, the SELECT PROCESS output shows a further VTAM state value such as send or receive.

While a Process is executing, the SELECT PROCESS command displays the number of file blocks or records and VTAM request/response units (RUs) sent or received to give you an indication of the status of COPY statements on the Process.

CONNECT:Direct TCQ Values and Commands

The following tables describe each of the CONNECT:Direct TCQ status values , CONNECT:Direct task state values, VTAM state values, and CONNECT:Direct commands applicable to each of the queues.

WAIT QUEUE

This table shows the CONNECT:Direct status value for the Wait Queue.

Status Value	Explanation
WC (Waiting for Connection)	This is the initial queue status when a Process is submitted without HOLD or RETAIN specified. This status means the Process is ready to execute as soon as possible. Process is ready to run but other Processes are executing with the same SNODE, and no other sessions are available. This Process will run as soon as an eligible session is available. If you find a Process in this state, it is most likely for this reason.

This table shows the applicable commands for the Wait Queue.

Command	Description
Change Process	Modifies Process attributes
Delete Process	Removes the Process from the queue
Select Process	Displays Process status and state

EXECUTION QUEUE

This table explains the status values for the Execution Queue.

Status Value	Explanation
EXEC	Node is in Process control and executing the Process shown.
WC (Waiting for Connection)	Process control is in negotiation while the two nodes determine which Process executes next, based on priority.
¬PR.CNTL	This node is not in Process control. This can occur: (1) during Process negotiation where highest priority on either node will run next, or (2) when this is the SNODE during Process execution.
SS	Session with other node is being started.

This table shows the task state values for the Execution Queue.

Task State	Explanation
DISPATCH	Task is waiting to be dispatched
INACTIVE	Task is dispatchable but inactive
VTAM I/O	Task is waiting on VTAM request
P=SNODE	PNODE equals SNODE task
NETEX I/O	Waiting on NETEX I/O request
VSAM I/O	Waiting on VSAM I/O request
MISC	Miscellaneous I/O, such as a WTO
FILE I/O	Non-VSAM I/O
LOCK	Waiting for CONNECT:Direct locked resource
SUBTASK	Waiting on a subtask, such as open or close, allocation, security, or a RUN TASK program
RUNNING	Executing instructions
TCA SCAN	TCA scan
TIMER	Waiting for timer event
LU1PRINT	Waiting for LU1 printer
ATTACH	Waiting for tape drive to be attached
ALLOCATE	Waiting for allocation to complete
MOUNT	Waiting for tape mount
OPEN	Waiting for OPEN to complete
$	Subtask running
+	Waiting on SYNC/4 submit
@	Waiting on STATS open/close
!	Waiting on STimer to be set
#	Waiting on strings
2	Task has subtask running
WTOR	Outstanding WTOR

(continued)

Task State	Explanation
STR WAIT	Waiting on VSAM string
LEV2 SUB	RUN TASK attached and running

This table explains the Subtask state values for the Execution Queue. The first characters of the subtask request state indicate the session protocol such as TCP or LU6.2.

Subtask State	Explanation
ADOPT V2	Performing Adopt
CALL V2	Performing Call
INIT V2	Performing Initialization
CLEANUP	Performing Cleanup
HANGUP	Performing Hangup
SEND V2	Performing Send
SEND RSP	Performing Send Response
SEND SIG	Performing Send Signal
RECV V2	Performing Receive
ANSWER V2	Performing Answer
IOCTL V2	Performing I/O Control
GET BUF	Performing I/O Control Get Buffer
RDY RCV	Performing I/O Control Ready Receive

This table shows the VTAM state values for the Execution Queue.

VTAM State	Explanation
NO SESSION	No VTAM session
SESSION EST	Session with another node being established

(continued)

VTAM State	Explanation
NO REQUEST	No VTAM request outstanding
RECEIVE	Waiting on VTAM RECEIVE request
OPEN	Waiting on VTAM OPEN request
CLOSE	Waiting on VTAM CLOSE request
SETLOGON	Waiting on VTAM SETLOGON request
REQSESS	Waiting on a request session request
OPNDST	Waiting on an open destination request
CLSDST	Waiting on a close destination request
GENCB EXTLST	Waiting on a GENCB EXLIST
GENCB ACB	Waiting on a GENCB ACB
GENCB NIB	Waiting on a GENCB NIB
GENCB RPL	Waiting on a GENCB RPL
REJSESS	Waiting on a reject session request
SESSIONC	Waiting on a session cancel request
INQUIRE	Waiting on an inquire request
OPNSEC	Waiting on an open secondary request
RSHUTD	Waiting on a request shutdown request
SIMLOGON	Waiting on a simulate logon request
SND RESPONSE	Waiting on a send response request
WAIT FOR +DR	Waiting on a definite response request
VTAM I/O	Waiting on VTAM request
SEND	Waiting on VTAM send request
DACTSESS	waiting on LU6.2 deactivate session request
RCVFMH5	Waiting on LU6.2 receive FMH-5 request
REJECT	Waiting on LU6.2 reject conversation request
ACTSESS	Waiting on LU6.2 activate session request
ALLOC ALLOCD	Waiting on LU6.2 allocate conversation until available request

(continued)

VTAM State	Explanation
PRERECV	Waiting on LU6.2 prepare to receive request
CNOS	Waiting on LU6.2 change number of session request
DELLOC	Waiting on LU6.2 deallocate conversation request
SEND CONFRMD	Waiting on LU6.2 send confirmation request
6.2 RECEIVE	Waiting on LU6.2 receive request
6.2 SEND	Waiting on LU6.2 send request
SEND ERROR	Waiting on LU6.2 send error request
DISPLAY	Waiting on LU6.2 display session limit request
DEFINE	Waiting on LU6.2 define session limits request
ALLOC IMMED	Waiting on LU6.2 allocate conversation immediately request

This table shows the allocate state values for the Execution Queue.

Allocate State Value	Explanation
SVC99	Performing SVC99
CAT SEARCH1	CAMLST locate
READ VTOC	CAMLST search

This table describes the applicable command descriptions for the Execution Queue.

Applicable Command	Description
Flush Process	Terminates and deletes an executing Process
Select Process	Displays Process status and state
Suspend Process	Terminates and moves an executing Process on the Hold Queue

HOLD QUEUE

This table explains the status values for the Hold Queue.

Status Value	Explanation
HC (Held for Call)	The Process was submitted with HOLD=CALL specified. A session started from either node will cause this Process to be put on the wait queue in WC status, and eventually EX Q when the first Process finishes.
HE (Held in Error)	The Process was submitted and received an error unrelated to allocation or session errors. The Process was being checkpointed and REQUEUE=YES was specified. A common error that would cause the Process to be in HE status is an out of space condition (Sx37 abend).
HI (Held Initially)	Process was submitted with HOLD=YES specified.
HO (Held by Operator)	An exception response was sent from the other node during FMH exchanges at process negotiation or step termination. It can also occur if an FMH is invalid or is sent out of sync, or if the remote node is not defined in the network map.
HP (Held due to Process Error)	An exception response was sent from the other node during EXIT exchanges at process negotiation or step termination. HP can also occur if an EXIT is invalid, sent out of sync, or if the remote node is not defined in the Network map.
HR (Held Retain)	Process was submitted with RETAIN=YES specified.
HS (Held for Suspension)	The operator issued a SUSPEND PROCess command. The Process can be released later.
RA (Held for Restart Due to Allocation Error)	During Process execution, an allocation error occurred that matched those specified in the initialization parameters. This allows the Process to be restarted after the allocation problem is resolved.
RE (Held while in retry state)	During Process execution, CONNECT:Direct failed to establish a session with the snode. The session with the snode is in the retry state. The number of retries and the interval between retries is specified in the initialization parameters. This Process has been placed on to the HO (Held by Operator) queue by manual intervention before the number of retries was exhausted.
RH (Restart Held)	A checkpointed Process was executing when an error such as a lost session or an I/O error occurred. This allows the copy to be restarted when the session is re-established.
WC (Wait For Connection)	Session establishment was attempted, including retries if specified, and failed. This Process will be put on the wait queue (and later EX Q) if a session with that node is established later. It also can be released.

This table describes the applicable commands for the Hold Queue.

Applicable Command	Description
Change Process	Modifies Process attributes
Delete Process	Removes the Process from the queue
Select Process	Displays Process status and state

TIMER QUEUE

This table shows the status values for the Timer Queue.

Status Value	Explanation
RE (Retry)	The session with the SNODE is in the retry state. The number of retries and interval between retries is specified in the initialization parameters. The Process can be in retry status for session establishment or for an allocation error.
WC (Wait For Connection)	The Process was submitted with a start time or date (STARTT) that has not expired. When the STARTT is reached, the Process will be put on the wait queue for scheduling to the EX Q.

This table shows the applicable commands for the Timer Queue.

Applicable Command	Description
Change Process	Modifies Process attributes
Delete Process	Removes the Process from the queue
Select Process	Displays Process status and state

Process Recovery and Copy Checkpoint/Restart

CONNECT:Direct provides facilities to recover from most errors that occur during Process execution. Recovery from the point of failure is usually accomplished quickly. The following types of errors can occur during normal operation:

- Link failure terminates a session between CONNECT:Direct systems
- File I/O error occurs during Process execution
- CONNECT:Direct abends because of a hardware or other error

CONNECT:Direct offers these facilities to address the errors:

- Session establishment retry
- VTAM automatic session retry
- Process step checkpoint
- COPY statement checkpoint/restart

Initialization parameters, the checkpoint/restart file, and the Copy statement checkpoint parameter are elements of the checkpoint/restart facility.

Session Establishment Retry

When one or more Processes are ready to run with a node, CONNECT:Direct will establish a session with that node and begin execution. If the session cannot be started, CONNECT:Direct will retry the session establishment. The number of retries and the interval between retries are specified by the MAXRETRIES and WTRETRIES initialization parameters.

If a session cannot be established after all retries are exhausted, the Process is placed in the TCQ in the Hold queue with a status of Waiting for Connection (WC). When a session is established with the other node by submitting another Process, by releasing a Process being held, or by contact initiated by the remote node, all other Processes are scanned and the highest priority Process is executed after the previous Process is finished.

VTAM Automatic Session Retry

If Process execution is interrupted because of a VTAM session failure, CONNECT:Direct automatically attempts to restart the session. This recovery facility is similar to and uses the same parameter values as the session establishment retry facility.

If the session cannot be re-established, the Process that was in execution and any other Processes that are ready to run with the other node are placed in the Hold queue with a status of Waiting for Connection (WC).

Process Step Checkpoint

As a Process executes, CONNECT:Direct records which step is executing in the TCQ. If Process execution is interrupted for any reason, the

Process is held in the TCQ. When the Process is available for execution again, CONNECT:Direct automatically begins execution at that step.

COPY Statement Checkpoint/Restart

For physical sequential files, VSAM, and partitioned data sets, CONNECT:Direct collects positioning checkpoint information at specified intervals as a COPY statement executes. If the copying procedure is interrupted for any reason, it can be restarted at the last checkpoint position.

The COPY statement checkpoint/restart works in conjunction with step restart. The restart is automatic if CONNECT:Direct can re-establish a session based on the initialization parameter values for MAXRETRIES and WTRETRIES.

The CHANGE PROCESS command can also invoke the checkpoint/restart facility. See the *CONNECT:Direct for VM/ESA User's Guide* for instructions on how to use the CHANGE PROCESS command.

Note: Checkpoint/restart is not supported for I/O exits at this time.

Initialization Parameters

CONNECT:Direct provides four initialization parameters used with the COPY statement checkpoint/restart facility. See the *Initialization Parameters* appendix beginning on page A-1 for detailed information about these parameters.

- CKPT.MODE specifies whether checkpointing will be performed when CONNECT:Direct is transferring a file in record mode or block mode. (Record mode transfer is used when reblocking of the output file is specified.) It also specifies whether checkpointing will be performed for a set of CMS files. Automatic checkpointing of VSAM files is also supported.

- CKPT specifies the default interval for checkpointing when it is not specified on the COPY statement. Be aware that there is additional overhead associated with specifying too small a checkpoint interval, particularly when transferring large files.

- CKPT.DAYS specifies the amount of time that checkpoint records should be kept if they are not deleted. (Checkpoint records are automatically deleted when a Process is restarted and runs to a successful completion.)

- CKPTDSN specifies the name of the checkpoint/restart file that holds checkpoint records during execution of the COPY statement.

Checkpoint File

The CONNECT:Direct checkpoint/restart file contains positioning information for both files involved in executing a COPY statement. CONNECT:Direct maintains the checkpoint records throughout data transmission and deletes them when a transmission completes successfully. Note that the checkpoints are taken on the receiving end of a transfer. During restart, this information is exchanged with the sender so that proper positioning can take place.

A checkpoint record can be left in the Checkpoint file if an interrupted Process is deleted by the operator. CONNECT:Direct scans the checkpoint records during initialization and deletes records older than the value specified in the CKPT.DAYS initialization parameter.

COPY Statement Parameters

The CKPT and REQUEUE parameters of the COPY statement also control aspects of the Checkpoint/Restart facility.

- You can use the CKPT parameter on the COPY statement to specify the interval CONNECT:Direct uses to record checkpoint information. CONNECT:Direct uses this CKPT value, rounded to the nearest block, in determining how many bytes to transfer before taking a checkpoint. If you do not specify the Copy statement CKPT parameter, CONNECT:Direct uses the value specified in the CKPT initialization parameter. Specifying a value of CKPT=0K, or not specifying CKPT in the initialization parameters, disables checkpointing. Be aware that there is additional overhead associated with specifying too small a checkpoint interval, particularly when transferring large files.

- You can use the REQUEUE parameter on the COPY statement to indicate whether CONNECT:Direct requeues Processes that end due to an abend or allows any subsequent steps to run to process termination. This parameter is only effective if checkpointing is in use. Refer to the *CONNECT:Direct for VM/ESA COPY Statement* chapter of the *CONNECT:Direct Process Guide* for a complete description of this parameter.

Additional Information Sent with Checkpoint/Restart for Non-PDS Files

Except for transfers using NNV2, when Checkpoint/Restart is in use, CONNECT:Direct will send an additional seven bytes per block or ten bytes per record of overhead during data transfer. Data is sent in record

mode when reblocking is taking place during transfer (source and destination block sizes differ). When no reblocking is taking place, CONNECT:Direct sends in block mode.

In the case of transfers using NNV2, CONNECT:Direct sends approximately 30 bytes of overhead after sending an amount of data equal to the checkpoint interval in effect.

Checkpoint/Restart Examples for Non NNV2 Transfers

When checkpointing is in effect, positioning information is transferred in addition to the file. This information is used to reposition the file in the event of an interruption. The following examples show how to determine this overhead in number of bytes when using Checkpoint/Restart for non NNV2 transfers.

In the following table, CONNECT:Direct will checkpoint the file in block mode and add 7 bytes to each block transmitted. This will add 2,100 bytes to the transmission, or less than .04 percent overall.

File Attributes	Sending File	Receiving File
BLKSIZE	19,069	19,069
LRECL	0	0
DSORG	PS	PS
RECFM	U	U
FILE SIZE	300 blocks	300 blocks

In the following table, CONNECT:Direct will checkpoint the file in record mode because the data is being reblocked. CONNECT:Direct will add 10 bytes to each record being transmitted. This will add 150,000 bytes to the transmission, or 5 percent overall.

File Attributes	Sending File	Receiving File
BLKSIZE	5,000	2,000
LRECL	200	200
DSORG	PS	PS
RECFM	FB	FB
FILE SIZE	600 blocks	1,500 blocks

Checkpoint/Restart Example for NNV2 Transfers

Using the transfer described in the preceding table, the file size is 3 million bytes. If the checkpoint interval is 10k, 30 bytes of overhead is sent 300 times, resulting in a total overhead of 9000 bytes. This is about 0.3 percent overhead overall.

Process Recovery and RUN TASK Checkpoint/Restart

CONNECT:Direct for VM/ESA offers a checkpoint/restart feature with the RUN TASK Process statement.

If a RUN TASK program is executing on the SNODE and a session failure occurs, the PNODE recognizes the session failure and puts the Process in the Timer queue for retry. The SNODE, however, is still running the RUN TASK program and is not aware of the session failure until the program finishes. The checkpoint/restart feature for RUN TASK ensures that when the Process restarts on the PNODE, the RUN TASK program will not execute a second time on the SNODE.

When and Where Checkpoint Records Are Written

CONNECT:Direct always writes a checkpoint record on the node where the RUN TASK program executes. The initial checkpoint record is written upon entry to DMGRUNT, the module that handles the RUN TASK Process statement. CONNECT:Direct updates the checkpoint record before attaching the program that is to execute. When the RUN TASK program finishes, CONNECT:Direct updates the checkpoint record again.

Restart Procedure

If a RUN TASK Process step restarts, the node where the program executes attempts to find the checkpoint record in the checkpoint file. If the RUN TASK step is still executing, the Process that is running for the restart of the step waits for the RUN TASK program to finish the first task and then proceed to the next step of the Process, if there is one.

At Process restart for a RUN TASK step, if the program is still executing, you will see two Processes in the EX queue for the same Process step. The first Process is executing the program. The second Process is waiting for the first Process to complete. When the first Process completes, it determines that the session it was running under has been lost and will post the second Process. The second Process records how the RUN TASK step that was still executing ended and proceeds to the next step in the Process.

Determining Reexecution of the RUN TASK Step

The RUN TASK step will not execute again if it is determined at restart that the RUN TASK step ended because it finished before the PNODE restarted the Process. However, if the RUN TASK program did not complete and is not currently running, then the RESTART parameter determines the restart of the Process.

Also, if at restart, CONNECT:Direct cannot find the checkpoint record and the RUN TASK program is not executing, CONNECT:Direct is unable to determine what action to take for the restart. If CONNECT:Direct cannot determine what action to take for the restart, it uses the RESTART parameter.

You can code the RESTART parameter on the RUN TASK step or in the initialization parameters.

- Refer to the *Initialization Parameters* appendix for information on the RUNTASK.RESTART initialization parameter.

- If you code the RESTART parameter on the RUN TASK step, it overrides the initialization parameter. When you code RESTART=YES, CONNECT:Direct executes the program again. When you code RESTART=NO, the Process skips the RUN TASK step.

Chapter 8

Using CONNECT:Direct Exits

This chapter includes information about these CONNECT:Direct exits:

- Statistics exits
- Submit exits
- Allocation exits
- I/O exits

General Information About Exits

This section provides general information about CONNECT:Direct exits.

Avoiding Out-of-Storage Abends

To avoid out-of-storage abends in CONNECT:Direct for VM/ESA, examine all user exits to verify that all obtained storage is freed. For each GETMAIN that an exit issues, it must issue a corresponding FREEMAIN to avoid accumulating storage; also, if an exit opens a file, a FREEPOOL may need to be issued after the file is closed.

Using Exits in 31-bit Addressing Environments

Observe the following requirements or restrictions for exits in 31-bit addressing environments:

- The module should be link-edited with AMODE 31 and be capable of executing in 31-bit mode, unless ALLOC.STORAGE=BELOW is specified in the CONNECT:Direct initialization parameters. This is

because information passed to the exit, including in some cases, the CONNECT:Direct-provided register save area, is located above 16 megabytes. Refer to the section in this chapter describing the particular exit to see if this applies.

- CONNECT:Direct honors the addressing mode (AMODE) and residence mode (RMODE) attributes of user exits. The exit modules are loaded based on the RMODE specification and given control in the addressing mode specified in the AMODE attribute. Exits that are to run above 16 megabytes in 31-bit mode should be linked to AMODE=31, RMODE=ANY.

- Verify that your exits are coded to receive control and execute in the AMODE with which they are linked.

- Exits should return control to CONNECT:Direct in the AMODE that was in effect when CONNECT:Direct invoked the exit. CONNECT:Direct will call your exit through Branch and Save and Set Mode (BASSM), and you should return to CONNECT:Direct through Branch and Set Mode (BSM).

Note: For security exit links, you may need to provide access to a load library containing the modules for the security system in use.

Statistics Exit

CONNECT:Direct generates and logs extensive statistics to an online journal. These records are available in both formatted and unformatted form by means of the Select Statistics command. CONNECT:Direct also provides a statistics exit that gives a user-written program access to the statistics records as they are generated. Use this exit to customize and output these records to a user-defined journal.

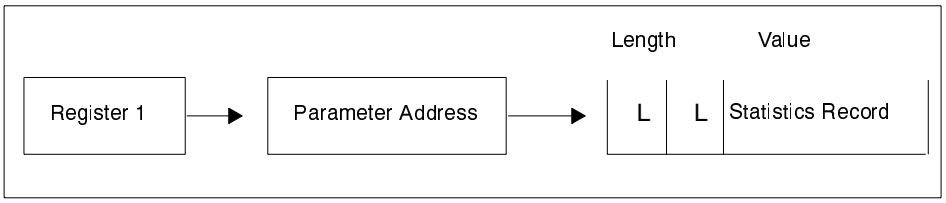
The statistics exit runs as a subtask in the CONNECT:Direct DTF virtual machine. If a user-defined journal is required, you must add the necessary data definitions to the CONNECT:Direct start-up EXEC. Activate the statistics exit by specifying STATISTICS.EXIT=modname in the CONNECT:Direct initialization parameters.

CONNECT:Direct provides a sample exit program called DMCXSTAT. You can use DMCXSTAT as a statistics exit to write VM accounting records, if properly authorized.

Note: The Statistics Exit must be re-entrant and reusable.

Parameter Convention of the Statistics Exit

This figure shows the convention for passing control to the user-written statistics program. Register 1 points to the parameter address that contains the address of the statistics record or the SQCB or both.



Statistics Records

Control Block Maps for the statistics records shown in the following table are in the CONNECT:Direct for VM/ESA macro library (for example, CDV3200 MACLIB for CONNECT:Direct for VM/ESA 3.2.00). Information is passed to the statistics exit program through any of these control blocks. Users are responsible for determining how the control block information will be used in their programs.

CONNECT:Direct for VM/ESA Statistics Records

This table lists the statistics record types, their corresponding record type identifiers, and the name of the assembler macro in THE cdv3200 maclib that generates the DSECT describing the record contents.

Identifier	Description	Macro
PS	Process submit	DMPSSR
CT	Copy termination	DMCTR
MC	PDS member copy	DMFMCR
RT	Run Task	DMRTTR
RJ	Run Job	DMRJTR
SW	Submit within a Process	DMPSSR
WO	WTO	DMFWTOST
PT	Process termination	DMPTR
SD	Start CONNECT:Direct	DMSDCR
ST	Stop CONNECT:Direct	DMSTDCR

(continued)

Identifier	Description	Macro
SI	Signon	DMSFR
SO	Signoff	DMSFR
SP	Select Process	DMDTR
DT	Select Task	DMDTR
SS	Select Statistics	DMDTR
SN	Select Netmap	DMDTR
IA	Inquire Statistics	DMDTR
ID	Inquire STATDIR	DMDTR
IP	Inquire Initparms	DMDTR
FP	Flush Process	DMFPTR
FS	Suspend Process	DMFPTR
CH	Change Process	DMCPTR
DP	Delete Process	DMDPTR
FT	Flush Task	DMDTR
TS	Suspend Task	DMFPTR
IU	Insert User	DMAER
SU	Select User	DMAER
DU	Delete User	DMAER
UU	Update USer	DMAER
UM	Update Netmap	DMAER
XO	Trace on/off	DMXOR
SF	Statistics format	DMFSFREC
IF	Process modal – IF statement	DMMODAL
GO	Process modal – GOTO, ELSE, or EXIT statement	DMMODAL
NL	Process modal – EIF or PEND statement	DMMODAL
CS	Statistics command	DMFSCMDR

(continued)

Identifier	Description	Macro
SC	Statistics control record	DMFSCR
S2	Statistics logging statistics	DMFS2R

The following table lists the statistics records control block maps.

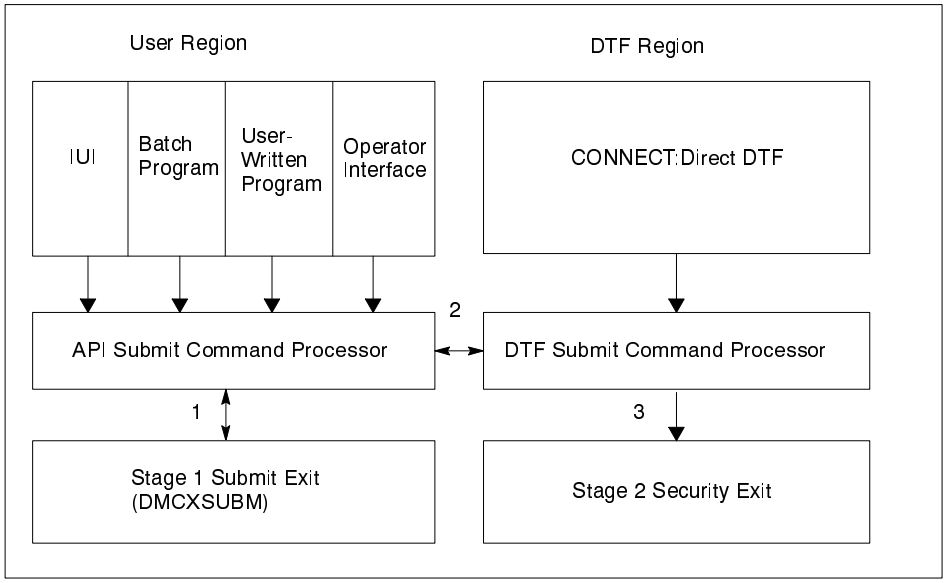
Macro Name	Description
DMAER	Authorization event statistics record
DMCPTR	Change Process statistics record
DMCTR	Copy termination statistics record
DMDPTR	Delete Process statistics record
DMFPTR	Flush and Suspend Process statistics record
DMLSR	Log swap statistics record
DMPSSR	Submit Process statistics record
DMRJTR	Run Job termination record
DMRTTR	Run Task termination record
DMSFR	Signon/Signoff statistics record
DMSTDCR	Stop CONNECT:Direct statistics record
DMXOR	TRACE on/off statistics record
DMFWTOST	WTO statistics record
DMFMCR	PDS member copy record
DMDTR	Display termination record
DMPTR	Process termination record
DMSDCR	Start CONNECT:Direct statistics record
DMTCQGT	GOTO statement
DMTCQIF	IF statement
DMTCQNUL	NULL statement
DMTCQSH	Statement header

(continued)

Macro Name	Description
DMFRJXCB	Run Job exit
DMWRPST	Statistics File wrap record
DMFREPRT	InterCONNECT report record containing text line from SYSPRINT (record type is RE)
DMFREPRT	InterCONNECT log file records produced if LOG=YES is specified for ADD and EXTRACT operations (record type is LF)
DMFROEVT	InterCONNECT report event record containing one record per report written (record type is RO)

Submit Exit

The Submit exit provides an interface to a user-written program when a CONNECT:Direct Process is submitted. With this interface, the user program can change Process information, such as Process name, priority, class, and secondary node, as well as copy step information such as data set name. The following figure shows the execution order of the CONNECT:Direct Submit command.



Submit Exit Processing Flow

The processing flow for the Submit exit is:

1. When a CONNECT:Direct Submit command or Submit statement is issued, the API Submit command processor calls the Stage 1 Submit exit.

2. If the submit is successful, the API Submit command processor calls the DTF Submit command processor.

3. The DTF Submit command processor calls the Stage 2 Submit exit.

CONNECT:Direct provides a sample submit exit in ASSEMBLE, called SUBMEXIT, which you can use as a model for either the Stage 1 (DMCXSUBM) or Stage 2 (SUBMIT.EXIT = modname) exit.

Note: In most cases, it is sufficient to run only the Stage 1 Submit exit.

Implementing the Submit Exit

This section presents information about the following topics related to the Submit exit:

- Interface specifications
- Control block format
- TCQ Statement Header (TCQSH)
- Copy Control Block
- Run Job Control Block
- Run Task Control Block
- Submit Control Block
- TCQE for If statements
- Transmission Control Queue Element (TCQE) Control Block (Process statement information)

Stage 1 Submit Exit

This control point executes in the API address space when a Submit command is processed, and in the DTF address space when a SUBMIT statement is encountered in a process.

- The CONNECT:Direct Stage 1 Submit exit is implemented as an executable load module.

- The name of the load module must be DMCXSUBM.

- The module must be link-edited as NORENT and NOREUS.

- The SUBMIT exits are made available to the CONNECT:Direct DTF GCS machine through the GLOBAL LOADLIB statement.
- For the ISPF IUI, the module should come from a library in the ISPLLIB.

Stage 2 Submit Exit

This control point executes in the DTF address space when a Submit command or a Submit statement is encountered.

- The Stage 2 Submit exit is implemented as an executable load module.
- The name of the load module is user-definable but must not conflict with any CONNECT:Direct load modules.
- Activation of the Stage 2 Submit exit is achieved by specifying SUBMIT.EXIT= (modname) in the CONNECT:Direct initialization parameters.
- The module must be link-edited re-entrant and placed in a load library that can be accessed by the CONNECT:Direct DTF.

Control Block Format

Because the Submit exits are invoked before the Process is actually submitted, some control block fields will not be filled in yet. This section presents control blocks used with all Process statements.

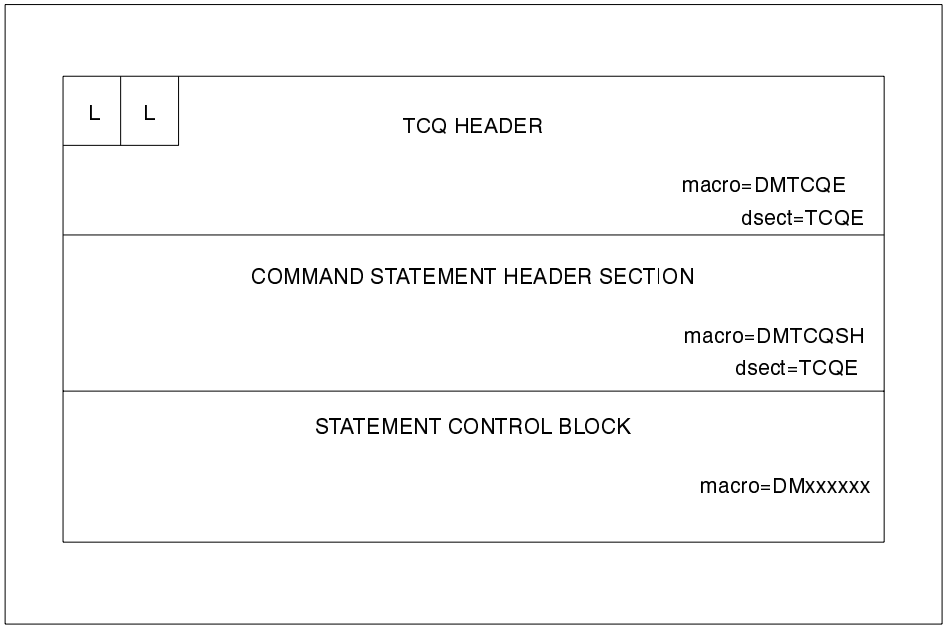
Upon entry to the Stage 1 submit exit, Register 1 points to a fullword containing the address of Header (DMTCQE). Chained off the TCQE is the Command Statement Header (DMTCQSH) followed by the statement control block (Copy, Run Job, Run Task, or Submit). At entry to the Stage 2 submit exit, Register 1 contains the address of a PLIST. If you specified the SECURITY.EXIT=(modname,ALL) initialization parameter, the PLIST contains the address of the TCQE and the SQCB. Otherwise, the PLIST contains only the address of the TCQE.

If Process submission is to be rejected, a nonzero value must be returned in Register 15. The exit must also set that value into the return code field (TQRTNCD) of the TCQE as well as setting a message ID in the TQMSGID field in the TCQE.

The following figure shows the layout of the TCQE. Note that DMxxxxxx represents the macro name for the statement (Copy, Run Job, Run Task, Submit, and so on) in the Process.

Displacement values found in the TCQE and the TCQSH are from the top of the TCQE. Displacement values found in the statement control

blocks are from the top of the TCQSH associated with that statement control block.

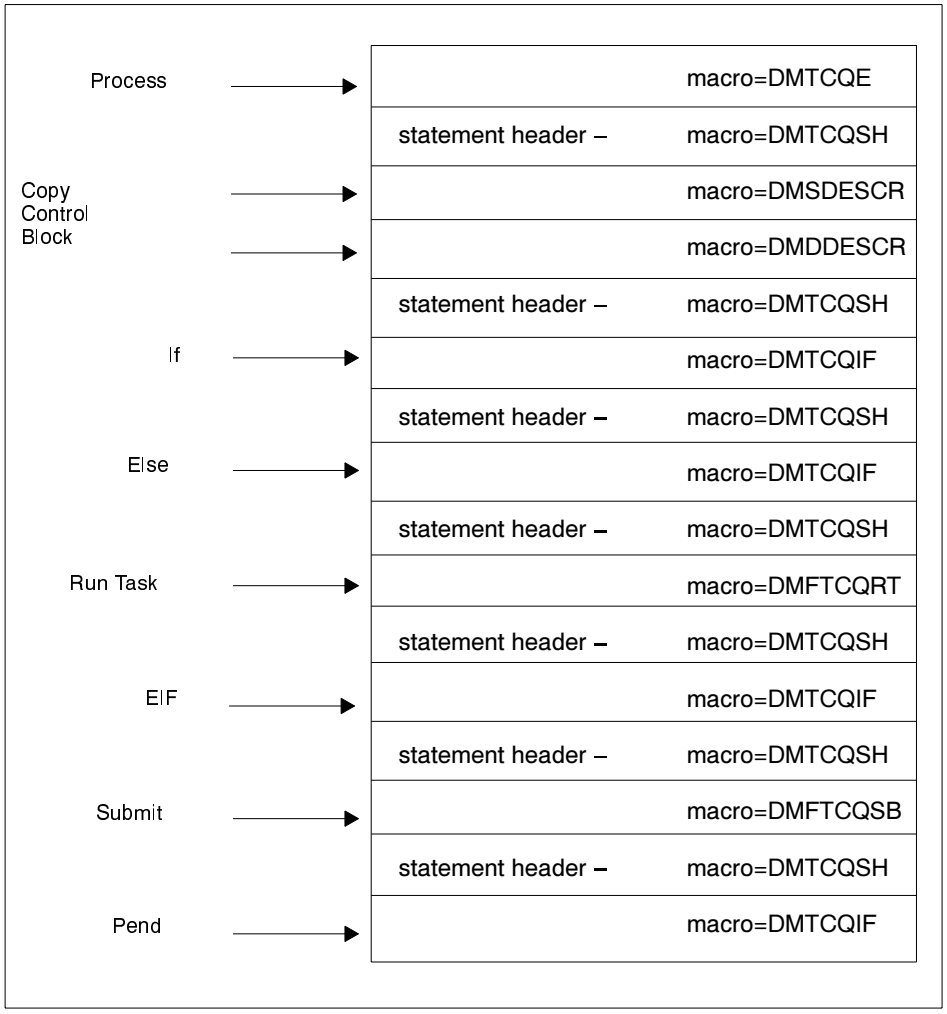


This example section shows how a Process is submitted and the control block that is created when the Submit exit is invoked.

The following Process called TEST01 is submitted.

```
TEST01       PROCESS     SNODE=THERE
STEP01       COPY  FROM  (DSN='FILE1 ASSEMBLE'              -
                           LINK=(VMID1,PASS1,RR,193))       -
                    TO     (DSN='FILE2 ASSEMBLE'            -
                           LINK=(VMID2,PASS2,W,125))
IF01         IF            (STEP01=0)  THEN
STEP02       RUN TASK       (PGM=RTEXAMPL,                  -
                           PARM=(CL44'FILE1 ASSEMBLE'))     -
                            PNODE
             EIF
STEP03       SUBMIT DSN='PROC1 NDMLIB' HOLD=Y
```

The following figure shows the resulting layout of the Process control block after submitting the Process named TEST01.



Modifiable TCQE Fields

The following list describes TCQE fields that you can examine or modify using the Submit exit.

TCQE Field	Content
TQCBHLNG	Contains the length of the entire TCQE. This length added to the address of the TCQE gives the address of the TCQSH.
TQSTMTN	Contains the number of statements in this Process.
TQUNODE	Contains the symbolic node name for the submitter of this Process.
TQUID	Contains the userid for the submitter of this Process.
TQUPAS	Contains the password for the submitter of this Process.
TQPUID	Contains the security userid at the primary node.

(continued)

TCQE Field	Content
TQOPPAS	Contains the old security password at the primary node.
TQNPPAS	Contains the new security password at the primary node.
TQSUID	Contains the security userid at the secondary node.
TQOSPAS	Contains the old security password at the secondary node.
TQNSPAS	Contains the new security password at the secondary node.
TQRTNCD	Contains the Process completion code. The user exit should change this when an error is encountered in the exit or if the Process is no longer to be submitted upon return from the exit.
TQMSGID	Contains the Process message ID. The user exit should enter a message ID related to any return codes set in the exit.
TQCSPRD	Contains the displacement to the first Process statement from the TCQE. This length added to the address of the TCQE gives the address of the TCQSH.
TQPRSBYT	Contains parallel session class. See the following section for details.
TQPRSBIT	Contains parallel session class. See the following section for details.
TQPROCNM	Contains the name of the Process being submitted.
TQSCHDTE	Contains the Julian date the Process is scheduled to be submitted.
TQSCHTME	Contains the time of day the Process is scheduled to be submitted.
TQSCHDAY	Contains the day of the week that the Process is scheduled to be submitted.
TQPRTY	Contains the priority for Process selection.
TQRETAIN	Contains the retain status for the Process.
TQSELDTE	Contains the Julian date a retained Process is to be submitted.
TQTODFLG	Contains the following interval control flags: – If TQTODTD is on, a Process has a scheduled time and date it is to be submitted. – If TQTOTME is on, a Process has a scheduled time it is to be submitted. – If TQTODDAY is on, a Process has a scheduled day of the week it is to be submitted. – If TQTODINT is on, a Process is scheduled to run when a specified interval expires.
TQPNODE	Contains the symbolic node ID of the primary node.
TQSNODE	Contains the symbolic node ID of the secondary node.
TQSTATUS	Contains the Process status.

Conversion of Parallel Session Values

The session class value is stored in two bytes in the TCQE (TQPRSBYT and TQPRSBIT). The class that was specified can be derived from these values. The following table shows a sample of the two bytes for the first 16 classes (maximum can be 256).

TQPRSBYT	TQPRSBIT	CLASS
00	80	1
00	40	2
00	20	3
00	10	4
00	08	5
00	04	6
00	02	7
00	01	8
01	80	9
01	40	10
01	20	11
01	10	12
01	08	13
01	04	14
01	02	15

Allocation Exit

The CONNECT:Direct Allocation exit provides an interface to a user-written program. If you supply a user exit in the initialization parameters, CONNECT:Direct will invoke the exit prior to any allocation activity by the receiving CONNECT:Direct. Through the exit you can change information that CONNECT:Direct will use during the allocation process. You can examine or modify information such as data set name (DSN) and type record name or set fields to terminate the Copy step prior to allocation.

CONNECT:Direct provides a sample Allocation exit, DMGALOEX, located in the ASSEMBLE distribution file. DMGALOEX demonstrates how to access the VSAMPL and the TCQE, and both the source and destination description in the TCQE. It shows how to change a value in the Data Set Description Control Blocks (DMDDESCR or DMSDESCR) and set a return code and message ID before return.

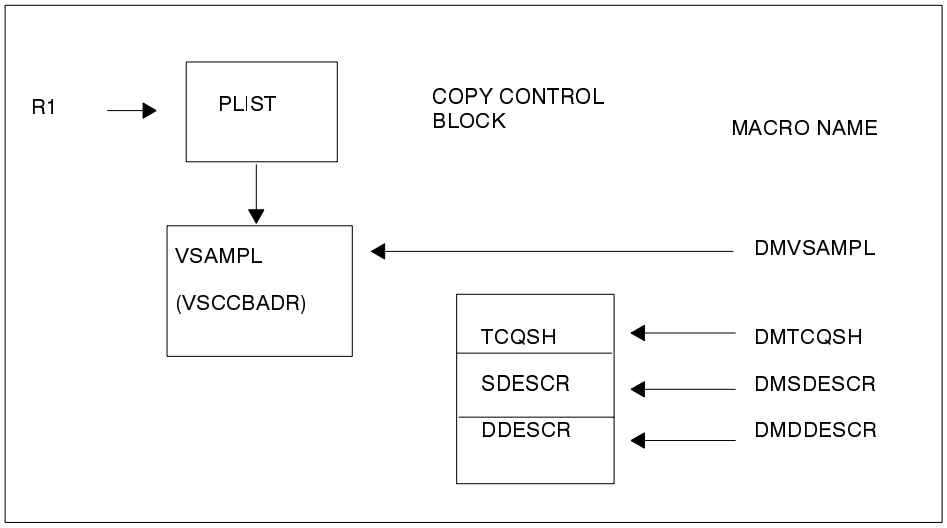
How the Allocation Exit Executes

The Allocation exit executes in the DTF address space when the following conditions exist:

- The Allocation exit is specified in the initialization parameters.
- A file is being received, and the PROCESS step that has initiated the Copy is not in restart mode.

The name of the allocation exit load module is user-definable, but it must not conflict with the name of any other CONNECT:Direct load modules.

To activate the exit, specify ALLOCATION.EXIT=modname in the CONNECT:Direct initialization parameter file. The Allocation exit module must be link-edited as re-entrant and placed in a load library that can be accessed by the CONNECT:Direct DTF. The following figure shows how the parameter list for the Allocation exit is structured.



The following is a list of the Allocation exit parameters:

Parameter	Explanation
R1	(Register 1) Contains the address of a standard parameter list upon entry into the user-written allocation exit.
PLIST	Stands for standard parameter list.
VSAMPL	Stands for VSAM parameter list, whose address is the first fullword in the PLIST.
VSCCBADR	The address of the Process step header and is contained in VSAMPL.
TCQSH	The process step header portion of the Copy control block. Each step of a process generates a TCQSH.

(continued)

Parameter	Explanation
SDESCR	The source data set-descriptor portion of the Copy control block.
DDESCR	The destination data set-descriptor portion of the Copy Control Block. A sample of the DDESCR Control Block is included in this section.
DMVSAMPL	The macro that defines the VSAMPL control block.
DMTCQSH	The macro that defines the TCQSH portion of the Copy control block.
DMSDESCR	The macro that defines the SDESCR portion of the Copy control block.
DMDDESCR	The macro that defines the DDESCR portion of the Copy control block.

Calculating Addresses and Values

Upon entry into the user-written Allocation exit, Register 1 contains the address of a standard parameter list. The first and only entry in the PLIST contains the address of the VSAMPL. The VSCCBADR field in the VSAMPL contains the address of the process step header, TCQSH. The SDESCR and DDESCR following the TCQSH can be found by adding displacements to the TCQSH address.

- To calculate the location of SDESCR, add the length of TCQSH (TSHCBHLN) to the TCQSH address (VSCCBADR).
- To calculate the location of DDESCR, add the length of SDESCR (S1SVSLNG) to the SDESCR address.

All displacement values in the Copy control block are referenced from the beginning of the TCQSH control block.

SDESCR and DDESCR contain both fixed-length fields and offsets to variable-length fields. Any fixed-length field in DDESCR may be modified by the Allocation exit.

The fields that are referenced in SDESCR and DDESCR using displacement values are variable in length and should *not* be modified by the Allocation exit. The exception is the data set name (DSN) field.

The DSN field is created with enough space to hold a 100-character name only if the TSEXPDSN bit is on (set) in the TCQSH. If the TSEXPDSN bit is off in the TCQSH, then the Copy control block does not contain the room to expand the DSN. This means that this copy originated from a CONNECT:Direct node that did not build the Copy control block with an expandable DSN field.

Note: In addition to CONNECT:Direct for VM 4.1.5 and later, other CONNECT:Direct products that build the Copy control block with the TSEXPDSN bit on are CONNECT:Direct for MVS 4.0.6 PUT 8805 and later, and CONNECT:Direct for MS-DOS 4.1.4 and later. CONNECT:Direct for VMS and CONNECT:Direct for VSE releases do not currently build the Copy control block with the TSEXPDSN bit on.

The DSN field can be found by adding D1DDSN to the address of the TCQSH. The DSN field contains a two-byte length field followed by the DSN. Even though the field may be up to 100 bytes long, the two-byte length field contains the actual length of the DSN. If the length of the DSN is changed, the two-byte length field must be modified. The other variable-length fields are created with their current values and **cannot** be lengthened. Do not modify the D1DDSN field.

When allocating the destination file, CONNECT:Direct first uses values from DDESCR, as specified in the CONNECT:Direct Copy statement. Any values needed but not set in DDESCR are taken from the Type record, if one was specified. Any remaining values are taken from the SDESCR portion of the Copy control block.

Note: If the D1DTYPE field is modified by the Allocation exit, the exit must clear any fields in the DDESCR portion of the Copy control block that would override the corresponding Type fields from the Type record.

For example, to use the value for LRECL in the Type record, the D1LRECLN bit in the D1FLAG byte in DDESCR must be set to zero (off). Details on DDESCR fields that override the corresponding type fields from the Type record are shown in the *DDESCR Control Block Format* section, and described in the *DDESCR Modifiable Field*s section of this chapter.

Copy Control Block Definitions

The following Copy control block definitions can be generated in the allocation exit program by including the macro name followed by DSECT=YES, as shown:

- DMVSAMPL DSECT=YES
- DMTCQSH DSECT=YES
- DMDDESCR DSECT=YES
- DMSDESCR DSECT=YES

Copy Control Block Modifications

You should modify only the VSAMPL and DDESCR control blocks. For the VSAMPL control block, you are required to modify the VSRTNCD and VSMSGID fields. The following describes how to make these modifications:

- The VSRTNCD field (and R15) must be set to 0 to allow the Copy step to execute.
- The VSRTNCD field (and R15) must be set to a nonzero value to fail the Copy step.
- The user must insert a message ID into the VSMSGID field if the VSRTNCD is set to a nonzero value. Precaution should be taken to avoid duplicating existing message IDs.
- The user should also place message text corresponding to these message IDs in the CONNECT:Direct message file.

The next section describes the DDESCR control block format and how to modify it.

DDESCR Control Block Format

The listing shows the DDESCR control block format. Information presented in bold in the control block is modifiable. The modifiable fields are described immediately following this figure.

Note: Turning a flag *on* means setting the bit in the byte where the flag is located to 1. Turning a flag *off* means setting the bit in the byte where the flag is located to a 0.

```
DDESCR   DS    0F         /* DESTINATION DATA SET DESCRIPTION      */
*-------------------------------------------------------------------*
*  TYPE: DATA SET DSECT                                             *
*-------------------------------------------------------------------*
D1DVSLNG  DC  Y(DDESCR#) /* LENGTH OF DESTINATION DATA SET VARIABLE */
D1DDESCR  DC   CL8'DDESCR'  VARIABLE SECTION NAME "DDESCR"         */
 BOOLEAN D1FLAG1,(D1FIL1,D1FIL2,D1FIL3,D1TYPE,D1MEMNAM,             -
          D1QUALFR,D1DSN,D1PWD)
*D1FLAG1  DC   XL1'00'    /* FLAGS FOR PARM PRESENT                */
*D1FIL1   EQU  X'80'      /* FILTER 1 SPECIFIED                    */
*D1FIL2   EQU  X'40'      /* FILTER 2 SPECIFIED                    */
*D1FIL3   EQU  X'20'      /* FILTER 3 SPECIFIED                    */
*D1TYPE   EQU  X'10'      /* DATA TYPE SPECIFIED                   */
*D1MEMNAM EQU  X'08'      /* MEMBER NAME SPECIFIED                 */
*D1QUALFR EQU  X'04'      /* DATA SET QUALIFIER SPECIFIED          */
*D1DSN    EQU  X'02'      /* DATA SET NAME SPECIFIED               */
*D1PWD    EQU  X'01'      /* DATA SET NAME PASSWORD                */
 BOOLEAN  D1FLAG,(D1BLKSZE,D1BUFNUM,D1DENSTY,DIDSORGN,              -
          D1KEYLN,D1LIMCNT,D1LRECLN,D1OPTCDE)
*D1FLAG2  DC   XL1'00'    /* FLAG FOR DCB PARMS                    */
*D1BLKSZE EQ   X'80'      /* BLOCK SIZE SPECIFIED                  */
*D1BUFNUM EQ   X'40'      /* NUMBER OF BUFFERS SPECIFIED           */
*D1DENSTY EQ   X'20'      /* DENSITY SPECIFIED                     */
*D1DSORGN EQ   X'10'      /* DATA SET ORGANIZATION                 */
*D1KEYLN  EQ   X'08'      /* KEYLENGTH SPECIFIED                   */
*D1LIMCNT EQ   X'04'      /* LIMCT SPECIFIED                       */
*D1LRECLN EQ   X'02'      /* LOGICAL RECORD LENGTH SPECIFIED       */
*D1OPTCDE EQ   X'01'      /* OPTION CODE SPECIFIED                 */
 BOOLEAN  D1FLAG3,(D1RECFMT,D1RKYP,D1TRKTCH,D1CYLOFL,D1NTM)
*D1FLAG3  DC   XL1'00'    /* FLAG FOR DCB PARM                     */
*D1RECFMT EQ   X'80'      /* RECORD FORMAT SPECIFIED               */
*D1RKYP   EQ   X'40'      /* RKP SPECIFIED                         */
*D1TRKTCH EQ   X'20'      /* TRTCH SPECIFIED                       */
*D1CYLOFL EQ   X'10'      /* CYL OVERFLOW SPECIFIED                */
*D1NTM    EQ   X'08'      /* NUMBER OF INDEXED TRACKS SPECIFIED    */
*UNUSED   EQ   X'04'      /* UNUSED                                */
*UNUSED   EQ   X'02'      /* UNUSED                                */
*UNUSED   EQ   X'01'      /* UNUSED                                */
 BOOLEAN D1FLAG4,(D1DS,D1LABEL,D1PSSED,D1NOPWRD,D1RETPRD,           -
          D1EXPDTE)
*D1FLAG4  DC   XL1'00'    /* FLAG FOR LABEL PARM                   */
*D1DS     EQ   X'80'      /* DATA SET SEQUENCE NUMBER SPECIFIED    */
*D1LABEL  EQ   X'40'      /* LABEL TYPE SPECIFIED                  */
*D1PSSWD  EQ   X'20'      /* PASSWORD SPECIFIED                    */
*D1NOPWRD EQ   X'10'      /* NO PASSWORD SPECIFIED                 */
*D1RETPRD EQ   X'08'      /* RETENTION PERIOD SPECIFIED            */
*D1EXPDTE EQ   X'04'      /* EXPIRATION DATE SPECIFIED             */
*UNUSED   EQ   X'02'      /* UNUSED                                */
*UNUSED   EQ   X'01'      /* UNUSED                                */
 BOOLEAN  D1FLAG5,(D1BLK,D1TRK,D1CYL,D1REC)
*D1FLAG5  DC   XL1'00'    /* FLAG FOR SPACE PARM                   */
*D1BLK    EQ   X'80'      /* ALLOCATION IN BLOCKS SPECIFIED        */
*D1TRK    EQ   X'40'      /* ALLOCATION IN TRACKS SPECIFIED        */
*D1CYL    EQ   X'20'      /* ALLOCATION IN CYLINDERS SPECIFIED     */
*D1REC    EQ   X'10'      /* ALLOCATION IN RECORDS SPECIFIED       */
*UNUSED   EQ   X'08'      /* UNUSED                                */
*UNUSED   EQ   X'04'      /* UNUSED                                */
*UNUSED   EQ   X'02'      /* UNUSED                                */
*UNUSED   EQ   X'01'      /* UNUSED                                */
```

```
*D1FLAG11 DC  XL1'00'     /* FLAG FOR COMPRESSION                  */
*D1COMP   DC  X'80'       /* COMPRESSION SPECIFIED                 */
*D1YCKPT  DC  X'40'       /* CHECKPOINTING EXPLICITLY SPECIFIED    */
*D1DOCKPT DC  X'20'       /* CHECKPOINTING REQUESTED BY RECEIVER   */
*D1RSTRT  DC  X'10'       /* PROCESS NOT EXEC THE FIRST TIME       */
*D1NOCKPT DC  X'08'       /* CKPT NEVER TAKEN                      */
*D1IODRVR DC  X'04'       /* RECEIVER IS A USER I/O DRIVER   .     */
*D1CKPTPO DC  X'02'       /* I/O DRIVER SAYS DO PDS CHECK POINTING */
*D1NGCKPT DC  X'01'       /* SRC SIDE CKPTING REQUESTED            */
 BOOLEAN D1FLAG12,(D1PDSNOD,D1PDSNGL,D1PDSNOA,D1PDSNOR,D1PDSEXL,     *
               D1PDSSEL,D1NOBLOK,D1SPOOL)
*D1FLAG12 DC   XL1'00'     /* FLAG FOR PDS SUPPORT                  */
*D1PDSNOD EQ   X'80'       /* PDS.DIR = NO SPECIFIED                */
*D1PDSNGL EQ   X'40'       /* SINGLE MEMBER SPECIFIED               */
*D1PDSNOA EQ   X'20'       /* ALIAS=NO SPECIFIED                    */
*D1PDSNOR EQ   X'10'       /* REPLACE = NO SPECIFIED                */
*D1PDSEXL EQ   X'08'       /* EXCLUDE LIST SPECIFIED                */
*D1PDSSEL EQ   X'04'       /* SELECT LIST SPECIFIED                 */
*D1NOBLOK EQ   X'02'       /* CANNOT SUPPORT BLOCK MODE             */
*D1SPOOL  EQ   X'01'       /* SPOOL DATASET                         */
          DC   CL1' '     /* RESERVED FOR COMPRESSION CHARACTERS   */
          DC   X'00'      /* ALIGNMENT                             */
          DC   F'00'      /* RESERVED FOR CHECKPT INTERVAL         */
          DC   F'00'      /* ALIGNMENT                             */
         DC   D'00'       /* RESERVED - # OF BYTES RECEIVED        */
D1EXIT1  DC   CL8' '      /* DATA FILTER EXIT 1                    */
D1EXIT2  DC   CL8' '      /* RESERVED                              */
D1EXIT3  DC   CL8' '      /* RESERVED                              */
D1DTYPE  DC    CL8' '     /* DATA TYPE NDM DEFINED                 */
*                         /* C'BINARY',C'TEXT',                    */
*                         /* C'DF'                                 */
D1DDSTYP DC    CL4' '     /* THE DESTINATION DATA SET TYPE         */
*                         /* (ESDS, KSDS, RRDS, PDS, SAM, LIB)     */
D1DDISP1 DC    C' '       /* SOURCE FILE DISPOSITION STATUS "N"=NEW*/
*                         /* "O" = OLD "S" = SHR "M"= MOD "R" = RPL*/
D1DDISP2 DC    C' '       /* SOURCE FILE DISPOSITION WITH NORMAL   */
*                         /* TERMINATION"D" = DELETE , "K" = KEEP  */
*                         /* "C" = CATALOG                         */
D1DDISP3 DC    C' '       /* SOURCE FILE DISPOSITION WITH ABNORMAL */
*                         /* TERMINATION "D" = DELETE , "K" = KEEP */
*                         /* "C" = CATALOG                         */
         DC    XL1'00'    /* ALIGNMENT                             */
D1BLKSIZ DC    CL5' '     /* BLOCK SIZE                            */
D1BUFNO  DC    CL2' '     /* NUMBER OF I/O BUFFERS                 */
D1DEN    DC    CL1' '     /* TAPE DENSITY                          */
D1DSORG  DC    CL4' '     /* DATA SET ORGANIZATION                 */
D1KEYLEN DC    CL3' '     /* KEY LENGTH                            */
D1LIMCT  DC    CL3' '     /* LIMIT FOR EXTENDED SEARCH             */
D1LRECL  DC    CL5' '     /* LOGICAL RECORD LENGTH                 */
D1OPTCD  DC    CL6' '     /* OPTIONAL SERVICE NUMBER               */
D1RECFM  DC    CL4' '     /* RECORD FORMAT                         */
D1RKP    DC    CL5' '     /* RELATIVE KEY POSITION                 */
D1TRTCH  DC    CL2' '     /* 7-TRACK RECORDING MODE                */
D1CYLO   DC    CL2' '     /* TRKS IN CYL OVERFLOW                  */
D1NTMIN  DC    CL2' '     /* NUMBER OF TRKS FOR CYL INDEX          */
D1DSSEQ  DC    CL5' '     /* DATA SET SEQUENCE NUMBER              */
D1LABTYP DC    CL3' '     /* LABEL TYPE                            */
D1RETPD  DC    CL4' '     /* RETENTION PERIOD                      */
D1EXPDT  DC    CL5' '     /* EXPIRATION DATE YYDDD                 */
D1LOCTYP DC    CL3' '     /* ALLOCATION TYPE - CYL,TRK,BLK,REC     */
D1PRILOC DC    CL8' '     /* PRIMARY ALLOCATION AMOUNT             */
D1SECLOC DC    CL8' '     /* SECONDARY ALLOCATION AMOUNT           */
D1DIRBLK DC    CL5' '     /* NUMBER OF DIRECTORY BLOCKS            */.
D1AVGBLK DC    CL5' '     /* AVERAGE BLOCK SIZE FOR SPACE PARM     */
D1UNIT   DC    CL8' '     /* UNIT TYPE OR GROUP NAME               */
D1UNITCT DC    CL2' '     /* UNIT COUNT                            */
D1VOLSEQ DC    CL4' '     /* VOLUME SEQUENCE NUMBER                */
D1VOLCT  DC    CL4' '     /* VOLUME COUNT                          */
D1SYSCL  DC    CL1' '     /* SYSOUT CLASS                          */
```

```
D1SYSPN  DC    CL8' '      /* SYSOUT PROGRAM NAME                    */
D1SYSCN  DC    CL4' '      /* SYSOUT CODE NAME                       */
D1DESTN  DC    CL8' '      /* DESTINATION OF OUTPUT                  */
D1COPY   DC    CL3' '      /* NUMBER OF SYSOUT COPIES                */
D1PASSWD DC    CL8' '      /* DATA SET PASSWORD                      */
         DC    F'00'       /* RESERVED                               */
D1EXDISP DC    H'00'       /* DISPLACEMENT TO CCB EXPANSION SECTION */
         DC    H'00'       /* RESERVED...USED IN SDESCR FOR CKPT PTR*/
         DC    CL5' '      /* RESERVED FOR AUTH FILE LRECL           */
         DC    X'00'       /* ALIGNMENT                              */
D1DMEMB  DC    H'00'       /* DISPLACEMENT TO MEMBER NAME FIELD      */
D1DMEMB  DC    H'00'       /* DISPLACEMENT TO MEMBER NAME FIELD      */
D1DQUAL  DC    H'00'       /* DISPLACEMENT TO QUALIFIER FIELD        */
D1DDSN   DC    H'00'       /* DISPLACEMENT TO DATA SET NAME FIELD    */
D1DVOLN  DC    H'00'       /* DISPLACEMENT TO VOLUME SERIAL NUMBER   */
D1DCBDSN DC    H'00'       /* DISPLACEMENT TO DCB REFERENCE NAME     */
D1VOLREF DC    H'00'       /* DISPLACEMENT TO THE VOLUME REFERENCE   */
D1DMEMBX DC    H'00'       /* DISPLACEMENT TO THE EXCLUDE LIST       */
         DS    0F          /* CONTROL BLOCKS MUST BE FULLWORD ALIGNED/
DDESCR#  EQU   *-DDESCR    /* SIZE OF THE DDESCR FIXED SECTION       */
 *-------------------------------------------------------------------*
*                                                                    *
*              Control block EXPANSION SECTION                       *
*                                                                    *
*--------------------------------------------------------------------*
D2EXSECT DSECT
D2MSVGP  DC    CL8         /* MSVGP=XXXXXXXX                         */
D2STRK   DS    F           /* STARTING TRACK ADDRESS (FOR VSE)       */
D2#TRK   DS    F           /* NUMBER OF TRACKS TO ALLOCATE (VSE)     */
D2LNKUID DS    H           /* DISPLACEMENT TO LINK USERID (VM)       */
D2LNKPWD DS    H           /* DISPLACEMENT TO LINK PASSWORD (VM)     */
D2LNKAM  DS    CL2         /* LINK ACCESS MODE (VM)                  */
D2LNCUU  DS    CL3         /* LINK CUU (VM)                          */
BOOLEAN  D2FLAG1,(D2IOKWDF,D2DTKWDF,D2SQKWDF,D2DBKWDF,
         D2IOXNAM,D2LNKCU4,D2VCCU4)
*D2FLAG1  DS    XL1         /* MISCELLANEOUS FLAGS                    */
*D2IOKWDF EQU   X'80'       /* - IOEXIT KEYWORD FOUND                 */
*D2DTKWDF EQU   X'40'       /* - DATAEXIT KEYWORD FOUND               */
*D2SQKWDF EQU   X'20'       /* - SQL KEYWORD FOUND                    */
*D2DBKWDF EQU   X'10'       /* - DBPARMS KEYWORD FOUND                */
*D2IOXNAM EQU   X'08'       /* - EXIT NAME IS PRESENT                 */
*D2LNKCU4 EQU   X'04'       /* - 4 CHAR. CCUU PRESENT                 */
*D2VCCU4  EQU   X'02'       /* - 4 CHAR. VSAM CCUU PRESENT            */
D2VCDSN   DS    H           /* DSPL TO VSAMCAT DATASET NAME (VM)      */
D2VCUID   DS    H           /* DISPLACEMENT TO VSAMCAT USERID (VM)    */
D2VCPWD   DS    H           /* DISPLACEMENT TO VSAMCAT PASSWD (VM)    */
D2VCAM    DS    CL2         /* VSAMCAT ACCESS MODE (VM)               */
D2VCCUU   DS    CL3         /* VSAMCAT CUU (VM)                       */
          DS    CL1         /* ALIGNMENT                              */
D2SODSP   DS    H           /* DISPLACEMENT TO SYSOPTS TEXT
D2GROUPD  DS    H           /* DISPLACEMENT TO GROUP FILE NAME
D2IOXCS   DS    H           /* DISPLACEMENT TO IOEXIT STRING          */
D2SQLCS   DS    H           /* DISPLACEMENT TO SQL STRING             */
D2DBPCS   DS    H           /* DISPLACEMENT TO DBPARMS STRING         */
D2EXITN   DS    CL8         /* EXIT NAME                              */
D2LKCCUU  DS    CL4         /* 4 CHAR CCUU FOR LINK                   */
D2VCCCU4  DS    CL4         /* 4 CHAR CCUU FOR VSAMCAT                */
          DS    10H         /* RESERVED
D2GMTOFF  DS    F           /* GMT OFFSET FOR SENDING MACHINE         */
D2EX#     EQU   *-D2EXSECT/* SIZE OF THE EXPANSION SECTION           */
```

Note: Do not modify the displacement fields.

DDESCR Modifiable Fields

The following table lists the modifiable fields in the DDESCR portion of the Copy control block:

Field	Explanation
D1DTYPE	Specifies the entry in the CONNECT:Direct type defaults file.
D1BLKSIZ	Specifies the block size. To use the block size indicated in the destination data set description portion of the Copy control block, set the D1BLKSZE flag on. To use the block size indicated in the type record, set the D1BLKSZE flag off.
D1DEN	Specifies the tape density. To use the tape density indicated in the destination data set portion of the Copy control block, set the D1DENSTY flag on.
D1DSORG	Specifies the data set organization. To use this value, set the D1DSORGN flag on. To use the value indicated in the type record, set the D1DSORGN flag off.
D1LRECL	Specifies the logical record length. To use this value, set the D1LRECLN flag on. To use the value indicated in the type record, set the D1LRECLN flag off.
D1RECFM	Specifies the record format. To use this value, set the D1RECFMT flag on. To use the value indicated in the type record, set the D1RECFMT flag off.
D1RKP	Specifies the relative key position. To use this value, set the D1RKYP flag on.
D1TRTCH	Specifies 7-track recording mode. To use this value, set the D1TRKTCH flag on.
D1LABTYP	Specifies the label type. To use this value, set the D1LABEL flag on.
D1RETPD	Specifies the retention period. To use this value, set the D1RETPRD flag on. To use the value indicated in the D1EXPDT field, set the D1RETPRD flag off.
D1EXPDT	Specifies the retention period. To use this value, set the D1EXPDTE flag on.
D1PRILOC	Specifies the primary allocation amount. All of the following bits must be set off to use the space allocation values specified in the type record: D1TRK, D1CYL, D1BLK
D1SECLOC	Specifies the secondary allocation amount. To use this value, set the D1SECALL flag on. All of the following bits must be set off to use the space allocation values specified in the type record: D1TRK, D1CYL, D1BLK
D1DIRBLK	Specifies the number of directory blocks. To use this value, set the D1DRBLK flag on. All of the following bits must be set off to use the space allocation values specified in the type record: D1TRK, D1CYL, D1BLK
D1UNIT	Specifies the unit type or group name. To use this value, set the D1GRPTYP flag on. To use the value indicated in the type record, set the D1GRPTYP off. All of the following bits must be set off to use the UNIT value specified in the type record: D1UNCNT, D1P, D1DEFER, D1GRPTYP
D1VOLSEQ	Specifies the volume sequence number. To use this value, set the D1VOLSQ flag on.
D1VOLCT	Specifies the volume count. To use this value, set the D1VOLCNT flag on.
D1PASSWD	Specifies the data set password. To use this value, set the D1PWD flag on.

(continued)

Field	Explanation
D1DMEMB	Specifies the displacement to the member name field.*
D1DDSN	Specifies the displacement to the data set name field.*
D1DVOLN	Specifies the displacement to the volume serial number.* All of the following bits must be set off to use the VOLSER value specified in the type record: D1VOLSER, D1PRIV, D1RETAIN, D1VOLSQ, D1VOLCNT, D1VOLRF.
D2MSVGP	Specifies the group of mass storage volumes that reside on a mass storage system (MSS) device. To use this value, set the D1DSN flag on.
D2LNKUID	Specifies the displacement to the LINK user ID. To use this value, set the D1LINK flag on. To use the value indicated in the type record, set the D1LINK flag off.
D2LNKPWD	Specifies the displacement to the LINK password. To use this value, set the D1LINK flag on. To use the value indicated in the type record, set the D1LINK flag off.
D2LNKAM	Specifies the LINK access mode. To use this value, set the D1LINK flag on. To use the value indicated in the type record, set the D1LINK flag off.
D2LNKCUU	Specifies the LINK CUU. To use this value, set the D1LINK flag on. To use the value indicated in the type record, set the D1LINK flag off.
D2VCDSN	Specifies the displacement to the VSAMCAT data set name. To use this value, set the D1VSMCAT flag on. To use the value indicated in the type record, set the D1VSMCAT flag off.
D2VCUID	Specifies the displacement to the VSAMCAT userid. To use this value, set the D1VSMCAT flag on. To use the value indicated in the type record, set the D1VSMCAT flag off.
D2VCPWD	Specifies the displacement to the VSAMCAT password. To use this value, set the D1VSMCAT flag on. To use the value indicated in the type record, set the D1VSMCAT flag off.
D2VCAM	Specifies the displacement to the VSAMCAT access mode. To use this value, set the D1VSMCAT flag on. To use the value indicated in the type record, set the D1VSMCAT flag off.
D2VCCUU	Specifies the VSAMCAT CUU. To use this value, set the D1VSMCAT flag on. To use the value indicated in the type record, set the D1VSMCAT flag off.

*Do not modify the fields D1DMEMB, D1DDSN, and D1DVOLN. These fields represent displacements to their corresponding values. However, after calculating the address of the values (by adding the displacement to the address of the TCQSH) the actual values may be changed by CONNECT:Direct. At the calculated address, you will find a halfword field representing the length of the data that follows. If the length of the data changes, this halfword must also be changed to reflect the new length. If the displacement to one of these fields is 0, do not insert a value or displacement. For example, if D1DMEMB=0, no member name was specified and a member name cannot be inserted. Do not set the D1MEMNAM flag on if D1DMEMB=0.

The same applies to D1DDSN. For D1DVOLN, if the displacement is 0, you cannot modify this field or turn on the D1DVOLSER flag. Also, you cannot add volume serial numbers to this list. You can delete volume serial numbers from the list or change the volume serial

number. If volume serial numbers are deleted, decrement the length field by 6 for each one deleted. If all volume serial numbers are deleted, make D1DVOLN=0 and turn off the D1VOLSER bit.

I/O Exit

The CONNECT:Direct I/O exit provides an interface to user-written programs, allowing them to read and write data to or from a file. This I/O exit allows you modify file attributes and data values as required.

Note: Checkpoint/restart is not supported for I/O exits.

Implementing the I/O Exit

If you plan to use an I/O exit, consider the following items.

- These exits must not alter any CONNECT:Direct control block fields (except in the EXTCB as indicated in the section *I/O Exit Access to Control Blocks* on page 8-24). If other CONNECT:Direct control block fields are altered, the results are unpredictable.
- If an Allocation exit is specified, it will not be given control when the COPY statement contains an IOEXIT keyword.
- Add any message IDs specified by an I/O exit to the CONNECT:Direct Message file. Refer to *Adding Messages to CONNECT:Direct Message Library* on page 9-1 for instructions.
- Return from the I/O exit in the AMODE under which it was called. For example, if the I/O exit is called in 31-bit mode, the return should be in that mode. Therefore, if CONNECT:Direct is running on an XA system, return from an I/O exit through a Branch Set Mode (BSM) instruction rather than a Branch (BR) instruction.

CONNECT:Direct provides you with a sample I/O exit in the member named RDEXIT. RDEXIT sample IOEXIT is AMODE/RMODE 24, NORENT. You need to write I/O exits to satisfy your specific data set requirements.

Note: RDEXIT is a read only example exit. Input file is LREL 80, RECFM F.

After you write the I/O exit, implement it by specifying the exit name on the IOEXIT keyword on a COPY statement or on the INSERT and UPDATE TYPE file commands.

Specifying the I/O Exit on the COPY Statement

Include the IOEXIT keyword on the COPY statement to indicate that an I/O exit is used. A sample of the COPY Process for RDEXIT follows.

```
MVSEXIT  PROCESS SNODE=CSD.VM.CD09 NOTIFY=NAME1
STEP1    RUN TASK (PGM=DMRTDYN,PARM=(                                  -
         C' ALLOC DSN="SAMPLE INITPARM"                                -
         C' DDNAME=INFILE '                                            -
         C' LINK=(NAME1,ALL,RR,0191) '                                 -
         C' DISP=OLD '                                                 -
         C' DCB=DSORG=PS ' ))
STEP2    COPY FROM (IOEXIT=(RDEXIT,CL44'SAMPLE INITPARM'               -
         )) TO (DSN='!SPOOL NAME1 TEST DATA')
STEP3    RUN TASK (PGM=DMRTDYN,PARM=(                                  -
         C' UNALLOC DD=INFILE'  )  )
```

The **exitname** is the name of the user-written program to receive control for I/O-related requests.

Note: The VM/ESA system requires the DMRTDYN program to be used as a pre/post-processing step. In the previous example, DMRTDYN allocates and deallocates a VM file for the COPY Process.

The IOEXIT keyword is valid in either the FROM or TO areas of the COPY statement. This enables you to specify a different user-written I/O exit on each side.

If you specify an exit, it may ignore the values of the other parameters on the COPY statement (the DCB information). This is beyond the control of CONNECT:Direct.

Specifying the I/O Exit in the TYPE File

Another method of specifying that an I/O exit is to include the IOEXIT keyword on the INSERT and UPDATE Type file commands. The format is the same as on the COPY statement. If you specify an IOEXIT parameter on the COPY statement, it overrides any IOEXIT specified in the Type file entry. The type defaults record must reside on the side (source or destination) of the copy on which it was referenced.

I/O Exit Access to Control Blocks

On entry to the exit, register 1 (R1) contains the address of the pointer to the EXTCB (Exit Control Block). As with other user exits, the parameter list addresses point to a two-byte length followed by the value. The macro DMEXITCB generates the EXTCB. DMEXITCB is supplied in the CDV3200 MACLIB.

I/O Exit Requests

The I/O exit is called with CONNECT:Direct requests that are found in EXTOPER, which is a field in EXTCB. The following are the requests that the input and output I/O exits receive.

BEGIN Request

CONNECT:Direct makes a BEGIN request to an I/O exit when it begins communication with the exit. This is when the exit should allocate work areas in preparation for future requests. This is the first request an I/O exit receives.

OPEN Request

CONNECT:Direct makes an OPEN request to an I/O exit when the exit should allocate and open the file. EXTDIR contains either **S** or **R** to indicate whether the file is to be read or written. The I/O exit should use EXTWKARA to anchor any storage obtained and set EXTMAXLN to the maximum record length.

INFO Request

CONNECT:Direct makes an INFO request to an I/O exit when it wants the exit to input the file attributes and place them into the INFO area (mapped by the DMINFO macro) which is pointed to by EXTVSWRK. These data set attributes are required by CONNECT:Direct. DMINFO resides within the CDV3200 MACLIB. For example,

```
L       2, EXTVSWRK
USING   INFO,2
```

Set the following fields in the INFO control block. The values listed are an example of those needed for a sequential data set.

```
INBLKSZ = F'80'     block size
INLRECL = F'80'     record size
INTYPE  = CL4'PS'   data set organization
INRECFM             = CL4' '    blank
INUNIT  = CL8' '    blank
INBLKS  = F'0'      nulls
INUSEBLK            = F'0'      nulls
INBLKTRK            = F'0'      nulls
INTRKCYL            = F'0'      nulls
IN2NDRY = 8C'0'     character zeros
INLOCTYP            = CL3' '    blanks
```

GET Request

CONNECT:Direct makes a GET request to an I/O exit when it wants a record/block to be read into the buffer. EXTINLNG should be set to the length of the data. EXTINARA should point to the record obtained.

The exit must indicate normal END-OF-DATA condition to CONNECT:Direct by returning a value of EXTRCEOD in EXTRTNCD. You may indicate other conditions by providing other values in the previously mentioned fields. This allows messages that have been added to the CONNECT:Direct message file to be issued.

ADD Request

CONNECT:Direct makes an ADD request to an I/O exit when it wants a record/block to be inserted. EXTOTLNG is set to the length of the data. EXTOTARA points to the new record/block.

CLOSE Request

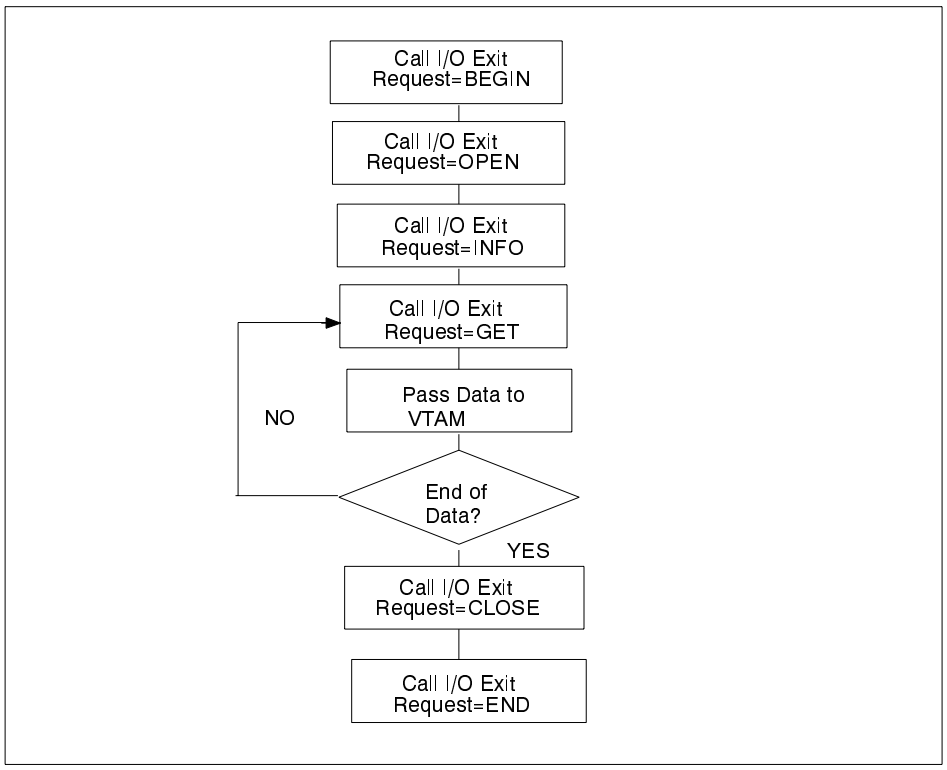
CONNECT:Direct makes a CLOSE request to an I/O exit when the file is to be closed. Since all I/O has been completed, errors returned by the exit on this request are ignored. The EXTABN flag is set on if the CLOSE request is due to abnormal termination.

END Request

CONNECT:Direct makes the END request to an I/O exit to end communication with the exit. The exit should release any work areas it may have allocated when the BEGIN request was received. This is the last request an I/O exit receives.

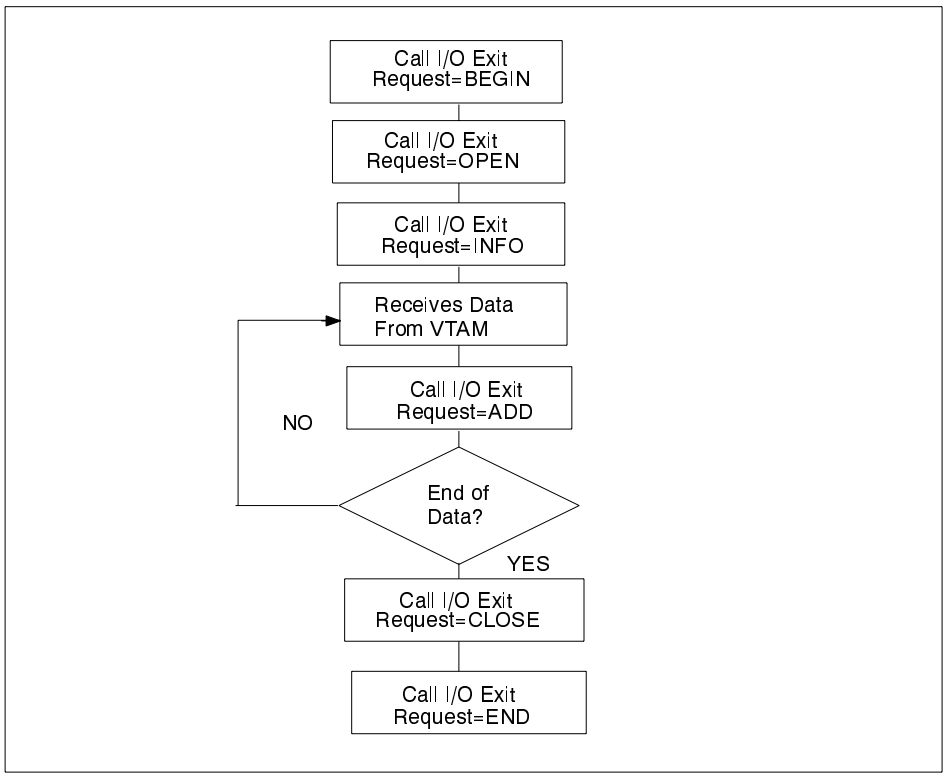
Normal Input Calling Sequence

This figure illustrates the normal input calling sequence for an I/O exit.



Normal Output Calling Sequence

This figure illustrates the normal output calling sequence of an I/O exit.



DMGIOX64 Exit

The DMGIOX64 exit is an I/O exit capable of handling files with up to 64K LRECL size.

System Requirements

In order to use the DMGIOX64 exit, the CONNECT:Direct system must be one of the following:

- CONNECT:Direct for VM Version 1.5 with P700120 maintenance
- CONNECT:Direct for VM/ESA Version 3.2

The minimum release level for VM/ESA and GCS is 2.0.

Note: You can only use the COPY function for VM to VM only. In addition, both systems must have the DMGIOX64 exit to ensure a successful transfer of the files.

Understanding How the DMGIOX64 Exit Works

CONNECT:Direct for VM normally uses the OS simulation capability of GCS to perform I/O functions. The DMGIOX64 exit implements an EXECIO in the GCS environment.

The EXECIO establishes a REXX environment within the GCS subtask in which a process is running. The DMGIOX64 exit calls EXECIO to read and write data through the EXECIO stem variables which capitalize on its 64K LRECL support.

Two examples follow.

```
RPLCOPY  PROCESS SNODE=VMCD02 NOTIFY=%USER CLASS=9
SCOPY    COPY  TO   (GROUP='%1% %2%'                                  -
                     LINK=(MAINT,WRITPASS,W,290)                      -
                     IOEXIT=(DMGIOX64)                                -
                     DISP=(RPL)                                       -
                     SNODE)                                           -
                FROM  (GROUP='* *'                                    -
                     LINK=(MAINT,ALL,RR,190)                          -
                     IOEXIT=(DMGIOX64)                                -
                     DISP=SHR                                         -
                     PNODE)
```

Both systems must have the DMGIOX64 exit to ensure a successful transfer of the files.

```
RPLCOPY  PROCESS SNODE=VMCD02 NOTIFY=%USER CLASS=2
SCOPY    COPY  TO   (DSN='DIRECTXA MODULE'                            -
                     LINK=(MAINT,WRITPASS,W,290)                      -
                     IOEXIT=(DMGIOX64)                                -
                     DISP=(RPL)                                       -
                     SNODE)                                           -
                FROM  (DSN='DIRECTXA MODULE'                          -
                     LINK=(MAINT,ALL,RR,190)                          -
                     IOEXIT=(DMGIOX64)                                -
                     DISP=SHR                                         -
                     PNODE)
```

Chapter 9

Customizing CONNECT:Direct

This chapter describes how to customize CONNECT:Direct to meet your user needs by:

- Adding Messages to the CONNECT:Direct Message Library
- Customizing Submit Screens

Adding Messages to CONNECT:Direct Message Library

You can load special user-defined messages into the CONNECT:Direct message library. The following sections show how to load a message into the CONNECT:Direct message library and provide a sample message source format.

Sample Format for Message Source

The sample format for the CONNECT:Direct message source, shown in the following figure, is in the MSGSOURC member. Use the format exactly as shown. Comments are not allowed.

```
DELETE =MSG00001
INSERT =MSG00001
MODULE =MSGSOURC
STEXT=This is an example of the short text message (one).
L01  = This is an example of the long text message (one).  As
L02  = many as 12 lines may be used for the long text message.
L03  =
L04  =
L05  =
L06  =
L07  =
L08  =
L09  =
L10  =
L11  =
L12  =
```

Observe the following rules for variables and message IDs:

- The DELETE, INSERT, and MODULE variables can be 1–8 characters.
- The STEXT and L01 through L12 variables can be 1–64 characters.
- To insert a message ID, INSERT, MODULE, STEXT, and L01 through L12 are required.
- To delete a message ID, DELETE is required.
- To replace a message ID, DELETE, INSERT, MODULE, STEXT, and L01 through L12 are required.

EXEC to Load CONNECT:Direct Message File

The following EXEC is in the MSGLOAD EXEC. After copying the message source into your message source library and making changes as needed, run this EXEC to add your messages to the CONNECT:Direct for VM/ESA message file.

```
/*    EXEC TO LOAD MESSAGES INTO THE NDM MESSAGE LIBRARY
/*    - CHANGE MODE TO THE MODE WHERE THE MESSAGE FILE IS LOCATED.  */
/*    - CHANGE HILQ TO THE HIGH LEVEL QUALIFIER.                    */
/*    - BE SURE A DLBL HAS BEEN ISSUED FOR IJSYSCT, THE VSAM        */
/*      MASTER CATALOG.                                             */
/*    - BE SURE A GLOBAL LOADLIB COMMAND FOR THE NDM LOAD LIBRARY   */
/*      HAS BEEN ISSUED.                                            */
FILEDEF SYSOUT TERM
FILEDEF INPUT DISK MSGSOURC NDMMSG
DLBL    NDMMSG MODE DSN 'HILQ.MSG'
SET     DOS OFF
OSRUN   DMMSGLOD
EXECOS
```

Customizing Submit Screens

CONNECT:Direct provides facilities to customize submit screens. The following sections describe the procedures necessary to accomplish this task.

When variables are entered into a submit screen, the IUI builds a SUBMIT command, and the command goes to a dialog for handling. As delivered on the installation tape, the primary IUI panel, DMI@PRIM, invokes the IUI submit panel, DMISUBMT, directly when the SB option is selected. See your *CONNECT:Direct for VM/ESA User's Guide* for an example of the Submit Process Screen.

You can construct a customized submit screen to contain customized submit options. All menus can contain as many choices as screen space allows. To customize the submit function modify DMI@PRIM to invoke DMI$SM03 instead of DMISUBMT when the SB option is selected, then perform the following steps:

1. DMI$SM03 is a submit menu panel which, as supplied, contains only one menu option which invokes DMISUBMT. Modify DMI$SM03 to include new menu selections to invoke the new customized submit panels.

2. Define at least one general purpose Process to be invoked by the new custom submit screen. This step is explained on page 9-4 in the section *Step 2 — Define a General Purpose Process*.

3. Provide custom submit screens. These screens:

   a. Process variables that are resolved as submission occurs.
   b. Build a command on each screen that communicates with the IUI dialog routines.

   This step is explained on page 9-5 in the section *Step 3—Provide a New Submit Screen*.

The following sections describe how to create a custom SUBMIT screen that copies a file to the existing file at another site at noon every day. The user is notified when the Process is complete.

Step 1 — Modify the Existing Menu DMI$SM03

The following describes how to modify the CONNECT:Direct Submit Menu. The following figure shows the Submit Menu after modification to accommodate an additional screen.

In Submit Menu (DMI$SM03), add the following:

- A line in the BODY section to specify the new option (+ 2 = => COPY TO EXISTING FILE AT ANOTHER SITE EVERYDAY AT NOON)
- A line in the PROC section to specify what to do when that option is selected (2,'PANEL(CUSTSUBM)')

The following figure shows the menu displayed after information has been added. Only the elements necessary to modify an existing menu are shown. If the user selects Option 2 on the command line, CONNECT:Direct gives control to the screen with the name CUSTSUBM and displays that screen.

```
)ATTR
  ”
)BODY
#UNODE           +       SUBMIT MENU                          +&ZDATE
+CMD%= = > _ZCMD                                              +&ZTIME
#STEXT
+
%PROCESSES:
+
+ 1 = = >  SUBMIT A PROCESS
+ 2 = = >  COPY TO EXISTING FILE AT ANOTHER SITE EVERYDAY AT NOON
+
)INIT
   ”
)PROC
   ”
   &SEL = TRANS(TRUNC(&ZCMD,'.')
          ”
          ”
          1,'PANEL(DMISUBMT)'
          2,'PANEL(CUSTSUBM)'
          *,'?'
     ”
)END
```

Step 2 — Define a General Purpose Process

The second step in creating a customized submit screen is to define a Process that will be invoked by the custom Submit screen. The following Process is named APROC.

```
APROC   PROCESS SNODE=&SNODE NOTIFY=%USER
STEP1   COPY    FROM (PNODE DSN=&DSN1  DISP=SHR) -
                TO (DSN=&DSN2 DISP=(SHR,KEEP))
```

The SNODE, source file name, and the destination file name specified in the BODY section of the custom submit screen are substituted into the symbolic fields currently in the PROCESS (SNODE, DSN1, and DSN2).

Step 3 — Provide a New Submit Screen

A new submit screen processes variables resolved during submission and builds a command to communicate with the dialog routines. Although the customized submit screen can have any appearance you prefer, the following information helps you design your screens:

1. Use the generic submit screen (CUSTSAMP) found on the DTF's 191 minidisk, as a base for creating the custom screen.

2. Use the least number of input fields necessary when creating the screen to accomplish Process submission.

3. Use existing variables from CUSTSAMP, if possible. You can use any variable name, but fewer changes are necessary when you use the existing code. The following ISPF variables are used in the CONNECT:Direct submit processing.

Variable	Explanation
&PNAME1	Name of Process to be submitted
&DSN	File name containing Process to be submitted
&SNODE	Secondary node name
&H	HOLD specification
&R	RETAIN specification
&PR	Priority of the Process
&NEWNAME	New name for the Process being submitted
&CLS	Process class
&NOTIFY	Notify CONNECT:Direct userid
&PNODEID	Security userid at PNODE
&PNODEPW	Current security password at PNODE
&PNODENPW	New security password at PNODE
&SNODEID	Security userid at SNODE
&SNODEPW	Current security password at SNODE
&SNODENPW	New security password at SNODE
&DSYMBPARM	Symbolic variation specification
&STIME	Start time value

(continued)

Variable	Explanation
&SDATE	Start day/date value
&CMD1	Used in constructing command string
&CMD2	Used in constructing command string
&CMD3	Used in constructing command string
&CMD4	Used in constructing command string

Note: The only ISPF/PDF variables that you should not change are &CMD1, &CMD2, &CMD3, and &CMD4. The SUBMIT command string is built into these four variables.

4. The following figure shows the ATTR and BODY sections of the customized submit screen CUSTSUBM. The ATTR section is the same as in the sample base screen, CUSTSAMP. As shown in the BODY section, FROMDSN and TODSN are variables that are symbolically substituted when the general purpose Process APROC is submitted.

```
)ATTR

  +  TYPE(TEXT) INTENS(LOW) SKIP(ON)
     :    TYPE(INPUT) INTENS(NON)
  #  TYPE(OUTPUT) INTENS(HIGH) JUST(ASIS) CAPS(OFF)
  @  TYPE(OUTPUT) INTENS(LOW) JUST(ASIS)
  ¢  PAD(_)
  )BODY
  #UNODE                                      + CUSTOMIZED SCREEN
  +CMD%= = > _ZCMD
  #STEXT                                      + TIME-&ZTIME
  %  COPY TO EXISTING FILE AT ANOTHER SITE    + DATE-&ZDATE
  %          EVERY DAY AT NOON                + JULIAN-&ZJDATE
  +   FILE TO BE SENT FROM HERE
    %  = = >   ¢FROMDSN                       +
  +
  +  NODE TO RECEIVE THE FILE
  %  = = >  ¢SNODE                            +
  +
  +  RECEIVING FILE ON ABOVE NODE
  +  = = >  ¢TODSN
                                              +
```

5. Make the necessary changes in the INIT section after deciding how you want to set up the screen. Initialize all INIT section variables to

the appropriate default value. For CUSTSUBM, the INIT section is shown in the following figure.

```
)INIT
  .ZVARS = '(V@SEC)'
  &NXTHELP = DMJSBMT1
 .CURSOR  = &FROMDSN             /*CHANGED  */
  &UNODE1 = &UNODE1
  &SPC    = ''
  &V@SEC  = 'N'
  IF (&PROC ~= 'Y')
    &STEXT    = ''
/*  &PNAME1   = ''                  DELETED  */
/*  &DSN      = ''                  DELETED  */
    &SNODE    = ''
/*  &Q        = ''                  DELETED  */
/*  &H        = ''                  DELETED  */
    &R        = 'Y'               /*CHANGED  */
/*  &PR       = ''                  DELETED  */
/*  &NEWNAME  = ''                  DELETED  */
/*  &CLS      = ''                  DELETED  */
    &STIME    = '12:00'           /*CHANGED  */
/*  &SDATE    = ''                  DELETED  */
/*  &SYMBPARM = ''                  DELETED  */
    &FROMDSN  = ''                /* ADDED   */
    &TODSN    = ''                /* ADDED   */
/*  IF (&UNODE  = &LNODE)           DELETED  */
      &NOTIFY   = '%USER'
/*  IF (&UNODE ~= &LNODE)           DELETED  */
/*    &NOTIFY   = ''                DELETED  */
  IF (&PROC = 'Y')
    IF (&PROC# ~= 'NONE')
      .MSG = IUSB000I
  IF (&STEXT = '')
    .HELP = &NXTHELP
  IF (&STEXT ~= '')
    .HELP  = DMI@MSG
```

The following changes were made in the INIT section:

- The cursor field was changed to FROMDSN.
- &R was changed to *Y* to indicate RETAIN=YES.
- &STIME was changed to *12:00* to indicate STARTT=(,12:00).
- &FROMDSN and &TODSN were added and initialized to blanks.
- Lines which carry a DELETED comment were removed because they are no longer necessary.

6. Make the necessary changes in the PROC section after deciding how you want to set up the screen. Verify PROC section field values and

build the command string to submit the Process. For CUSTSUBM, the PROC section is shown in the following figure.

```
)INIT
)PROC
  &PROC# = 'NONE'
  &CMD = &ZCMD
  &SEL = TRANS( TRUNC (&ZCMD,'.')
         SPF,'PANEL(ISR@PRIM) NEWAPPL(ISR)'
         WHO,'PANEL(DMI@WHO)'
         SW,'PGM(DMICMD) PARM(&CB@)'
         M,'PANEL(DMI@MSG)'
         AUTH,'PANEL(DMI@AUTH)'
         ' ','PGM(DMICMD) PARM(&CB@)'
         *,'?' )
  &ZTRAIL = .TRAIL
  IF (&ZCMD = 'SW')
    &SPC = 'SLN'
  IF (&CMD = ' ')
    VER(&FROMDSN,NONBLANK)                          /* ADDED   */
    VER(&SNODE,NONBLANK)                            /* ADDED   */
    IF (&TODSN= '')                                 /* ADDED   */
      &TODSN=&FROMDSN                               /* ADDED   */
/*  VER(&PNAME1,NAME,MSG=IUSB001I)                    DELETED  */
/*  IF (&PNAME1 ~= ' ')                               DELETED  */
/*    IF (&DSN ~= ' ')                                DELETED  */
/*      .MSG = IUSB002I                               DELETED  */
/*      .CURSOR = PNAME1                              DELETED  */
/*  IF (&DSN = ' ')                                   DELETED  */
/*    VER(&PNAME1,NONBLANK,MSG=IUSB003I)              DELETED  */
/*  VER(&Q,LIST,Y,N,MSG=IUSB022I)                     DELETED  */
/*  VER(&H,LIST,Y,N,C,MSG=IUSB005I)                   DELETED  */
    VER(&R,LIST,Y,N,I,MSG=IUSB006I)
/*  VER(&V@SEC,LIST,Y,N,I,MSG=IUSB006I)               DELETED  */
/*  VER(&PR,RANGE,0,15,MSG=IUSB007I)                  DELETED  */
/*  VER(&NEWNAME,NAME,MSG=IUSB008I)                   DELETED  */
/*  VER(&CLS,RANGE,1,255,MSG=IUSB009I)                DELETED  */
    &USER = TRUNC(&NOTIFY,1)
    IF (&USER ~= %)
      VER(&NOTIFY,NAME,MSG=IUSB010I)
    &PROC = 'Y'
    &CMD1 = ' SUB PROC=APROC'                       /*CHANGED  */
    &CMD2 = ''
    &CMD3 = ''
    &CMD4 = &SYMBPARM
    IF (&PNAME1 ~= ' ')                             /*CHANGED  */
      &CMD1 = '&CMD1 PROC=&PNAME1'                  /*CHANGED  */
    IF (&DSN ~= ' ')                                /*CHANGED  */
      &CMD1 = '&CMD1 DSN=&DSN'                      /*CHANGED  */
    IF (&SNODE ~= ' ')
      &CMD1 = '&CMD1 SNODE=&SNODE'
/*  IF (&Q ~= ' ')                                    DELETED  */
/*    &CMD1 = '&CMD1 REQUEUE=&Q'                      DELETED  */
/*  IF (&H ~= ' ')                                    DELETED  */
/*    &CMD1 = '&CMD1 HOLD=&H'                         DELETED  */
    IF (&R ~= ' ')
      &CMD1 = '&CMD1 RETAIN=&R'
/*  IF (&PR ~= ' ')                                   DELETED  */
/*    &CMD1 = '&CMD1 PRTY=&PR'                        DELETED  */
/*  IF (&NEWNAME ~= ' ')                              DELETED  */
/*    &CMD1 = '&CMD1 NEWNAME=&NEWNAME'                DELETED  */
    IF (&NOTIFY ~= ' ')
      &CMD2 = ' NOTIFY=&NOTIFY'
    &PARMX   = ''
    &PARMX2  = ''
/*  IF (&SDATE ~= ' ')                                DELETED  */
/*    &PARMX = 'Y'                                    DELETED  */
/*    &CMD2 = '&CMD2 STARTT=(&SDATE,'                 DELETED  */
    IF (&STIME ~= ' ')
      IF (&PARMX ~= 'Y')
        &PARMX  = 'Y'
        &PARMX2 = 'Y'
        &CMD2 = '&CMD2 STARTT=(,&STIME'
      IF (&PARMX2 ~= 'Y')
        &CMD2 = '&CMD2 &STIME'
    IF (&PARMX = 'Y')
      &CMD2 = '&CMD2'
/*  IF (&CLS ~= '')                                   DELETED  */
/*    &CMD2 = '&CMD2 CLASS=&CLS'                      DELETED  */
/*  IF (&V@SEC = 'Y')                                 DELETED  */
/*    &ZSEL = 'PANEL(DMIUSRID)'                       DELETED  */
    &CMD3= '&&DSN1=&FROMDSN'                        /* ADDED   */
    &CMD3= '&CMD3 &&DSN2=&TODSN'                    /* ADDED   */
)END
```

The following changes were made in the PROC section:

- A verify was added for the &FROMDSN variable. It must be nonblank.
- A verify was added for the &SNODE variable. It must be nonblank.
- A test for blanks in &TODSN was added. If &TODSN is blank, it will be set to &FROMDSN.
- &CMD1 was changed to contain the string SUB PROC=APROC. This is the command default.
- &CMD3 was added to contain the string &&DSN1=&FROMDSN. This is how symbolic substitution is accomplished. When APROC is submitted, &DSN1 is translated to whatever value is in the &FROMDSN file.
- The next-to-last line was added to concatenate the string &&DSN2=&TODSN to what is already in &CMD3. When APROC is submitted, &DSN2 is translated to whatever value is in &TODSN.
- Lines which carry a DELETED comment were removed because they are no longer necessary.

7. If Y12.FROMHERE is the file to be sent, CD.THERE is the node to receive the file, Z12.TOHERE is the receiving file, and the Process is APROC, then the command string is built as follows.

```
SUB  PROC=APROC SNODE=CD.THERE RETAIN=Y NOTIFY=%USER      -
       STARTT=(,12:00) &DSN1=Y12.FROMHERE &DSN2=Z12.TOHERE
```

When the Process APROC is submitted, it is resolved as follows.

```
APROC   PROCESS   SNODE=CD.THERE NOTIFY=%USER
STEP1   COPY      FROM (SNODE DSN=Y12.FROMHERE DISP=SHR)-
                     TO   (DSN=Z12.TOHERE DISP=(SHR,KEEP))
```

Chapter 10

Administering Statistics

This chapter includes the following topics:

- Understanding the Statistics Facility
- Monitoring the Statistics Facility
- Tuning the Statistics files
- Changing the file pair configuration
- Archiving statistics
- Displaying the status of the Statistics logging facility
- Displaying the Statistics Archive file directory
- Switching the Statistics file pair
- Recording Statistics for specific record types
- Notifying CONNECT:Direct of Statistics file archival

Understanding the Statistics Facility

The CONNECT:Direct statistics facility logs statistics to a series of VSAM file pairs. Each pair consists of an entry-sequenced file and a key-sequenced file, both with the REUSE attribute.

Understanding File Pair Configuration

Within each file pair, CONNECT:Direct writes the statistics records to the entry-sequenced file, while the key-sequenced file maintains index information about the records. On average, CONNECT:Direct writes records to the key-sequenced file at the rate of about one for every two records written to the entry-sequenced file. The minimum configuration is two file pairs, or four files.

Configuring and Updating File Pairs

Perform the following steps to either update existing file pairs or to configure new file pairs to the statistics file pair list:

1. Use the STAT.DSN.BASE initialization parameter to specify the data set name high level qualifiers.

2. Specify the number of file pairs using the STAT.FILE.PAIRS initialization parameter.

3. Review the statistic file pairs list to verify your changes.

Retrieving Statistics with the SELECT STATISTICS Command

All the file pairs defined to the DTF are available to SELECT STATISTICS command processing. CONNECT:Direct searches any file pair that may contain records that satisfy the SELECT STATISTICS command, not just the files currently being written.

When you issue SELECT STATISTICS commands, the system locates the requested records by using the key-sequenced file as an index to the entry-sequenced file.

How Records are Written

CONNECT:Direct writes the statistics records to the entry-sequenced VSAM files in chronological order, starting at the beginning of the file and proceeding until the file or its paired key-sequenced file is full. The oldest record is always at the beginning of the file and the newest record is at the end. The system records each statistics record as a single VSAM record. The system does not compress the records or add control information.

Switching Files

When a file pair is full, the system switches to the next in the sequence, and begins writing to it. Think of the list of file pairs as processing in a circular fashion, or as being *wrapped*. When the last file pair in the list is full, the system wraps back to the first pair in the sequence.

In addition to switching to a new file when the active one becomes full, a switch occurs at certain times of the day, if you specify the time in the STAT.SWITCH.TIME initialization parameter. For example, you can specify that statistics files must switch at midnight every day, thereby limiting a file pair to records from a single day. CONNECT:Direct also provides a statistics switch API command that directs the DTF to perform a switch at any time.

Once the system writes to all the pairs in the list, the system reuses the pairs. When a switch is made, the system closes the active pair and makes the pair with the oldest data the new active pair. When the system switches to a file, or a file becomes active, CONNECT:Direct does a VSAM RESET. This erases any records and index information in the active file. The system then writes new records starting at the beginning of the file.

Monitoring the Statistics Facility

CONNECT:Direct provides the following means of monitoring various aspects of the statistics facility.

- INQUIRE STATISTICS command
- Type S2 statistics records
- Type SS statistics records
- SCCSTAT utility

Using the INQUIRE STATISTICS Command

The INQUIRE STATISTICS command gives a *snapshot* of the status of the statistics facility. See page 10-15 for an explanation of the INQUIRE STATISTICS command. INQUIRE STATISTICS produces a report that includes the following:

- List of any currently EXCLUDEed record types
- File pair list configuration including indication of active file pairs
- Date and time range covered by each file pair
- Size of each file
- Utilization percentage of the entry-sequenced files
- Count of active SELECT STATISTICS commands for each file pair
- Logging data to verify if a file pair reset is waiting for archive to end
- Reason for the last switch from each file pair
- Most recent file access return code and message ID for each file
- Utilization percentage of the nonactive key-sequenced files
- Archive notification data to verify the receipt of nonactive files
- Indication if logging is waiting for SELECT STATISTICS to finish so a file pair can be reset

Monitoring Statistics with S2 Statistics Records

The S2 statistics records contain information about the statistics logging function. The system writes the records about once per hour when there is activity in the DTF. Each S2 record contains statistics about the period

of time since the prior S2 record was written. The S2 statistics records include the following information:

- Beginning time and length of the period covered
- Count of records written in the period
- Count of ESDS control intervals written in the period
- Count of total bytes written to the ESDS
- Average statistics record length
- Average records per control interval
- Average ESDS writes per second
- Average KSDS writes per second
- Average logging service time
- Total waits for logging queue element
- Each indexed field including max keys and average keys per control interval

Use the TYPE parameter of the SELECT STATISTICS command to view the S2 records. The system writes the S2 records with the userid specified in the STAT.USER initialization parameter. If you code a unique ID for STAT.USER and you specify the USER parameter on the SELECT STATISTICS request, you will greatly reduce the search time because the userid is an indexed field.

For example, if you code STAT.USER=statuser, a SELECT STATISTICS request to display all S2 records would look like the following figure.

```
SELECT STATISTICS WHERE (USER=statuser, TYPE=(S2)) TABLE
```

See the *CONNECT:Direct for VM/ESA User's Guide* for more information on how to use the SELECT STATISTICS command.

Using SS Statistics Records

The SS statistics records contain information about SELECT STATISTICS processing. One SS record is written for each SELECT STATISTICS command that executes. The SS record includes information such as which index was used to search the files and how many requests were issued to the keyed and entry-sequenced clusters. The record also includes how many records were examined and rejected.

Use the SELECT STATISTICS command with the TYPE parameter to view the SS record. See the *CONNECT:Direct for VM/ESA User's Guide* for more information on how to use the SELECT STATISTICS command.

Using selection criteria with the SELECT STATISTICS request improves the performance by efficiently locating the requested records.

For example, you can include the ID of the user that issued the SELECT STATISTICS command or the approximate time the request was issued, using the STARTT, STOPT, and USER parameters. The following figure shows an example using this selection criteria.

```
SELECT STAT WHERE      -
(TYPE=(SS) USER=USER1 STARTT=(,NOON)    -
STOPT=(,13:00)) TABLE
```

Using the SCCSTAT Utility to Determine File Usage

Use the SCCSTAT utility to find out the rate at which the system generates statistics records. SCCSTAT also performs an analysis of the contents of the statistics file showing what percentage of the records are of each record type. This utility is executed by the SCCSTAT EXEC or the SCCSTAT1 EXEC to analyze a single statistics entry-sequenced file.

To help with your estimates of statistics file usage, CONNECT:Direct provides the utility programs SCCSTAT1 and SCCSTAT.

- SCCSTAT1 reports on the contents of the statistics file for previous versions of CONNECT:Direct for VM.
- SCCSTAT reports on the statistics files for Version 3.2.00. It calculates the average number of CIs used per day at one DTF.

SCCSTAT1 Utility

Use the SCCSTAT1 EXEC to report on the contents of the statistics file for CONNECT:Direct versions prior to Version 3.2.00.

Perform the following steps to use the SCCSTAT1 utility:

1. Set the required variables to run the EXEC. The following table defines each of the required variables for the SCCSTAT1 EXEC.

Variable	Definition
oldmcat	Indicates the VSAM Master Catalog DSName of the prior version of CONNECT:Direct for VM
olduid	Indicates the VSAM userid name of the prior version of CONNECT:Direct for VM
oldmdev	Indicates the VSAM minidisk for the prior version of CONNECT:Direct for VM
oldhlq	Indicates the VSAM qualifier for the prior version of CONNECT:Direct for VM

(continued)

Variable	Definition
mastcat	Indicates the VSAM Master Catalog of CONNECT:Direct for VM/ESA Version 3.2.00
loadlib	Indicates the LOADLIB name of CONNECT:Direct for VM/ESA Version 3.2.00

2. Execute the SCCSTAT1 EXEC to obtain an analysis of the pre-3.2 STATS file.

SCCSTAT Utility

Use the SCCSTAT EXEC to report on the contents of the statistics file for CONNECT:Direct for VM/ESA Version 3.2.00.

Perform the following steps to use the SCCSTAT utility:

1. Set the required variables to run the EXEC. The following table defines each of the required variables for the SCCSTAT EXEC.

Variable	Definition
hlq3200	Indicates the VSAM high-level qualifier for CONNECT:Direct for VM/ESA Version 3.2.00
loadlib	Indicates the LOADLIB name of CONNECT:Direct for VM/ESA Version 3.2.00
mastcat	Indicates the VSAM Master Catalog of CONNECT:Direct for VM/ESA Version 3.2.00

2. Execute the SCCSTAT EXEC to run an analysis of the Version 3.2 STATS file ESDS.

Tuning the Statistics Files

This section describes how to determine the most efficient use of your Statistics file space.

Statistics Files Space Allocation Example

In this example, CONNECT:Direct was installed using the default statistics installation of two file pairs with a total capacity of 13,500

records. After running CONNECT:Direct for a time, a user determines that this allows the logging of records for about 2.5 days before the file pair list wraps. The administrator wants to provide space for 7 days worth of records to be available at any given time.

The administrator does the following:

1. Use SCCSTAT to determine the number of records written daily.

   Run the SCCSTAT utility against the statistics entry-sequenced clusters to determine the rate at which the system generates the statistics records.

   For example, SCCSTAT shows that records per day is approximately equal to 5,400.

2. Determine the total capacity of the statistics file.

   ```
   capacity = (records per day) * days
   ```

   Determine the total capacity in this example by multiplying the 5,400 records per day by 7 days. In this case, the total capacity of the statistics file is 37,800 records.

3. Determine the number of records per file pair.

   In this example, the administrator decides to define four file pairs, so each will be given a capacity of 9,500 records, for a total capacity of 38,000 records.

4. Determine the RECORDS parameter value for the key-sequenced clusters.

   ```
   KSDS-records = 0.75 * (ESDS-records)
   ```

   Determine the number of KSDS records by multiplying 75% by 9,500, the number of records per file pair. The RECORDS parameter value for the key-sequenced clusters is 7,125.

Based on these calculations, the administrator allocates four file pairs. The entry-sequenced cluster (ESDS) of each pair is defined with RECORDS(9500). The key-sequenced clusters (KSDS) are defined with RECORDS(7125).

Changing the File Pair Configuration

Make changes to the statistics files or to the configuration of the file pair list while the DTF is not running. During DTF execution, the files remain allocated by CONNECT:Direct.

The restrictions that CONNECT:Direct places on changes to the configuration maintain the integrity of the facility. At DTF initialization time, CONNECT:Direct checks the usability, validity, and accessibility of the statistics files data.

How File Pair Verification Works

CONNECT:Direct performs a verification procedure at initialization as follows:

1. Within each file pair, CONNECT:Direct verifies the appropriate sizing, relative to each other, of the entry-sequenced cluster and the key-sequenced cluster.

   Note: If the file pair is not relatively sized, then CONNECT:Direct issues a warning message and initialization continues.

2. If either of the files of a pair has data, CONNECT:Direct attempts to verify that the two files are actually a statistics file pair.

3. CONNECT:Direct verifies that the key-sequenced file contains index information for the associated entry-sequenced file.

4. CONNECT:Direct uses control information maintained in the key-sequenced file to perform the verification.

   Note: CONNECT:Direct keeps a control record in the KSDS that contains the data set name and the control interval size of the paired entry-sequenced file. If this information does not match, statistics initialization fails.

There are a few implications of this verification.

- Changing the control interval size of the ESDS or renaming the clusters causes initialization to fail because the control record in the KSDS no longer matches the files. There are two ways to resolve this statistics initialization failure:

- First, you can use the DMSTBKEY utility to rebuild the key-sequenced cluster. This recreates the KSDS control record so that it matches the new names or control interval size. In this way the records in the file pair remain accessible when the DTF is available again.

- The second solution is to empty the files. In this way the file pair is available for logging new records; however, the old records are no longer available. You may want to archive the files before emptying them.

- Changing the size of a file pair should not be a problem. The sizes of both files of a pair should change together so that the relative sizes do not change.

  - If the files are made larger, and the names remain the same, then copy the records from the old smaller entry and key-sequenced clusters to the new larger ones.

  - Use the same procedure to make a file pair smaller, if all the existing records from both files will fit into the smaller space of the new files. If the existing records do not fit, then the new smaller file pair must be left empty initially, and the old records will become unavailable.

File Pair List Verification

CONNECT:Direct generates the statistics file pair list from the initialization parameters STAT.DSN.BASE and STAT.FILE.PAIRS. For an example of a file pair list, refer to page A-31 in the *Initialization Parameters* appendix.

The CONNECT:Direct statistics facility processes the statistics file pair list in a circular, or *wrap-around* fashion. The system maintains statistics records in strictly chronological order both within each file pair, and with regard to the file pairs in the list.

At DTF initialization time, unless STAT.INIT=COLD is specified in the initialization parameters, CONNECT:Direct verifies that the order of the file pairs is valid. This is done by examining the date and time range of each non-empty file pair in the list. These must be in strictly ascending order throughout the list with the exception of across the *wrap point*. Empty file pairs may appear anywhere in the list.

To change the number of file pairs, perform the following steps:

1. Use the STAT.FILE.PAIRS initialization parameter to add or remove file pairs from the statistic file pair list.

2. Add empty file pairs to the end of the list unless you specify STAT.INIT=COLD.

3. Remove records from the end of the list by reducing the STAT.FILE.PAIRS value.

Note: When you remove these records, they become unavailable and may in some cases leave *gaps* in the statistics data. You may want to archive these records before removing them.

Archiving Statistics

Archiving refers to the process of copying the records from a statistics entry-sequenced cluster to another data set for long-term storage. The output of this process is an archived statistics file. You can write the archive file to a VSAM entry-sequenced cluster with the same characteristics as a statistics ESDS, or to a non-VSAM sequential file on DASD. The system does not store the statistics records in the ESDS in any special format. The system records each statistics record as a VSAM record in an ordinary VSAM ESDS. You can also write the archive file to a magnetic tape or a database table.

CONNECT:Direct provides a number of features for archiving statistics records.

Archiving Using a Predefined Process

By using the DTF initialization parameter STAT.SWITCH.SUBMIT, you can specify that when the DTF switches from one statistics file pair to another, CONNECT:Direct submits a pre-defined Process to archive the records in the previously active ESDS. CONNECT:Direct submits this archive Process with a symbolic parameter indicating the data set name of the ESDS of the pair.

The Process can then use CONNECT:Direct to copy the data to another location. A sample archive Process is in SAMPLIB member ARCHSTAT.

Timing the Archive

The archive must complete before CONNECT:Direct needs to reuse the file being archived, that is, at the time of wrap-around of the file pair list. The completion of the archiving Process is important because CONNECT:Direct erases the contents of the statistics file when the system switches to that file. In other words, archiving must complete within the time required for the file pair list to wrap. Normally, this should not present a problem.

Several options are available to coordinate the archiving of statistical data with the reusage of the file pairs.

Requiring Confirmation of Archival

The STAT.ARCH.CONFIRM initialization parameter specifies whether or not to ensure that data is archived before the system erases the file. If you do not want archiving, CONNECT:Direct simply resets the files when the switching occurs, and begins writing. If you want archiving, CONNECT:Direct verifies that the archive is complete before proceeding. In this case, CONNECT:Direct requires notification of archival. There are several ways to notify CONNECT:Direct.

- If the archive is done using the COPY statement in a CONNECT:Direct Process, then the Process can also invoke the DMSTARRT utility when the COPY successfully completes. CONNECT:Direct invokes DMSTARRT through a RUN TASK statement, and notifies CONNECT:Direct that the data has been archived.

- Also, you can issue the API command STATISTICS ARCHIVED to inform CONNECT:Direct to reuse a file pair.

If there is no indication regarding the completion of the archive when CONNECT:Direct needs to reuse the files, the system issues a message to the operator console and waits for a reply indicating permission to reuse the file.

Note: In this situation, all activity in the DTF ceases until there is a response to the message from the operator indicating that the statistics file can now be over-written. This occurs as a result of the request that the DTF not erase statistics data unless it is certain that archiving the statistics is complete.

Not Requiring Confirmation of Archival

CONNECT:Direct does not require archive confirmation before reusing a statistics file pair when you specify or default to the DTF initialization parameter, STAT.ARCH.CONFIRM=NO. If you specify this

initialization parameter, you are responsible for ensuring that the archive successfully completes before CONNECT:Direct resets the file. If the file is reset before copying the records, the data is lost.

If the records are in the process of being copied when CONNECT:Direct needs to reset the file, then CONNECT:Direct must wait for the copy to complete. This is because CONNECT:Direct must have exclusive access to the file to do the VSAM reset. In this situation, CONNECT:Direct also issues a message to the operator console and waits for a reply indicating that the file can be reset.

Using the SELECT STATISTICS Command with Archived Statistics

CONNECT:Direct for VM/ESA provides a means of issuing the SELECT STATISTICS command against archived statistics. To make archived statistics available to SELECT STATISTICS, you must put the archived statistics in the format of a statistics file pair. You must make available a VSAM entry-sequenced cluster with a paired VSAM key-sequenced cluster containing the index information.

For example, if the records are archived to a magnetic tape file, you must first copy the archived records to a VSAM ESDS. Then you can run the DMSTBKEY utility to build the required associated VSAM KSDS. Refer to the *CONNECT:Direct for VM/ESA Installation Guide* for a discussion of estimating space requirements and information about the characteristics and relative sizes of the keyed and entry-sequenced clusters of a file pair. See page 10-14 of this manual for an explanation and example of the DMSTBKEY utility.

Use the ARCHDSN parameter of the SELECT STATISTICS command to search archive files. The ARCHDSN parameter names only the key-sequenced clusters of the archive pairs; CONNECT:Direct locates the associated entry-sequenced clusters using control information in the key-sequenced clusters.

CONNECT:Direct does not examine the statistics file pair list of the DTF when using the ARCHDSN parameter. CONNECT:Direct only searches the archive files. SELECT STATISTICS processing does not let you name files currently in the file pair list of the DTF in the ARCHDSN parameter or combine archive files with files in the file pair list.

Refer to the *CONNECT:Direct for VM/ESA User's Guide* for a description of the SELECT STATISTICS command and the ARCHDSN parameter.

Maintaining an Archive File Directory

CONNECT:Direct also provides the capability of maintaining a directory of statistics archive files. The directory is a VSAM key-sequenced file that contains a record for each archive file.

Information in the record includes the data set name of the archive file and the range of dates and times covered by the archived records. Refer to the *CONNECT:Direct for VM/ESA Installation Guide* for an explanation of estimating space requirements allocating the directory file.

In order to use the directory feature, you must allocate the directory file and specify its name in the STAT.ARCH.DIR initialization parameter. CONNECT:Direct provides a means of viewing the directory contents by using the INQUIRE STATDIR command, described on page 10-17.

The archive notification utilities, DMSTARRT, writes the directory records. If you want to use the directory feature, you must execute DMSTARRT from the Process that archives the records. This is true even if you do not specify STAT.ARCH.CONFIRM=YES in the DTF initialization parameters. You must also use DMSTARRT to send archive notification when you are not using the directory feature, but specify STAT.ARCH.CONFIRM=YES in the Initialization Parameters file.

Archive Related Utilities

This section explains the archiving related utilities: DMSTARRT and DMSTBKEY.

DMSTARRT

The DMSTARRT utility (STatistics ARchive Run Task) has the following functions:

- Notifies CONNECT:Direct of the availability of a statistics file pair for reuse due to the completion of archiving
- Optionally adds an entry to the directory of archive files

You can invoke DMSTARRT from within a Process through the RUN TASK statement. You should use this utility when submitting a Process at statistics file pair switch time that archives the statistical data using CONNECT:Direct to COPY the statistics to another file. Once the COPY successfully completes, the system can update the directory and send the archive notification.

The program accepts two parameters through the RUN TASK statement.

- The first parameter is required and is the data set name of the statistics entry-sequenced cluster that was archived.
- The second parameter is optional, and is the data set name of the archive file.

DMSTARRT *always* sends archive notification to the DTF. If you specify STAT.ARCH.CONFIRM=NO and there is no requirement of notification, the notification has no effect.

The addition of the entry in the directory of archive files is done optionally depending on the specification of the second parameter string. If the second parameter is present, then the system updates the directory to contain an entry for the new archive file.

The following is an example of an archive Process. This Process copies a statistics file to a sequential tape file and then invokes DMSTARRT to send archive notification to the DTF and update the directory of archive files. CONNECT:Direct passes the data set name of the statistics file to the Process in the form of the symbolic parameter &EDSN.

```
ARCHSTAT PROCESS SNODE=secondary.node                              -
                 STARTT=(TODAY)                                   -
                 &EDSN=
ARC      COPY    FROM (DSN=&EDSN)                                 -
                       LINK=($OWNER,$MULTPW,MW,$CUU)              -
                       DCB=(DSORG=VSAM)                           -
                       DISP=SHR                                   -
                 TO   (DSN=archive.file)                          -
                       DISP=(RPL)
          RUN TASK (PGM=DMSTARRT,PARM=("&EDSN",                   -
                                       "archive.file"))
         EIF
```

DMSTBKEY

The DMSTBKEY utility (STatistics Build KEYs) loads a statistics key-sequenced cluster with index information for an associated statistics entry-sequenced cluster. The DMSTBKEY executes the DMSTBKEY utility.

DMSTBKEY allows the recreation of index information for archived statistics data so that you can issue a SELECT STATISTICS command. You can also use this utility to rebuild index information for statistics files in the DTF file pair list in certain cases. Refer to the *Changing the File Pair Configuration* section on page 10-8 for additional information.

DMSTBKEY requires the allocation of DDNAMEs, ESDSnn, and KSDSnn with the entry-sequenced and key-sequenced clusters respectively. CONNECT:Direct loads the entry-sequenced cluster with the statistics records for building the index information. The key-sequenced cluster must either be empty, or be defined with the REUSE attribute. DMSTBKEY erases any records in the KSDS before writing the new information.

The size of the KSDS should be about 15% of the size of the associated ESDS. The KSDS must have the characteristics of a statistics key-sequenced cluster. Refer to the *VSAM Files DASD Requirement and*

Description section in the *CONNECT:Direct for VM/ESA Installation Guide* for details about allocating statistics clusters.

The sample EXEC in the following figure, DMSTBKEY, is found on the GCS virtual machine minidisk.

```
/* REXX EXEC for rebuilding STAT KSDS file from ESDS file          */
trace "N"

/* It is recommended that you always run with an ESTA filedef      */
'FILEDEF ESTAE DUMMY'

/* Set SYSOUT output to the terminal                               */
'FILEDEF SYSOUT TERMINAL'

/* ************************************************************** */
/* Step 1 - Set REXX variables used by this EXEC                  */
/* ************************************************************** */

mast cat = 'MASTCAT' /* C:D VM 3.2.00 VSAM Master Catalog DSName   */
hlq3200  = 'CDV3200' /* C:D VM 3.2.00 VSAM high level qualifier    */
loadlib  = 'CDV3200' /* C:D VM 3.2.00 LOADLIB name                 */

/* ************************************************************** */
/* Step 2 - Rebuild 3.2.00 STATS file KSDS from ESDS data         */
/* ************************************************************** */

/* Set DLBL for Master Catalog                                     */
'DLBL IJSYSCT V DSN' mastcat'(PERM'                                */
/* Set DLBLs for STATS file pairs                                  */
'DLBL ESDS01 V DSN' hlq3200'.STAS.ESDS01 (VSAM'
'DLBL KSDS01 V DSN' hlq 3200'.STATS.KSDS01 (VSAM'
/* Set LOADLIB name for the C:D VM 3.2.00 program library          */
'GLOBAL LOADLIB' loadlib
/* Execute the STATS file KSDS build program                       */
'OSRUN DMSTKEY'

EXIT
```

Displaying the Status of the Statistics Logging Facility

The INQUIRE STATISTICS command displays the current status of the CONNECT:Direct statistics logging facility.

INQUIRE STATISTICS Command Format

The INQUIRE STATISTICS command has the following format.

Label	Command	Parameters
(optional)	**INQuire STATistics**	

The INQUIRE STATISTICS command has no required parameters.

Batch Interface Use of INQUIRE STATISTICS

To issue this command from the Batch interface, perform these steps:

1. Place the INQUIRE STATISTICS commands in a batch job stream as shown in the *CONNECT:Direct for VM/ESA User's Guide.*
2. Verify that CONNECT:Direct is running.
3. Submit the job and verify the results of your inquiry.

Note: Set the fifth character of the DMBATCH output parameter specification to **Y** to print the result of the command that is in the temporary data set.

IUI Use of INQUIRE STATISTICS

To issue this command through the IUI, perform the following steps:

1. Access the Inquire DTF Internal Status screen by selecting **INQ** from the Administrative Options Menu.
2. Type **ISTA** and press **ENTER** to display the status of the statistics logging facility.

Example of the INQUIRE STATISTIC Status Report

The INQUIRE STATISTIC Status Report includes information including:

- the configuration of the statistics file pair list
- the active file pair
- file percentage utilizations
- date and time ranges in the files
- additional information about the statistics facility

A partial sample report of the statistics logging facility follows.

```
============================================================
node.name         *INQ STATS* DATE: mm/dd/yyyy  TIME: hh:mm:ss
============================================================

Status       => Enabled               Sec. Name  => USER01
Return Code  => 0                     Message ID => SSTW000I
Last "S2"    => 16:25:12              Que Wait   => No
Dsn Base     => USER01.R320.STATS     Q threshld => 40
                                      Q low-watr => 91
Excluded     => MC
*****************  F I L E  P A I R  #01  ********************
Status       => Active
Start Date   => 01/20/1998            End Date   => 01/20/1998
Start Time   => 15:25:29              End Time   => 15:51:35
KSDS Size    => 204800                ESDS CIS   => 4096
ESDS Size    => 2457600               ESDS Loc.  => 8684
Reset Pend.  => No                    Arch. Wait => No
Last Switch  => SWITCH.TIME event     Sel. Count => 0
KSDS Status  => Alloc, Open           ESDS Stat. => Alloc, Open
     L-cmd   => ENDREQ                     L-cmd => ENDREQ
     L-msg   => SVS0000I                   L-msg => SVS0000I
     L-rc    => 0                          L-rc  => 0
     L-fdb   => 0                          L-fdb => 0
                      .
                      .
                      .
```

Displaying the Statistics Archive File Directory

The INQUIRE STATDIR command displays the CONNECT:Direct statistics archive file directory.

INQUIRE STATDIR Command Format

The INQUIRE STATDIR command has the following format and associated parameters.

Label	Command	Parameters
(optional)	**INQuire STATDIR**	STARTT = ([date | day] [,hh:mm:ssXM])

Required Parameters

There are no required parameters for the INQUIRE STATDIR command.

The following are descriptions for the optional parameters used with the INQUIRE STATDIR command.

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

specifies that directory display is to begin with the first archive file created after the designated starting date and time. The date or day and time are positional parameters. If you do not specify the date or day, a comma must precede the time. If you omit this parameter, the display will begin with the first directory entry.

date specifies to display the first archive file created after this date. You can specify the day (dd), month (mm), and year (yy for 2-digit year and yyyy for 4-digit year).

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

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

To specify the order of a Gregorian date, you *must* define the DATEFORM initialization parameter. If you do not specify the DATEFORM parameter, CONNECT:Direct for VM/ESA defaults to the MDY format.

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

If you specify DATEFORM=MDY, CONNECT:Direct interprets the date format as:

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

If you specify DATEFORM=DMY, CONNECT:Direct interprets the date format as:

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

If you specify DATEFORM=YMD, CONNECT:Direct interprets the date format as:

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

If you specify DATEFORM=YDM, CONNECT:Direct interprets the date format as:

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

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

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

If you only specify date, the time will default to 00:00.

day specifies to display the first archive file created after this day of the week. Valid names include MOnday, TUesday, WEdnesday, THursday, FRiday, SAturday, and SUnday. You can also specify YESTER to search for archive files created after yesterday or TODAY to search for the archive files created after today.

hh:mm:ssXM requests the first archive file created after this time of day, specified in hours (hh), minutes (mm), and seconds (ss). XM can be set to AM or PM. You can express the time of day using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are 00:00–24:00. If you use the 12-hour clock, 1:00 hours could be expressed as 1:00AM, and 13:00 hours could be expressed as 1PM.

If you do not use either AM or PM, CONNECT:Direct assumes the 24-hour clock. You do not need to specify minutes and seconds. You can also specify NOON, which will display files created after noon, or MIDNIGHT, which will display archive files created after midnight. The default for the time is 00:00:00, the beginning of the day.

If you specify time of day but not date, the output will show the first available entry in the archive directory for files created after that time of day. Archive files from all later times and dates will display up to and including the stop time.

Batch Interface Use of INQUIRE STATDIR

To use this command from the batch interface, perform these steps:

1. Place the INQUIRE STATDIR command in a batch job stream as presented in the *CONNECT:Direct for VM/ESA User's Guide*.

2. Submit the job while CONNECT:Direct is running.

3. Verify the results.

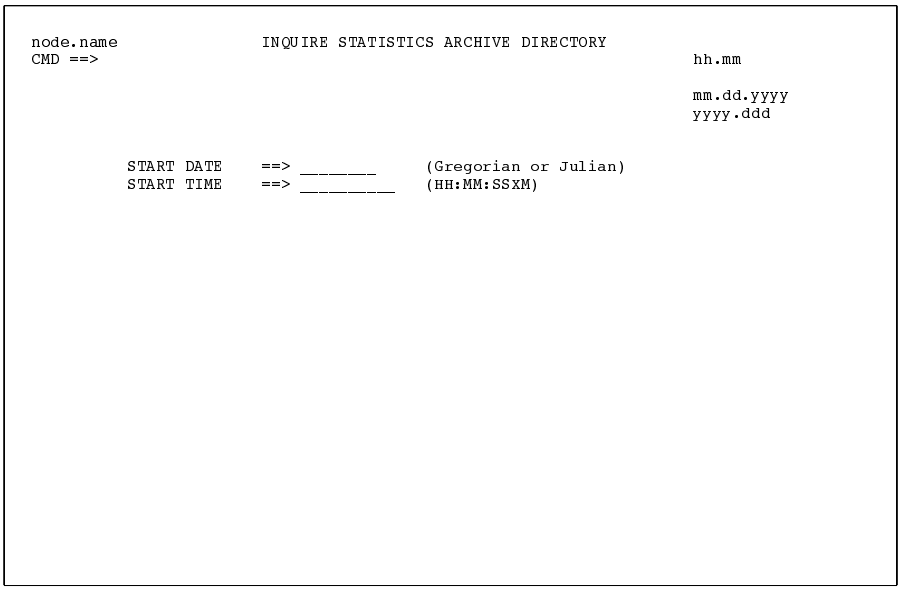
You must set the fifth character of the DMBATCH output parameter specification to Y to print the result of the command that is in the temporary data set.

IUI Use of INQUIRE STATDIR

To issue this command through the IUI, perform the following steps:

1. Select option **INQ** from the Administrative Options Menu. The Inquire DTF Internal Status screen displays.

2. Type **IDIR** and press **ENTER** to display the directory.



3. Type the beginning date and time to limit the display for the INQUIRE STATDIR command.

4. Verify the results of the inquiry. The following figure shows a partial sample report.

```
================================================================
node.name         *INQUIRE STATDIR*  DATE:  mm.dd.yyyy  TIME: hh:mm:ss
================================================================

Archival DSN:             USER01.STT.ARCHSTAT.G0008V00
Archival Notification:    03/02/1995  95.061  00:01:28
Oldest Record:            03/01/1995  95.060  00:00:06
Newest Record:            03/01/1995  95.060  23:59:54

Archival DSN:             USER01.STT.ARCHSTAT.G0009V00
Archival Notification:    03/03/1995  95.062  00:01:35
Oldest Record:            03/02/1995  95.061  00:00:11
Newest Record:            03/02/1995  95.061  23:59:45
                               .
                               .
                               .
```

Switching the Statistics File Pair

The STATISTICS SWITCH command initiates a statistics file pair switch. The currently active file pair closes, and logging continues on the next file pair in sequence. This command provides a means of initiating a file pair switch at any given time. Otherwise, switching occurs when the active file pair fills, or when a time of day specified in the STAT.SWITCH.TIME initialization parameter occurs.

STATISTICS SWITCH Command Format

The STATISTICS SWITCH command has the following format.

Label	Command	Parameters
(optional)	**STATistics SWITCH**	

There are no required parameters for the STATISTICS SWITCH command.

Batch Interface Use of STATISTICS SWITCH

To issue this command from the batch interface, perform these steps:

1. Place the STATISTICS SWITCH commands in a batch job stream as shown in the *CONNECT:Direct for VM/ESA User's Guide*.
2. Verify that CONNECT:Direct is running.
3. Submit the job.
4. Verify the results.

IUI Use of STATISTICS SWITCH

To issue this command through the IUI, perform the following steps:

1. Verify that CONNECT:Direct is running.
2. Access the Statistics Command screen by selecting **STAT** from the Administrative Options Menu.
3. Select option **FS** on the panel to initiate the file pair switch.
4. Verify the results of the file pair switch.

Recording Statistics for Specific Record Types

The STATISTICS ON/OFF command enables and disables recording of specific statistics record types. When you initialize the DTF, CONNECT:Direct enables the recording of all record types unless you specify the STAT.EXCLUDE initialization parameter. Use the INQUIRE STATISTICS command to find out which types are currently disabled.

STATISTICS ON/OFF Command Format

The STATISTICS ON/OFF command has the following format and associated parameters.

Label	Command	Parameters
(optional)	**STATistics** **ON** | **OFF**	TYPE = (record type list)

Required Parameters

The following parameter is required for the STATISTICS ON/OFF command.

TYPE

specifies the list of statistics record types whose recording is to be enabled or disabled. Use the 2-character identifier to specify record types. A list of these identifiers is given in the table on page 8-3.

Batch Interface Use of STATISTICS ON/OFF

To issue the STATISTICS ON/OFF command from the Batch interface, perform the following steps:

1. Place your commands in a batch job stream as shown in the *CONNECT:Direct for VM/ESA User's Guide*.

2. Verify that CONNECT:Direct is running.

3. Submit the job and verify the results.

IUI Use of STATISTICS ON/OFF

To issue the STATISTICS ON/OFF command through the CONNECT:Direct, perform the following steps:

1. Access the Statistics Command screen by selecting the **STAT** option from the Administrative Options Menu.

2. Select option **EN** to enable logging or option **DI** to disable logging.

3. Supply the list of affected record identifiers in the area provided, and press **ENTER.**

WARNING: Excluding record types may make problem analysis and resolution more difficult.

The use of the STATISTICS ON/OFF command to exclude Statistics record logging should be done prudently, because some types of records are critical for problem diagnosis. Do not exclude the following record types:

- CT Copy Termination
- PS Process Submit
- PT Process Termination
- RJ Run Job
- RT Run Task
- SW Submit within Process
- WO Write to Operator

Other record types are usually less critical and may be excluded.

Notifying CONNECT:Direct of Statistics File Archival

The STATISTICS ARCHIVED command notifies CONNECT:Direct that a statistics file has been archived. This notification allows the system to erase and overwrite the file with new records.

When you specify STAT.ARCH.CONFIRM=YES in the DTF initialization parameters, CONNECT:Direct cannot reuse a statistics file pair until it receives confirmation that the archive is complete.

In addition, The STATISTICS ARCHIVED command provides a means of sending this notification. Ordinarily it is sent by the DMSTARRT utility after the archive is done by a CONNECT:Direct COPY Process, or by the DMSTARBT utility after the archive is done by a batch step.

STATISTICS ARCHIVED Command Format

The STATISTICS ARCHIVED command has the following format and associated parameters.

Label	Command	Parameters
(optional)	**STATistics ARCHived**	file pair number

Required Parameters

The following parameter is required for the STATISTICS ARCHIVED command.

file pair number

specifies a number from 1–20 that identifies the statistics file for which archive notification is to be sent. This is given as the relative number of the file pair in the file pair list. The first pair in the list is file pair number 1.

Batch Interface Use of STATISTICS ARCHIVED

To issue the STATISTICS ARCHIVED command through the Batch interface, perform the following steps:

1. Place your commands in a batch job stream as shown in the *CONNECT:Direct for VM/ESA User's Guide.*

2. Verify that CONNECT:Direct is running.

3. Submit the job and verify your results.

IUI Use of STATISTICS ARCHIVED

To issue the STATISTICS ARCHIVED command through the CONNECT:Direct IUI, perform the following steps:

1. Access the Statistics Command screens by selecting the **STAT** option from the Administrative Options Menu.

2. Select option **CF**, supply the number of the file pair for notification of archival, and press **ENTER**.

Chapter 11

Using the Program Interface

This chapter covers the following topics:

- Writing a User Application Program
- API High-Level Interface (DMCHLAPI)
- How DMCHLAPI Interprets Parameters
- Examples of DMCHLAPI Parameters and Calling Sequences
- Sample EXEC for Executing a CONNECT:Direct Program

Writing a User Application Program

The two main CONNECT:Direct components are the Data Transmission Facility (DTF) and the Application Program Interface (API).

- The DTF performs actual transmission of data.
- The API is the set of rules that govern the interchange of requests and responses between the DTF and a CONNECT:Direct application program.

Application Interface Program

The high-level application interface program, DMCHLAPI, serves as a communications vehicle between the API and any application program that supplies CONNECT:Direct command strings for processing in a batch environment. You can write an application program following the rules described in the following sections.

CONNECT:Direct Command Strings

You can also process CONNECT:Direct command strings in a batch environment through the CONNECT:Direct batch interface program DMBATCH. Both DMCHLAPI and DMBATCH can be found in LOADLIB distribution file. For more information on using DMBATCH, refer to the *CONNECT:Direct for VM/ESA User's Guide*.

Writing a User Application Program

User-written applications can be designed to interface with CONNECT:Direct. The applications can be written in any computer language, including PL/1, Assembler, and COBOL.

A sample of an Assembler language user-written application, called ASMSAMP, can be found in the ASSEMBLE distribution library.

A sample of a PL/1 language user-written program, called PLISAMP, can be found in the ASSEMBLE distribution library.

API High-Level Interface (DMCHLAPI)

The high-level CONNECT:Direct interface program, DMCHLAPI, communicates with the API through a control block interface called the User Interface Control Block (UICB).

DMCHLAPI works in the following sequence:

1. DMCHLAPI accepts CONNECT:Direct command strings as input from an application program and passes the strings to the API.

2. The user-written application requests DMCHLAPI to perform output formatting routines after returning from the API. These formatting routines display information about the CONNECT:Direct command that just completed processing.

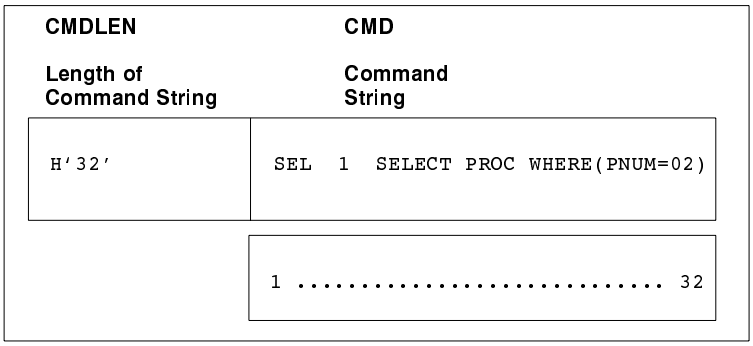
3. After execution of each command, DMCHLAPI issues a return code reflecting the status of API communications.

Completing the Required Parameters

DMCHLAPI requires three parameters to be passed on every invocation from an application program. The parameters are:

CMDLEN

(first parameter) points to a variable-length character string that contains the string length in the first halfword and the command text in the remainder of the string. The command string format follows.



UICB@

(second parameter) points to a fullword pointer that is initialized at Signon and cleared by the Signoff command.

OUTSPECS

(third parameter) points to a seven-character string, containing the output format specifications. The output is written to the filename defined by DMPRINT. Each specification is one character long. The following table lists the valid values and meaning of each specification.

Field	Values	Meanings
1	Y	Displays the command string that was executed.
	N	Does not display the command string that was executed.
2	Y	Displays the API return code and message ID.
	N	Does not display the API return code and message ID.
3	S	Displays the short message text when there is a return code of zero from the API.
	L	Displays long message text when there is a return code of zero from the API.
	N	Does not display message text when there is a return code of zero from the API.

(continued)

Field	Values	Meanings
4	S	Displays the short message text when there is a nonzero return from the API.
	L	Displays long message text when there is a nonzero return code from the API.
	N	Does not display message text when there is a return code greater than zero from the API.
5	Y	Displays the data that was generated in the temporary file.
	N	Does not display the data that was generated in the temporary file.
6	Y	Displays the string that identifies UICB fields which the Extract feature will return information about.
	N	Does not display the string that identifies UICB fields which the Extract feature will return information about.
7	Y	Displays a dump of the area that received output from the Extract feature.
	N	Does not display a dump of the area that received output from the Extract feature will return information about.

All output generated as a result of these specifications is routed to a DLBL NAME of DMPRINT. No output is created if the DLBL NAME is not present.

The following example shows the most common specifications for this parameter.

```
OUTSPECS  DC  C'YYSLYNN'
```

Positions 6 and 7 are not used by DMCHLAPI unless the extract feature of DMCHLAPI is used, as explained in the next section.

Using the UICB Extract Feature

Use the extract feature if the return code from the processed CONNECT:Direct command is wanted within the user program. This allows for extraction of certain UICB fields after CONNECT:Direct command execution.

To activate the optional extract feature, the program must pass the following additional parameters to DMCHLAPI: EXTSTRLN, EXTAREA, EXTRC, EXTMSG, and INVALKEY.

The following figure shows how the passed parameters look in a sample Assembler program.

```
*********************************************************************
**      THE PARAMETERS WOULD BE DEFINED AS FOLLOWS:                **
*********************************************************************

CMDLEN  DS      H
CMD     DS      CL1022
UICB@   DS      A
OUTSPECS        DC      C'YYSLYYY'
EXTSTR  DC      C'UITMPDDN UIESF UIPROC# UIMSGID'
EXTAREA DS      0CL24
**OUTPUT FROM THE UICB EXTRACT**
EXTMPDDN        DS      CL8
**VALUE OF UITMPDDN    **
EXESF   DS      F
**VALUE OF UIRTNCD     **
EXPROC# DS      XL4
**VALUE OF UIPROC#     **
EXMSGID DS      CL8
**VALUE OF UIMSGID     **
EXTRC   DS      F
EXTMSG  DS      CL8
INVALKEY        DS      CL8

*********************************************************************
**  NOTE THAT THE EXTRACT STRING ITSELF IS 30 BYTES LONG, BUT THE   *
**  DUMP OF ITS EXTRACT WILL USE 31 BYTES                           *
**                                                                  *
**  INSIDE THE MAIN BODY OF THE PROGRAM, THE CALL TO DMCHLAPI WOULD *
**  LOOK LIKE:                                                      *
**                                                                  *
*********************************************************************
        CALL    DMCHLAPI,(CMDLEN,UICB@,OUTSPECS,EXTSTRLN,EXTAREA,  *
                EXTRC,EXTMSG,INVALKEY),VL
```

The following is a list of optional parameters:

EXTSTRLN

(fourth parameter) is the UICB extract string. The user-supplied application must set up an extract string identifying the UICB fields to be extracted. The string consists of a halfword length field containing the length of the extract string (excluding the itself halfword length field), followed by the first UICB field to be extracted, a space, the second UICB field to be extracted, a space, and so on.

A sample extract string follows:

H'32'	UITMPDDN UIRTNCD UIPROC# UIMSGID

Only the UICB fields listed in the *UICB Field Listing* section on page 11-6 can be specified in the extract string. If an invalid keyword is encountered in the string, the extract routine will terminate execution, and no more

information will be extracted. The application will be informed of the error by means of the EXTRC, EXTMSG, and INVALKEY parameters. If field 6 of the OUTSPECS parameter has been set to **Y**, the extract string will be written to the file defined by DMPRINT.

EXTAREA

(fifth parameter) is where the UICB extracted information is placed. The application is responsible for ensuring that the area is large enough to accommodate the information requested by the extract string. If field 7 of OUTSPECS has been set to Y, the extract string will be written to the file defined by DMPRINT.

The following table shows how to calculate required storage for the extract area necessary for the extract string in the previous figure.

UICB Field	Data Definitions	Required Storage
UITMPDDN	Character length of 8	8 bytes
UIRTNCD	Fullword	4 bytes
UIPROC#	Fullword	4 bytes
UIMSGID	Character length of 8	8 bytes

EXTRC

(sixth parameter) is a four-byte binary field containing the extract feature return code.

EXTMSG

(seventh parameter) is an eight-character field containing an eight-character message ID from the extract feature.

INVALKEY

(eighth parameter) is an eight-character field containing an invalid extract string keyword. DMCHLAPI relays the location of the extract string error to the application in this parameter. This field will contain the invalid item in the extract string if EXTRC is nonzero.

UICB Field Listing

The following table shows the valid UICB fields. Specifying these fields in the EXTSTRLN parameter will result in feedback in the extract area, defined by the EXTAREA parameter.

Exact definitions can be found in the DMUICB macro found in SAMPLIB. The field names, which appear boldfaced, are explained further on page 11-10 in *DMCHLAPI Return Codes*.

In following table, the Control Block Builder Syntax Error Work Areas have a value only when appropriate. For the Boolean Flags, output is **Y** or **N** indicating bit is *on* or *off*.

Name	Type	Length	Description
UIRCBLNG	halfword	2	The length of the control block
UIDESCR	character	16	UICB identifier
UITCA	address constant	4	Task Control Area
UIBRCB	address constant	4	Batch Region Control Block
UIDYNCB	address constant	4	Dynamic Allocation Control Block
UITPCB	address constant	4	Text Parser Control Block
UILEVEL	fullword	4	Modal level counter
UITMPDCB	address constant	4	Temporary file DCB
UIMSGCB	address constant	4	Message Control Block
UIUNODE	character	16	User node ID
UIUID	character	8	Userid
UIPSWD	character	8	Signon password
UINPSWD	character	8	New Signon password
UITMPVOL	character	6	Volume serial number of temporary file if not specified by user
UITMPDDN (VM)	character	8	DDNAME used for temporary DSN
UIUSRTYP	character	1	User Type (operator, administrator, user)
UIAPPLID	character	8	VTAM logon ID
UIRTNCD	fullword	4	API return code
UIMSGID	character	8	API message ID
UILNODE	character	16	Name of node that is “local” to DTF
UILPP	halfword	2	Lines per page for printed output
UITMPLNG (VM)	halfword	2	Length of temporary filename
UITMPDSN (VM)	character	44	Temporary file filename
UIPUBLNG	halfword	2	Length of Process Library name
UIPUBDSN	character	44	Process Library name
UIMSGLNG	halfword	2	Length of message library name
UIMSGDSN	character	44	Message library name
UINETMAP	character	64	Network Map file name

(continued)

Name	Type	Length	Description
UIPROC#	fullword	4	Process number from submit
UIDSPLY	address constant	4	Address of SCDSPLY
UISTRING	address constant	4	Address of SCSTRING
**********	**********		START OF CONTROL BLOCK BUILDER SYNTAX ERROR WORK AREA
UILABL#	halfword	2	Length of CONNECT:Direct label
UILABL	character	8	CONNECT:Direct label
UICMD1#	halfword	2	Length of first word in CONNECT:Direct command
UICMD1	character	8	First word in CONNECT:Direct command
UICMD2#	halfword	2	Length of second word in CONNECT:Direct command
UICMD2	character	8	Second word in CONNECT:Direct command
UIKLST#	halfword	2	Length of keyword that starts a list
UIKLST	character	8	Keyword that starts a list
UIKEYW#	fullword	4	Length of keyword in list before error
UIKEYW	character	8	Last keyword in list before an error
UIPARM#	halfword	2	Length of parameter associated with UIKEYW
UIPARM	character	8	Parameter in error associated with UIKEYW
UIERRM1#	halfword	2	Length of UIERRM1 string
UIERRM1	character	64	All of the above work areas resolved into a string
UIERRM2	character	64	Msg ID and text for parsing error
**********	**********		END OF CONTROL BLOCK BUILDER SYNTAX ERROR WORK AREA
**********	**********		START OF BOOLEAN FLAGS
UIERRON	character	1	Indicates message in UIERRM1
UIERRLAB	character	1	Indicates something in UILABL
UIERRCM1	character	1	Indicates something in UICMD1
UIERRCM2	character	1	Indicates something in UICMD2
UIERRSCP	character	1	Indicates parsing error
UIERRC10	character	1	open delimiter after command keyword, for example, “IF (”
UIERRC20	character	1	Indicates open delimiter after second command keyword
UIERRG	character	1	Indicates VTAM error msg in UIERRM1
UIERRLST	character	1	Indicates something in UIKLST
UIERRLOP	character	1	Indicates open delimiter after UIKLST
UIERRLCL	character	1	Indicates close delimiter after UIKLST
UIERRLEQ	character	1	Indicates equal sign after UIKLST
UIERRLCM	character	1	Indicates comma after UIKLST

(continued)

Name	Type	Length	Description
UIERRLSP	character	1	Indicates space after UIKLST
UIERRKEY	character	1	Indicates something in UIKEYW
UIERRKOP	character	1	Indicates open delimiter after UIKEYW
UIERRKCL	character	1	Indicates close delimiter after UIKEYW
UIERRKEQ	character	1	Indicates equal sign after UIKEYW
UIERRKCM	character	1	Indicates comma after UIKEYW
UIERRKSP	character	1	Indicates space after UIKEYW
UIERRBPC	character	1	Indicates close delimiter before a parameter in a list
UIERRPRM	character	1	Indicates something in UIPARM
UIERRPOP	character	1	Indicates open delimiter after UIPARM
UIERRPCL	character	1	Indicates close delimiter after UIPARM
UIERRPEQ	character	1	Indicates equal sign after UIPARM
UIERRPCM	character	1	Indicates comma after UIPARM
UIERRPSP	character	1	Indicates space after UIPARM
UIF1SUBM	character	1	Indicates a Process was submitted
UIGOTDSN	character	1	Indicates found temporary file name as Signon command parameter
UITFILE	character	1	Indicates data was generated into temporary file
UIEOF	character	1	Indicates reached EOF of Process file
UIMODAL	character	1	Indicates modal statement processed
UITFILEX	character	1	Reserved
UIMASTER	character	1	Indicates this UICB is master
UIRECON	character	1	Indicates a reconnect attempted
UIINACT	character	1	Indicates VTAM session for this UICB failed
UILOCAL	character	1	Indicates local node session
UIESF	character	1	Indicates ESF=YES on Signon command
UIZOPSWD	character	1	Indicates blank password on Signon
**********	**********		END OF BOOLEAN FLAGS
UIRLSE#	halfword	2	Current release, version and mod level
UIPUF#	halfword	2	Current PUF level
UI@MASTR	address constant	4	Master (user's) UICB
UI@ACTIV	address constant	4	Currently active UICB
UI@FPTR	address constant	4	Next UICB
UI@BPTR	address constant	4	Previous UICB
UIALOTYP	character	8	Allocation type for temporary file
UIALOPRI	character	8	Allocation of prime space for temporary file

(continued)

Name	Type	Length	Description
UIALOSEC	character	8	Allocation of secondary space for temporary file
UIALOUNI	character	8	Allocation unit for temporary file
UIALOVOL	character	8	Allocation of volume serial number for temporary file
UIFOLD	character	3	Fold to uppercase if YES UIPRTALC
UIPRTALC	character	80	User-defined CONNECT:Direct print destination
UIPACCT#	halfword	2	Length of PNODE accounting data
UIPACCT	character	256	PNODE accounting data text
UISACCT#	halfword	2	Length of SNODE accounting data
UISACCT	character	256	SNODE accounting data text

How DMCHLAPI Interprets Parameters

As noted earlier, either the first three or all eight parameters are specified by DMCHLAPI. If an incorrect number of parameters is passed, DMCHLAPI issues an error message and assigns a return code of 20. Processing cannot occur during this time.

DMCHLAPI Return Codes

The return codes in the following table reflect the status of the DMCHLAPI communications with the API. These codes only reflect whether DMCHLAPI could process the request and pass the command to the DTF. They do not reflect the completion status of the CONNECT:Direct command.

Return Code	Meaning
00000000	The command executed normally.
00000004	Signon to the master session failed, but the Extended Submit Feature (ESF) environment was established.
00000008	A non-ESF command attempted in an ESF environment.
0000000C	A session was lost in a multiple session environment.
00000010	The master session was lost.
00000014	The master session was signed off successfully.
00000018	The master session signon failed, and there is no ESF ability.

(continued)

Return Code	Meaning
0000001C	A non-master signon failed.
00000020	DMCHLAPI received an invalid number of input parameters.
00000024	The output specifications included an invalid parameter.
00000028	An invalid pointer to the UICB was passed to DMCHLAPI.

Extract Function Fields of Interest

The return feedback from the DTF regarding the completion status of the CONNECT:Direct command must be obtained through the Extract function. Return codes received through the function inform the user of an error or information generated by the requested command.

Three particular fields of interest in the Extract function, applicable to DMCHLAPI return codes, are as follows:

UIRTNCD
: lists return codes. UIRTNCD is set on completion of every command processed by the DTF.

UIMSGID
: lists message IDs. UIMSGID is set on completion of every command processed by the DTF.

UIPROC#
: identifies process numbers. This is set for every successful Submit command.

Refer to the *UICB Field Listing* section beginning on page 11-6 for the complete list of UICB fields.

Examples of DMCHLAPI Parameters and Calling Sequences

These examples show how the required and optional parameters and calling sequences might appear in Assembler and PL/1 applications.

DMCHLAPI Parameters for an Assembler Program

The following is an example showing how the required parameters and calling sequences appear in a sample Assembler program.

```
*****************************************************************
*    THE DATA DEFINITIONS ARE AS FOLLOWS:                       *
*****************************************************************
CMDLEN     DS   H                    LENGTH OF COMMAND STRING
CMD        DS   CL1022               COMMAND STRING
UICB@      DS   A                    POINTER TO UICB
OUTSPECS   DC   C'YYSLYNN'           OUTPUT SPECIFICATIONS

*****************************************************************
*                                                               *
*    THE DMCHLAPI INVOCATION WOULD BE:                          *
*                                                               *
*****************************************************************/
 CALL DMCHLAPI,(CMDLEN,UICB@,OUTSPECS),VL
```

DMCHLAPI Parameters for a PL/1 Program

The following is an example showing how the required parameters and calling sequences appear in a sample PL/1 program.

```
*****************************************************************
*    THE DATA DECLARATIONS ARE AS FOLLOWS:                      *
*****************************************************************
DECLARE
CMD_PARM              FIXED BINARY (31)  BASED (R),
UICB_PARM      FIXED BINARY (31)  INITI (0'B'),
OUTSPEC_PARM   CHAR (7)  INIT ('YYSLYNN'),
CMD            CHAR (1022)  VARYING,
R              POINTER;

*****************************************************************
*                                                               *
*    THE DMCHLAPI INVOCATION WOULD BE:                          *
*                                                               *
*****************************************************************/
R=ADDR(CMD) ;
CALL DMCHLAPI(CMD_PARM, UICB-PARM, OUTSPEC_PARM);
```

The following example shows how the passed optional parameters would look in a sample Assembler program.

```
*********************************************************************
**      THE PARAMETERS WOULD BE DEFINED AS FOLLOWS:                **
*********************************************************************

CMDLEN   DS   H
CMD      DS   CL1022
UICB@    DS   A
OUTSPECS      DC   C'YYSLYYY'
EXTSTR   DC   C'UITMPDDN UIESF UIPROC# UIMSGID'
EXTAREA  DS   OCL24    **OUTPUT FROM THE UICB EXTRACT               **
EXTMPDDN DS   CL8      **VALUE OF UITMPDDN                          **
EXESF    DS   F        **VALUE OF UIRTNCD                           **
EXPROC#  DS   XL4      **VALUE OF UIPROC#                           **
EXMSGID  DS   CL8      **VALUE OF UIMSGID                           **
EXTRC    DS   F
EXTMSG   DS   CL8
INVALKEY DS   CL8

*********************************************************************
**      NOTE THAT THE EXTRACT STRING ITSELF IS 30 BYTES LONG,       *
**      BUT THE DUMP OF ITS EXTRACT WILL USE 31 BYTES.              *
**                                                                  *
**      INSIDE THE MAIN BODY OF THE PROGRAM, THE CALL TO            *
**      DMCHLAPI WOULD LOOK LIKE:                                   *
**                                                                  *
*********************************************************************/
CALL DMCHLAPI, (CMDLEN,UICB@,OUTSPECS,EXTSTRLN,EXTAREA,
     EXTRC,EXTMSG,INVALKEY),VL
```

Sample EXEC for Executing a CONNECT:Direct Program

The following example shows a sample EXEC that executes a CONNECT:Direct program that invokes DMCHLAPI.

```
*********************************************************************
*    FILE DEFINITION FOR OUTPUT                                     *
*********************************************************************
'FILEDEF DMPRINT DISK DMPRINT OUTPUT A'
'FILEDEF NDMCMDS DISK NDMCMDS OUTPUT A'

*********************************************************************
*    DLBL (NETMAP AND MSG FILE)                                     *
*********************************************************************
'DLBL DMNETMP E DSN NDM.VSAM.NETMAP(PERM)'
'DLBL DMMSGFL E DSN NDM.VSAM.MESSAGE(PERM)'

*********************************************************************
*    INVOKE USER PROGRAM                                            *
*********************************************************************/
'USERPROG'
```

Certain files must be defined using the Conversational Monitor Service (CMS) FILEDEF and DLBL commands before executing a CONNECT:Direct application program that calls DMCHLAPI. The following are definitions for these files.

DMPRINT

is optional, but recommended. All output from Select Process, Select Statistics, Select Type, and Select User commands are written to this file. Also, output as a result of the OUTSPECS specification is written to DMPRINT.

DMNETMAP

is required if no NETMAP keyword was specified on the SIGNON command. This is the name of the Network Map file that contains the names of all the nodes with which CONNECT:Direct will be communicating.

DMMSGFL

is required. The name of the Message file.

NDMCMDS

is optional. If allocated, all command strings are written to this file.

Chapter 12

DBCS Support

This chapter covers the following topics :

- Understanding DBCS
- Default translation tables
- Customizing the translation tables

Understanding DBCS

Some languages have too many symbols for all of their characters to be represented using single-byte codes. For example, the English language can be defined within a single-byte range from 1–256, or x'00' through x'FF'. Hanqeul, the language of Korea, and other ideographic languages contain several thousand characters. To create these coded character sets, two bytes are needed for each character.

The Double-byte Character Set (DBCS) support provides a mechanism for translating ASCII and EBCDIC DBCS data. DBCS support translates Single-byte Character Set (SBCS) and DBCS data in the form that is supported on the requested platform.

DBCS Representation

The representation for DBCS characters is different between operating system platforms. Specifically, a mainframe represents data in 8-bit EBCDIC code and a PC represents data in 7-bit ASCII code. For the mainframe environment, DBCS can be used exclusively within a file or be mixed with SBCS characters. Special character indicators exist to tell the difference between SBCS and DBCS characters. The special

character indicators are shift-out (SO) and shift-in (SI), or x'0E' and x'0F' respectively for IBM mainframes. Shift-out denotes shifting from SBCS to DBCS mode and shift-in denotes shifting from DBCS to SBCS mode. There is no need for SO/SI combinations if DBCS is exclusive within a file. For the PC, the SO/SI characters are not recognized. In this environment, DBCS is represented by setting the high order bit of the ASCII code. See the table on page 12-6 for proper mapping of DBCS characters by language.

Default Translation Tables

CONNECT:Direct provides translation support for the Korean Standard Code Page (KS5601), DBCS-PC Korean, Chinese 5550, and Chinese Big5 to host (EBC/NHC). These tables are provided in load module and source form. See the *CONNECT:Direct for VM/ESA COPY Statement* chapter in the *CONNECT:Direct Process Guide, Mainframe Products, Volume 2* for an explanation of how to use these tables with the SYSOPTS parameter in the COPY statement.

The following translation tables are in the CDV*nnnn* LOADLIB.

- **EBCXKSC** — host EBCDIC to ASCII KS5601
- **KSCXEBC** — ASCII KS5601 to host EBCDIC
- **EBCXKPC** — host EBCDIC to DBCS-PC Korean
- **KPCXEBC** — DBCS-PC Korean to host EBCDIC.
- **EBCXJIS** — host EBCDIC to Japanese International Standard
- **JISXEBC** — Japanese International Standard to host EBCDIC
- **NHCXBG5** — Chinese new host code to Chinese Big5
- **BG5XNHC** — Chinese Big5 to Chinese new host code
- **NHCXC55** — Chinese new host code to Chinese 5550
- **C55XNHC** — Chinese 5550 to Chinese new host code

The following source tables are on the DTF's 191 minidisk.

- **EBCXKSC** — EBCDIC to ASCII KS5601
- **KSCXEBC** — ASCII KS5601 to host EBCDIC
- **EBCXKPC** — host EBCDIC to DBCS-PC Korean
- **KPCXEBC** — DBCS-PC Korean to host EBCDIC.
- **EBCXJIS** — host EBCDIC to Japanese International Standard
- **JISXEBC** — Japanese International Standard to host EBCDIC
- **NHCXBG5** — Chinese new host code to Chinese Big5
- **BG5XNHC** — Chinese Big5 to Chinese new host code
- **NHCXC55** — Chinese new host code to Chinese 5550
- **C55XNHC** — Chinese 5550 to Chinese new host code

These translation tables are provided in source code format so, if needed, they can be copied or customized for your unique processing environment.

Customizing the Translation Tables

For proper translation of code pages other than the supplied translation tables, or if the supplied translation tables are not sufficient, a means of creating and updating translation tables is available. This is done by a preprocessor that takes simple batch input in a predefined format and creates output compatible with the assembler. The output can then be assembled and link-edited to produce a loadable translation table.

Input to the batch preprocessor consists of six main parameters and the END parameter. All input begins in column one. The following table defines the batch preprocessor parameters.

Parameter	Required	Default	Format	Definition
NAME	No	XLATE	8 characters	Table name information
TITLE	No	DBCS TRANSLATION TABLE	60 characters	Table title information
DEFAULT	No	0000	2 byte hex representation	Default translation character
RULES	No	80–FF	2 byte hex representation	Language rules
SBCS	No	Standard	2 byte hex representation	Single-byte character set translation table
DBCS	Yes	None	4 byte hex representation	Double-byte character set translation table
END	Yes	None		Terminates DBCS, SBCS, and RULES parameters

Required Parameters

The following parameters are required.

DBCS

is used to create the double-byte character set translation table. This table is used to translate all double-byte data during a file transfer. This parameter has no default and is required. The DBCS parameter data begins in column one and is terminated with the END statement.

This example shows the syntax for the DBCS parameter.

```
DBCS
f1f2,t1t2
END
```

f1 denotes the first byte of the FROM DBCS character.

f2 is the second byte of the FROM DBCS character.

t1 is the first byte of the TO DBCS character.

t2 is the second byte of the TO DBCS character.

END

is mandatory to terminate each of the following parameters:

- DBCS
- RULES
- SBCS

Optional Parameters

The following parameters are optional.

NAME=[tablename | XLATE]

is an 8-character parameter for displaying table information in batch format. NAME is optional and is for informational use only. If used, NAME must be the first parameter defined. If used with TITLE, NAME and TITLE must be the first two parameters defined. The default for NAME is XLATE.

The following example shows the syntax for the NAME parameter.

```
NAME=EBCXKSC
```

The EBCXKSC table is provided on the DTF's 191 minidisk.

TITLE=[title name | DBCS TRANSLATION TABLE]

is a 60-character parameter for displaying table information in batch format. TITLE is optional and is for informational use only. If used, TITLE must be the first parameter defined. If used with NAME, NAME and TITLE must be the first two parameters defined. The default for TITLE is DBCS TRANSLATION TABLE.

This example shows the syntax for the TITLE parameter.

```
TITLE=HOST EBCDIC TO ASCII KS5601 TRANSLATION
```

The EBCXKSC table is provided on the DTF's 191 minidisk

DEFAULT=nnnn

contains the hexadecimal representation you define as the replacement for invalid DBCS code points. This default character appears wherever a non-translatable character appears in the data being received. The default is 0000.

nnnn denotes the hexadecimal character defined to replace an invalid DBCS code point.

This example shows the syntax for the DEFAULT parameter.

```
DEFAULT=FFFF
```

RULES

is used to define what constitutes a double-byte character for the defined language. RULES is only used when receiving a file from a platform other than MVS. This is because the host cannot determine valid DBCS characters without language rules. The default is any character within the range of x'80' through x'FF', meaning CONNECT:Direct interprets any character within this range as the first byte of a DBCS pair. Both characters in the pair are translated to host DBCS. If specified, use the END statement to terminate the RULES parameter.

Valid language options for the RULES parameter are:

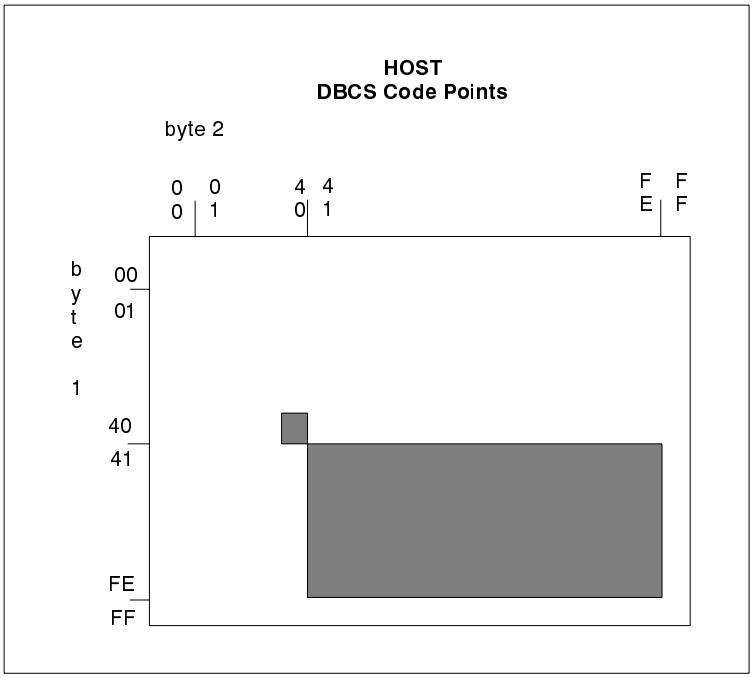
Language Option	Range
KS5601 (Korean Standard)	x'A1'–x'AC' x'B0'–x'FD'
KOREAN (Old Style)	x'81'–x'BF'
JAPANESE	x'81–x'9F' x'E0'–x'FC'
CHINESE (Traditional/Simplified and 5550)	x'81'–x'FC'
BIG5 (Chinese)	x'A4'–x'C6' x'C9'–x'F9'
x'01'–x'FF'	user selectable

The following example shows the syntax for the RULES parameter.

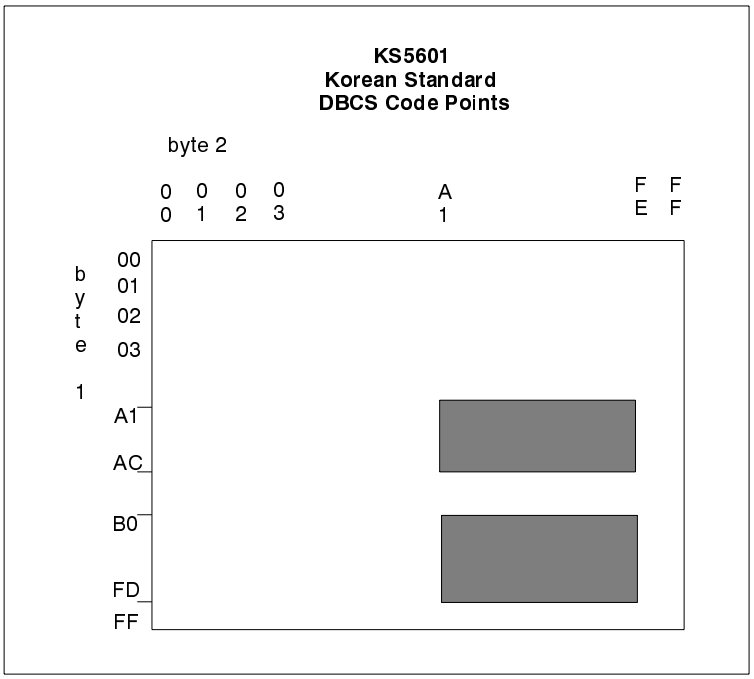
```
RULES
KS5601
END
```

The KS5601 language option is in the EBCXKSC table, which is provided on the DTF's 191 minidisk.

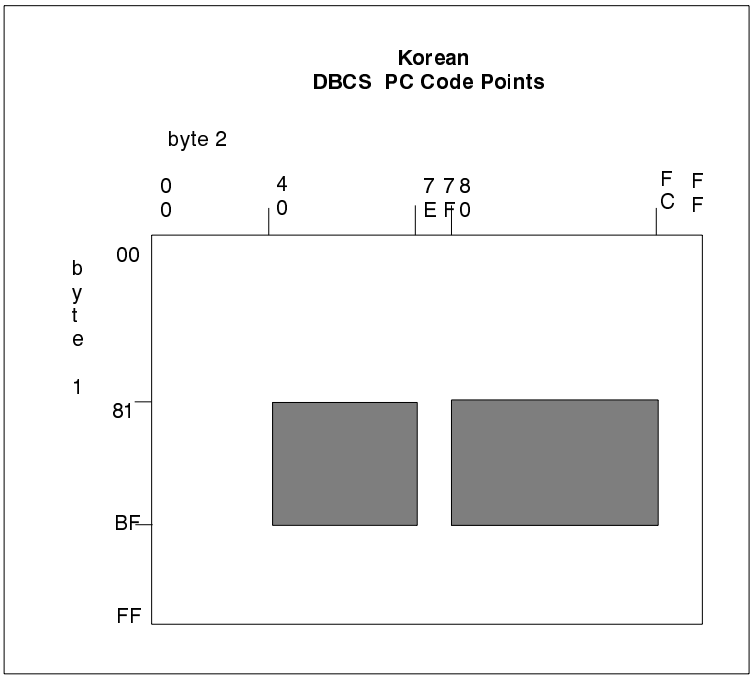
The following figure is a graphic representation of the CONNECT:Direct for VM/ESA hexadecimal DBCS code points.



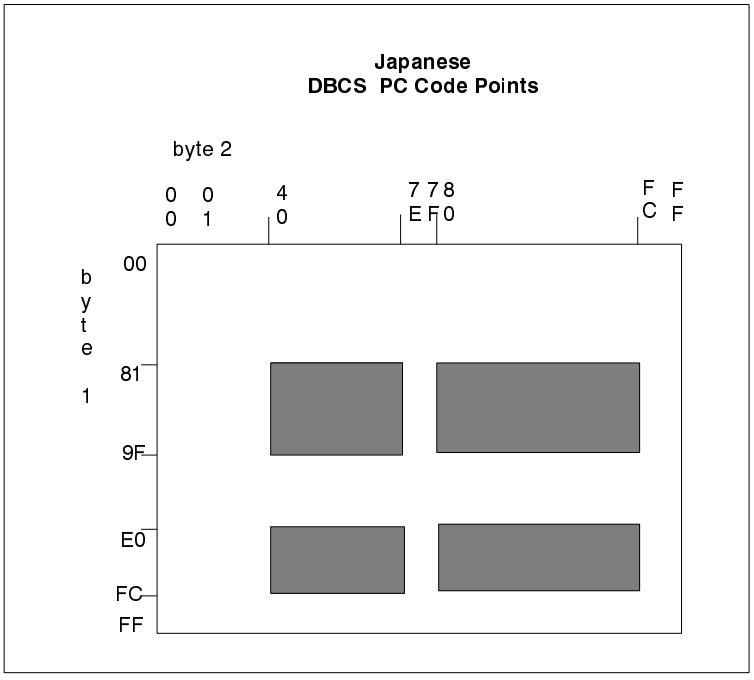
The following figure is a graphic representation of the Korean Standard (KS5601) hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 12-6.



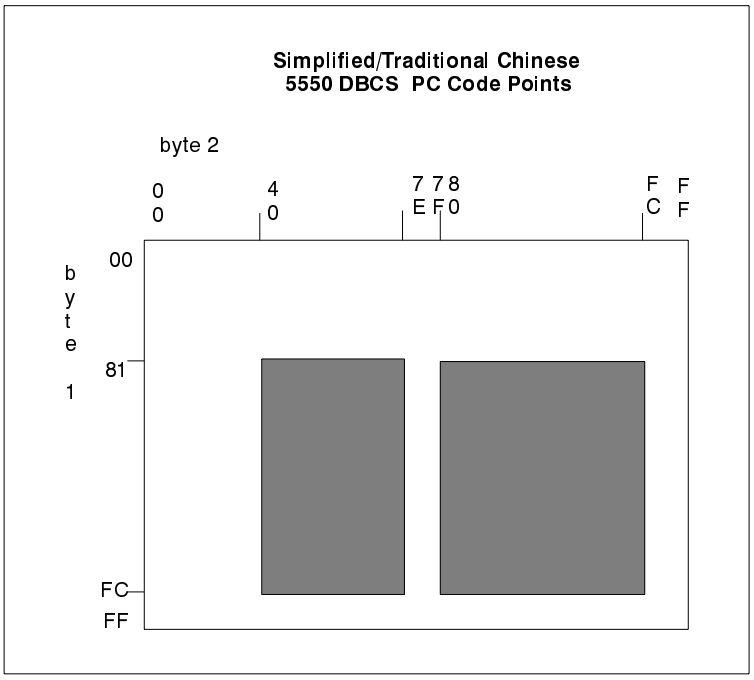
The following figure is a graphic representation of the Korean hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 12-6.



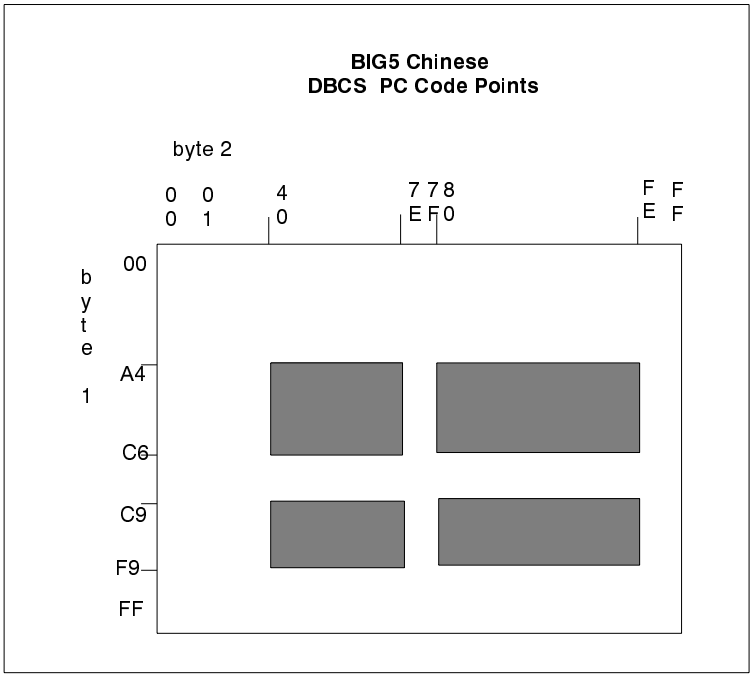
The following figure is a graphic representation of the Japanese hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 12-6.



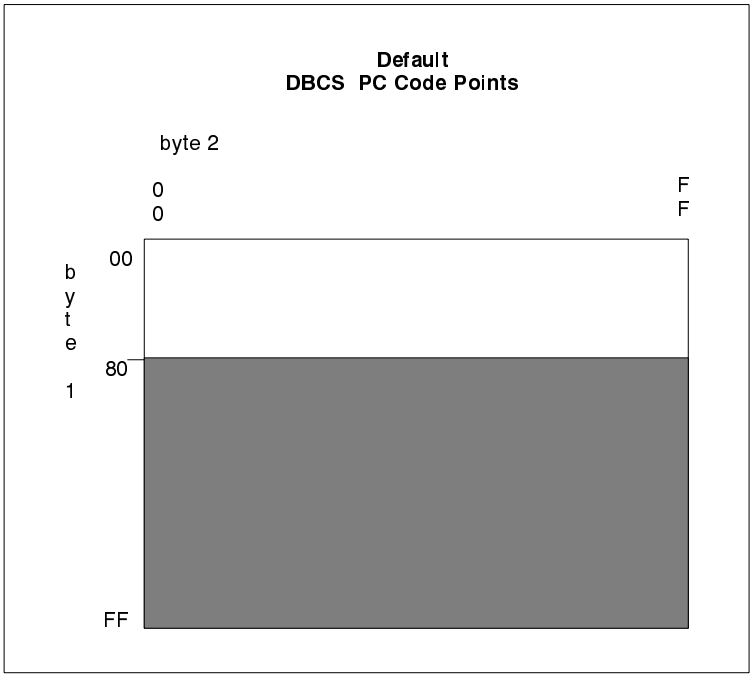
The following figure is a graphic representation of the Traditional Chinese hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 12-6.



The following figure is a graphic representation of the Chinese (BIG5) hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 12-6.



The following figure is a graphic representation of the default hexadecimal DBCS code points.



SBCS

creates the single-byte character set translation table. Use this table to translate all single-byte data during a file transfer. The default translation table provided when the parameter is not specified, translates all EBCDIC characters in the range of x'00' through x'FF' to its ASCII equivalent, within the range of x'00' through x'7F'. When receiving the file from a PC, the data is translated from ASCII to EBCDIC. Terminate the SBCS parameter with the END statement.

If SBCS is defined, all data must begin in column one and only one hexadecimal character pair is allowed per line.

This example shows the syntax for the SBCS parameter.

```
SBCS
ff,tt
END
```

ff denotes the FROM translation.

tt denotes the TO translation.

Comments

Comments allow you to include additional information in a batch preprocessor. Comments are available as a convenience to the user and do not affect CONNECT:Direct. The format for a comment is an asterisk (*) in column 1, followed by the comment. The following figure is a sample comment with an asterisk in column 1.

```
* DEFAULT=FFFF instead of 0000.
```

Using the RULES Parameter

The following example translates all characters as DBCS that adhere to KS5601 standard, or all characters that start with an x'A1' through x'AC' or x'B0' through x'FD'. Treat these characters as double-byte characters.

```
RULES
KS5601
END
```

The following example will translate all characters as DBCS that adhere to the customized table. Treat all characters that start with x'90' through x'94' or x'B0' through x'B4' as double-byte characters.

```
RULES
90
91
92
93
94
B0
B1
B2
B3
B4
END
```

Using the SBCS Parameter

The following example will translate x'C1' to x'41', x'C2' to x'42', x'C3' to x'43', and so on.

```
SBCS
C1,41
C2,42
C3,43
C4,44
C5,45
C6,46
END
```

Using the DBCS Parameter

A DBCS table can be extremely large and complex; therefore, the following examples are sample tables. The examples should be used as a reference only and will not successfully translate all characters.

The following example translates x'89A1' to x'B0ED', x'89A2' to x'B0EE', x'89A5' to x'B0EF', and so on to x'D37B' to x'C8F0'.

```
DBCS
89A1,B0ED
89A2,B0EE
89A5,B0EF
89A8,B0F0
89A9,B0F1
89AA,B0F2
89AB,B0F3
D375,C8EE
D377,C8EF
D37B,C8F0
END
```

Sample Preprocessor Input Data Stream

The following sample is the syntax for a preprocessor input data stream. The SBCS and DBCS tables are incomplete and would require many pages to produce a valid table.

```
NAME=MYTABLE
TITLE=SAMPLE TRANSLATION TABLE
RULES
80
81
82
83
84
END
SBCS
C1,41
C2,42
C3,43
C4,44
C5,45
C6,46
END
DBCS
89A1,B0ED
89A2,B0EE
89A5,B0EF
89A8,B0F0
89A9,B0F1
89AA,B0F2
89AB,B0F3
D375,C8EE
D77,C8EF
D37B,C8F0
END
```

DBCSBLD EXE to EXECUTE the Preprocessor

The DBCSBLD EXEC is located on the DTF's 191 minidisk. To execute the preprocessor, type the following command:

```
DBCSBLD tablename
```

The specified tablename in the previous command must reside on the 'A' mdisk. In addition, the tablename must be a DBCS filetype. Two output files are generated with the file types of:

- ASSEMBLE
- PRINT

Each use the file name specified on the request. You will need to assemble and link-edit the output file with the file type of ASSEMBLE to produce a load module on the DBCS table. The output file with the file type PRINT is the output generated by the preprocessor.

Appendix A

Initialization Parameters

This appendix presents the definitions and default values for the initialization parameters. The CONNECT:Direct initialization module processes the file that contains these parameters. The initialization parameters specify alternate values for various CONNECT:Direct parameters.

CONNECT:Direct processes initialization parameters during its startup. You specify the file name containing the initialization parameters as a parameter in the execute statement of the startup job stream.

The *CONNECT:Direct Quick Reference for MVS, VM/ESA, VSE/ESA, and MSP* also contains a table of the initialization parameters and their default values.

CONNECT:Direct Initialization Parameters

This section contains a description and the default value(s) for each initialization parameter.

ALLOC.CODES=(allocation errors)

specifies allocation errors for which CONNECT:Direct will retry the Process step as specified in the ALLOC.RETRIES and ALLOC.WAIT initialization parameters. These allocation errors are the Dynamic Allocation Interface Routine (DAIR) codes.

CONNECT:Direct has defined special error codes to enable retries for specific failures. These error codes are DSNR and TAPR.

When you attempt to send or receive a file, but cannot because the file is in use, CONNECT:Direct retries the Process (specified by the retry option) for the number of times specified, if you have coded the DSNR option.

TAPR indicates that a tape process is retried if the number of tape processes specified by MAX.TAPE is reached.

Allocation retries are controlled by the PNODE and supported for MVS, VM, and VSE platforms only. The following table lists the default allocation error codes and their meanings.

Code	Definition
020C	Exclusive use of shared file
0210	Allocated to another job
0218	Volume not mounted
0220	Volume not available
0234	One device required
0068	VM minidisk already linked read-only, if transferring with VM
0069	VM minidisk already linked read-write, if transferring with VM
006A	VM minidisk already linked read-write and read-only, if transferring with VM

Default: 020C 0210 0218 0220 0234 0068 0069 006A

ALLOC.RETRIES=number of retries

specifies the number of CONNECT:Direct retries of an allocation failure.

Default: 20

ALLOC.STORAGE=ABOVE | BELOW

specifies whether storage allocated by CONNECT:Direct for control blocks and data areas should be allocated above or below the 16 Mb line in a VM environment. This parameter allows you to keep storage areas below the 16 Mb line that may be referenced by user exits.

Because many of the VM modules and control blocks are in extended memory, any user exits must be able to receive control from a program in the extended area, address control blocks in the extended area, and

return control to programs in the extended area. Always reassemble any user exits against the new release.

Default: BELOW

ALLOC.WAIT=hh:mm:ss

specifies the amount of time that CONNECT:Direct waits between retries of an allocation failure.

Default: 00:03:00

ALLOCATION.EXIT=modname

specifies the name of the user-written program to interface with CONNECT:Direct. You can invoke the allocation exit prior to any allocation activity by CONNECT:Direct, thereby allowing the exit program to examine and modify information that CONNECT:Direct uses during the allocation process, such as the data set name (DSN) and destination name. In addition, you can set parameters to terminate a copy step before allocation takes place.

DMGALOEX is the supplied CONNECT:Direct sample exit. It is located on the DTF's 191 minidisk.

Default: No allocation exit

CKPT=nK | nM

allows automatic checkpointing of eligible files if no CKPT keyword is specified on the CONNECT:Direct COPY statement. (See the CKPT.MODE initialization parameter for further details of automatic checkpointing.) The value of **n** can be from 1–*999,999,999*. **K** means thousands of bytes; **M** means millions of bytes. CONNECT:Direct uses the value specified, rounded to the nearest block boundary, to determine when a checkpoint should be taken. The CKPT specification on the CONNECT:Direct Copy statement always overrides the CKPT initialization parameter value.

Default: No automatic checkpointing

CKPT.DAYS=number of days

specifies the number of days that checkpoint records stay in the Checkpoint file before automatic deletion during CONNECT:Direct initialization. The records can be left in the Checkpoint file if

transmission is interrupted and the Process is deleted without being restarted.

Default: 4

CKPT.MODE=(RECORD | BLOCK BLOCK | RECORD PDS | NOPDS NOPDS | PDS VSAM | NOVSAM VSAM | NOVSAM)

allows you to control checkpointing as it occurs in both record and block level transfers.

CONNECT:Direct transfers a file in record mode or block mode. Transferring in block mode is more efficient than record mode because the data is accessed one block at a time rather than record by record. However, record mode is necessary when the receiving data set is of a different block size than the sending data set.

The first two positional subparameters of the CKPT.MODE initialization parameter refer to transferring physical sequential (PS) files. The next two positional subparameters refer to transferring set of files. The final two positional subparameters refer to checkpointing VSAM files.

The first positional subparameter refers to transferring physical sequential (PS) files. It determines whether checkpointing is allowed when the CKPT parameter has been specified on the COPY statement. Valid options are RECORD and BLOCK. If a record level transfer is taking place, BLOCK does not allow record level checkpointing, even if a user has coded the request on the Copy statement. In this case, no checkpointing occurs. Because record level checkpointing could cause a significant amount of overhead, BLOCK might be a desirable choice. If a block level transfer is being performed, checkpointing requests coded on the COPY statement are honored. RECORD allows record level checkpointing. RECORD is the default for the first positional subparameter.

The second positional subparameter also refers to transferring physical sequential (PS) files. It determines what type of checkpointing occurs when automatic checkpointing is in effect. You enable automatic checkpointing by specifying a value in the CKPT parameter in the initialization parameters. If the CKPT parameter is used, checkpointing does have to be requested on each COPY statement. The second subparameter also has the choices RECORD and BLOCK. If you specify RECORD for this parameter, both record level and block level automatic checkpointing occur, depending on the mode of transfer for each copy. BLOCK prevents automatic checkpointing on a record level transfer. BLOCK is the default for the second positional subparameter.

The third positional subparameter refers to transferring a set of CMS files. Multiple CMS files can be copied with a single copy statement using the set of files format. The internal logic for set of files resembles the logic used for MVS PDS members.

Note: Refer to the *CONNECT:Direct Process Guide* for more information on CMS set of files.

Since checkpointing information is sent with each file, this subparameter specifies whether checkpointing is allowed with a set of files transmission if a request is coded on the COPY statement. Valid options are PDS and NOPDS. NOPDS will prevent checkpointing, even if the user requested it on the Copy statement. PDS allows set of files transmissions to be checkpointed. PDS is the default for the third positional subparameter.

The fourth positional subparameter also refers to transferring a set of files. It determines what type of checkpointing occurs when automatic checkpointing is in effect. To enable automatic checkpointing, specify a value in the CKPT parameter in the initialization parameters. If you use the CKPT parameter, you will not have to request checkpointing on each COPY statement. This subparameter also has the choices PDS and NOPDS. If you specify PDS, all transmissions are automatically checkpointed. NOPDS will prevent automatic checkpointing of set of files transmissions. NOPDS is the default for the fourth positional subparameter.

The fifth positional subparameter specifies whether checkpointing takes place for VSAM files when the checkpoint parameter is specified in the COPY statement. Valid options are VSAM and NOVSAM. The default is VSAM.

The sixth positional subparameter specifies whether automatic checkpointing takes place for VSAM files. Valid options are VSAM and NOVSAM. The default is VSAM.

Default**:** RECORD BLOCK PDS NOPDS VSAM VSAM

CONNECT.WAIT=number of minutes

specifies the number of minutes CONNECT:Direct waits for a timeout during session establishment to a PC before assuming the connection is down and terminating the Process.

This timeout control is only for connection time—session establishment through Process negotiation—and does not pertain to sends and receives. It is necessary for connections to PCs only. The range is 0–60 minutes.

Default: 3

DATEFORM=(MDY|DMY|YMD|YDM)

specifies how Gregorian dates are displayed on screens and how Gregorian dates should be input. *All* displayed dates present the year in 4-digit format.

Note: CONNECT:Direct interprets 2-digit years as twentieth century if the value is 80 or greater. If the value is less than 80, the year is interpreted as a twenty-first century date.

Symbolic choices for this parameter indicate the order of the year (Y), month (M), and day (D) are displayed or accepted. You can use periods or slashes (/) to separate the components of a date value.

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

MDY specifies CONNECT:Direct to interpret Gregorian dates in one of the following formats:

- MMDDYYYY
- MM/DD/YYYY
- MM.DD.YYYY
- MMDDYY
- MM/DD/YY
- MM.DD.YY

DMY specifies CONNECT:Direct to interpret Gregorian dates in one of the following formats:

- DDMMYYYY
- DD/MM/YYYY
- DD.MM.YYYY
- DDMMYY
- DD/MM/YY
- DD.MM.YY

YMD specifies CONNECT:Direct to interpret Gregorian dates in one of the following formats:

- YYYYMMDD
- YYYY/MM/DD
- YYYY.MM.DD
- YYMMDD

- YY/MM/DD
- YY.MM.DD

YDM specifies CONNECT:Direct to interpret Gregorian dates in one of the following formats:

- YYYYDDMM
- YYYY/DD/MM
- YYYY.DD.MM
- YYDDMM
- YY/DD/MM
- YY.DD.MM

Default: MDY

DEBUG=xxxxxxxx

turns on a specific trace option or any combination of options, where xxxxxxxx represents a debug setting in hexadecimal. Refer to the *CONNECT:Direct Problem Isolation Guide* for more information.

This table shows the available function traces for CONNECT:Direct for MVS, VM/ESA, VSE/ESA, and MSP, with their respective DEBUG settings, ddnames used for output, and a description of the trace type.

DEBUG	Trace Type	DDNAME
80000000	COPY routine and RUN TASK traces	RADBDD01
10000000	Full TPCB/SYMBOLICS from DMCBSUBM	DMCBSUB
08000000	Session Manager trace	RADBDD05
04000000	Separate trace per task	Rnnnnnnn
02000000	API session trace	RADBDD07
01000000	DMGCBSUB trace	RADBDD08
00800000	NETEX task termination disconnect trace	NTXTRACE
00400000	TCQSH from DMCBCOPY	DMCBCOPY
00040000	GETMAIN/FREEMAIN trace	RADBDD16
00008000	I/O buffer trace	RADBDD21

(continued)

DEBUG	Trace Type	DDNAME
00004000	Write to operator (WTO) all dynamic allocation and parameters	RADBDD22
00002000	CONNECT:Direct for VM/ESA tape -handling trace	RADBDD23
00001000	CONNECT:Direct for VM/ESA dynamic allocation trace	RADBDD24
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

Default: 00000000

ECZ.COMPRESSION.LEVEL=n

determines the compression level of the ECZ extended compression. 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.

Default: **1**

ECZ.MEMORY.LEVEL=n

identifies how much virtual memory is allocated to the internal compression routine. 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 highest degree of compression, but it uses the most memory (256K)

Default: **4**

ECZ.WINDOWSIZE=n

determines the size of the compression window or history buffer. 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.

Default: **13**

ESTAE=YES | NO

specifies whether error recovery procedures will be invoked for CONNECT:Direct for VM/ESA. ESTAE=YES causes error recovery procedures to be invoked.

WARNING: Do not specify ESTAE=NO unless directed by CONNECT:Direct Technical Support. If you specify ESTAE=NO and a subtask abends, the error recovery is not invoked and one of the nodes enters into a wait state.

When you specify ESTAE=NO, no error recovery procedures are established.

Default: YES

EXPDT=(TT,DD,TD,DT) (if multiple values)
EXPDT=TT | DD | TD | DT | ALL | NONE (if only one value)

specifies CONNECT:Direct system defaults for propagating the expiration date from the FROM data set to a NEW data set. The following table lists the valid keywords for the EXPDT parameter and coding conventions.

Value	Meaning	Result
TT	tape-to-tape	Propagate the expiration date if the data set on the sending side and the data set on the receiving side are both on tape.
DD	DASD-to-DASD	Propagate the expiration date if the data set on the sending side is on DASD and the data set on the receiving side is also on DASD.
TD	tape–to-DASD	Propagate the expiration date if the data set on the sending side is on tape and the data set on the receiving side is on DASD.
DT	DASD-to-tape	Propagate the expiration date if the data set on the sending side is on DASD and the data set on the receiving side is on tape.

(continued)

(continued)

Value	Meaning	Result
ALL		Always propagate the EXPDT from data sets on all device types to data sets on all device types (works only for DASD and tape).
NONE		Never propagate the expiration date of the sending data set to the receiving data set. This is the default.

If you specify multiple keywords, enclose them in parentheses and separate them by a comma. If you code a single value, the parentheses are not required. If you code ALL or NONE, no other keywords can be coded.

The receiving side determines whether or not CONNECT:Direct propagates the expiration date. If the sending side specifies ALL in its initialization parameter, but the receiving side specifies NONE, the EXPDT is not propagated. Therefore, if the copy is from SNODE to PNODE, the PNODE side makes the determination; if the copy is from PNODE to SNODE, the SNODE side determines if the EXPDT is propagated.

CONNECT:Direct overrides the EXPDT initialization parameter in a Process when the following conditions occur:

- If you code an EXPDT or RETPD parameter for the receiving side (TO side) in the Process, CONNECT:Direct will use that EXPDT or RETPD and ignore the initialization parameter EXPDT.
- If you code an EXPDT or RETPD for the sending side (FROM side) in a Process and not for the receiving side, CONNECT:Direct will use the EXPDT in the Process, according to the EXPDT initialization parameter setting on the receiving side.
- If you do not specify the EXPDT in the Process and the input (FROM) data set is on DASD, CONNECT:Direct will obtain the EXPDT from the DSCB. If the input data set is on tape and the tape is SL or AL (Standard or ASCII), CONNECT:Direct will use the tape label. When CONNECT:Direct dynamically allocates the data set on the receiving side, this EXPDT is used, according to the initialization parameter EXPDT setting on the receiving side.

When you transfer a data set with no associated EXPDT, the following occurs:

If an input data set does not have an EXPDT, and the EXPDT is to be propagated, then the dynamic allocation string for the output data set specifies LABEL=EXPDT=00000. DASD data sets are considered to not have an EXPDT if the DSCB EXPDT is 00000. Tape data sets are

considered to not have an EXPDT if the HDR1 label contains 00000 for the EXPDT. When a data set is allocated with LABEL=EXPDT=00000, the tape header label or the DASD DSCB contains zeroes for the EXPDT on the output data set. If you have a tape management system or DASD management system, their databases may reflect a different EXPDT than the tape label or DASD DSCB, depending upon the defaults on the receiving side.

Default: NONE

IDRC.DEFAULT=COMP | NOCOMP

specifies the format for CONNECT:Direct tapes mounted on IDRC compatible drives.

COMP specifies all output tapes mounted on IDRC drives will use the compressed format.

NOCOMP specifies all output tapes mounted on IDRC drives will not use the compressed format.

Default: NOCOMP

INVOKE.ALLOC.EXIT=SEND|RECV|BOTH

determines whether to invoke the allocation exit upon sending a file, receiving a file, or both sending and receiving a file.

Default: RECV

INVOKE.ALLOC.EXIT.ON.RESTART=NO|YES

indicates whether to invoke the allocation exit on restart of a previously failed process.

Default: NO

INVOKE.SPOE.ON.SNODEID=NO|YES

indicates whether to invoke Security Point of Entry when a user codes SNODEID=parameter on the PROCESS.

Default: NO

LOG.PRINTER=luname

specifies the name of the LU1 printer dedicated to collecting CONNECT:Direct log data. For more information about log printer support, see the *CONNECT:Direct Console Operator's Guide*.

Default: None

LU2.WAIT=xx

specifies the amount of time CONNECT:Direct waits after a VTAM SEND or RECEIVE request is issued to an LU2 PC connection. The value *xx* can be from 0–60, and specifies the number of minutes to wait before the connection is considered to be lost. At that time, the CONNECT:Direct for VM/ESA or CONNECT:Direct for MS-DOS session is disconnected.

This parameter is valid only for LU2 PC connections. If you omit it or specify 0, the default is 3 minutes. A typical use of this parameter is to prevent Processes from hanging in the CONNECT:Direct for VM/ESA EX queue if the PC is booted while the transmission is in progress.

Default: 3

MAXBATCH=number of users

specifies the maximum number of batch users that can sign on to CONNECT:Direct at any one time. No other (DMBATCH) batch users are allowed to sign on when this limit is reached. The range is from 0–512. If 0 is used, this value is set to equal the number of MAXUSERS. If the value specified is larger than the MAXUSERS value, (default of 6), the MAXBATCH value will be reduced to equal the MAXUSERS value.

Default: 0

MAXPRIMARY=number of primary sessions

specifies the maximum number of primary node-to-node sessions that are allowed to start on a node. A CONNECT:Direct primary node represents the node-to-node half-session that is started when one or more Processes are ready to run. The CONNECT:Direct primary node is the session initiator. The range is from 2–500.

Default: 6

MAXPROCESS=number of executing PNODE and SNODE processes

specifies the maximum number of executing PNODE and SNODE processes allowed at one time. The valid range is 2–1024.

Default: Value of MAXPRIMARY plus MAXSECONDARY

MAXRETRIES=number of retries

specifies the maximum number of retries that is made to start a node-to-node session. If the session cannot be started, any Processes destined for the secondary node are placed in the timer queue for retries (TI RE). After all retries are exhausted, they go into the HO WC (hold queue, waiting connection). The range for MAXRETRIES is from 0–512. For related information, see the WTRETRIES initialization parameter.

Default: 7

MAX.TAPE=number of tape processes

specifies the maximum number of tape processes that are allowed to start in a node. When this limit is reached, any processes attempting to allocate a tape unit are either ended with a return code of 8 and an associated message of SDETAPRI or are placed on the timer retry queue if the ALLOC.CODES initialization parameter is updated to include the code TAPR. If TAPR if specified in ALLOC.CODES, the process is retried a number of times corresponding to the ALLOC.RETRIES parameter, at an interval corresponding to the ALLOC.WAIT parameter. The range of acceptable values for MAX.TAPE is 1–32767.

Default: 10

MAXSECONDARY=number of secondary sessions

specifies the maximum number of secondary node-to-node sessions that are allowed to start on a node. A CONNECT:Direct secondary node represents the node-to-node half-session that was initiated externally (SNODE). The range for MAXSECONDARY is from 2–512.

Default: 6

MAXSTGIO=maximum storage in bytes to be used for non-tape sequential data set transfers

specifies the maximum amount of storage used for non-tape sequential data set transfers. The size determines the number of buffers to allocate. The range is 1–99999. The higher the value, the better the performance for sequential file transfers. However, the higher the value, the higher the REGION size required for the DTF. The default value is 61440.

This value is used to determine the number of buffers/Network Control Programs (NCP) used for sequential I/O. The number of buffers/NCPs is calculated by dividing this value by the block size of the data set being transferred. The minimum number of NCPs is 3 and the maximum is 99, no matter how you code this value.

For example, if you specify 32000 for MAXSTGIO, the following number of NCPs/buffers are allocated for data sets with the block sizes listed below. Also listed is the amount of storage required for buffers for this transfer.

BLKSIZE	NCP/Buffers	Storage Used
4080	7	28,560
6400	5	32,000
32760	3 (minimum)	98,280
80	99 (maximum)	7,920

If you specify a large value for MAXSTGIO, be sure to re-evaluate the REGION size specified for the DTF. The region size must be large enough to accommodate the maximum number of sequential transfers that could take place at any one time, multiplied by the value coded for MAXSTGIO, plus the normal amount of region that the DTF requires.

If you are upgrading CONNECT:Direct for VM/ESA, coding a small number will cause CONNECT:Direct to use three buffers per transfer. However, the improvement in performance will only be slight. If you normally transfer data sets with small block sizes, you may want to code a higher value, such as 32000, in order to gain throughput on sequential files.

To determine this value, calculate your average blocksize for each data set and then determine the trade-off in performance versus region size. If your average blocksize is 16000, for example, then coding a MAXSTGIO of 16000 will have a slight effect on region size, but data sets with blocksize smaller than 16000 will be transferred in a more timely manner.

Default: 61440

MAXUSERS=number of users

specifies the maximum number of interactive users and (DMBATCH) batch users that can sign on to CONNECT:Direct at any one time. When this limit is reached, no other users are allowed to sign on. The range for MAXUSERS is from 2–512.

Default: 6

MCS.CLIST=console operator's CLIST library file name

specifies the file mode of the VM console operator's minidisk. This parameter is required for use of the console operator interface.

Default: None

MCS.SIGNON=[SIGNON USERID=(userid,password) NETMAP=network map]

specifies the console operator's Signon command for the Operator Interface. The SIGNON, USERID, and NETMAP keywords must be specified. This parameter is required for installations that use the console operator interface. All the parameters allowed on the SIGNON command may be specified here.

Default: None

NDM.KEY

is obtained from the documentation shipped with the installation tape. It must be entered in uppercase letters. This is a required parameter.

NDM.NODE

is obtained from the documentation shipped with the installation tape. This is a required parameter.

NETMAP.CHECK=NO | (ALL | TCP, ALL | BOTH | NODENAME, FAIL | WARN | PASS)

defines the communication types that will perform NETMAP checking, the verification to perform, and the action to take if the node does not exist.

NETMAP.CHECK=NO indicates that the CONNECT:Direct node attempting to establish a session with this CONNECT:Direct node need not be defined in the Network Map at this node. This is convenient when another CONNECT:Direct node initiates contact the majority of the time.

If you want to require the CONNECT:Direct node attempting to establish a session with this CONNECT:Direct node must be defined in the Network Map at this node under certain conditions, define three parameters, as follows:

The first parameter (ALL or TCP) defines what communication types perform NETMAP checking. ALL enables NETMAP checking for all communication types except for TCP/IP. TCP enables NETMAP checking for TCP/IP communication.

Note: If you code NETMAP.CHECK=TCP, you must provide a Network Map entry for each TCP/IP node. The adjacent node entry must specify the logical node name, port number, TCP/IP address, and a session type of TCP. An example follows:

```
ADJACENT.NODE=((UNIX.DALLAS,5555,199.5.5.5,TCP) ENVIRONMENT=UNIX)
```

The second parameter (ALL or BOTH or NODENAME) defines what verification is performed. ALL or BOTH for SNA enables verification on both the logical nodename and APPLID/LUNAME. For TCP, ALL or BOTH enables verification on both the logical nodename and IP address. For LU1 ALL or BOTH enables verification on both the logical nodename and LUNAME. NODENAME enables verification on the logical nodename only.

The third parameter (FAIL or WARN or PASS) defines what action should be taken if the node does not exist in the NETMAP. FAIL indicates that access to the system is denied. WARN indicates that access is allowed, but a warning message is issued. PASS indicates that access is allowed without any warning message being issued.

Default: (ALL, ALL, FAIL)

To enable NETMAP checking for all communication types, you must code the NETMAP.CHECK parameter for each. An example follows.

```
NETMAP.CHECK=(ALL,ALL,FAIL)
NETMAP.CHECK=(TCP,NODENAME,WARN)
```

In the previous example:

- The first entry for NETMAP.CHECK causes CONNECT:Direct to check all LU types and TCP, for both NODENAME and APPLID/LUNAME.
- The second NETMAP.CHECK entry causes CONNECT:Direct to check TCP nodes for NODENAME only. If the node does not exist, CONNECT:Direct issues a warning message but allows access.

PC.ENABLE.CHECK=YES | NO

specifies whether or not CONNECT:Direct checks for a match in security information at a CONNECT:Direct for MS-DOS node during a CONNECT:Direct for MS-DOS ENABLE operation.

YES causes a CONNECT:Direct for MS-DOS node to verify the userid and password stored at the PC node against the security information

associated with each process that is queued for that CONNECT:Direct for MS-DOS node. If a mismatch is found, the Process ends with a return code of 12 and a message ID of SPCB200I.

Default: NO

PRTYDEF=Process priority

specifies the default priority for Processes submitted to CONNECT:Direct. If the CONNECT:Direct user does not specify priority on the Process statement, CONNECT:Direct uses the default priority when placing the Process on the TCQ. The priorities range from 0–15. The highest priority is 15.

Default: 10

QUIESCE=YES | NO

specifies whether or not CONNECT:Direct holds Processes from execution. If you specify **YES,** no DTF-to-DTF sessions are started, but interactive sessions may be established. Any Process that would normally be executed will be placed in the WAIT queue.

See page 2-19 for how to resume normal operations by setting SESSIONS to R (Resume) with the MODIFY command.

Default: NO

REQUEUE=YES | NO

specifies whether to requeue Processes which abend, such as an x37, or with a return code greater than 4, or to allow any subsequent steps to run, or go to Process termination. This parameter will only be effective if checkpointing is in use. REQUEUE only applies to the PNODE, or submitting, side that has process control.

REQUEUE is not effective under any of the following conditions:

- SHUTDOWN IMMEDIATE is requested
- Session error caused the Process to terminate

YES will cause the Process to be placed in the hold queue if it did not end with any of the errors listed above but it abended or had a return code greater than 4 and one of the following is true:

- The Process or SUBMIT command has REQUEUE=YES
- Neither the Process nor the SUBMIT command has REQUEUE specified, but REQUEUE=YES is specified in the initialization parameters

- The data set on the PNODE side is a tape data set

If a dynamic allocation error occurs, the Process goes to ALLOCATION RETRY. When the specified number of allocation retries is exhausted, if REQUEUE=YES is specified the Process is placed in the hold queue with a status of HO RA (HO=Held by Operator; RA=Held for Restart Due to Allocation Error).

If the Process is abended, the status on the hold queue will be HE (hold/error). If the Process received a return code greater than 4, the status will be RH (restart/held).

NO allows the remaining steps in a Process to execute following a failed COPY STEP, but the failed COPY STEP will not be requeued. If REQUEUE is specified on a PROCESS or SUB statement, it will override the initialization REQUEUE specification.

Default: NO

REUSE.SESSIONS=YES | NO

allows you to control the usage of the sessions initiated by the local node. Once a Process is selected for execution between two nodes, control of the session is negotiated. If only one DTF has work destined for the other DTF, then the DTF with work to process controls the session. If they both have work to process, then the one with the higher priority work controls the session. This negotiation takes place at the completion of each Process. It is possible for the local DTF to initiate a session and be significantly delayed in utilizing that session based on the workload of the partner DTF.

YES specifies that the previously allowed negotiation takes place as described.

NO specifies that the remote DTF is not allowed to use the sessions established by the local DTF. (CONNECT:Direct does not allow Processes that are waiting for an eligible session to run when an SNODE session becomes available.)

Default: YES

RUN.TASK.EXIT=modname

specifies the name of the module responsible for verifying that a user is authorized to run a specified program in the DTF address space. The modname can be from one to eight alphanumeric characters, with the first character alphabetic.

Sterling Commerce ships sample exits, DMCXRT and ASMTASK. The sample exits can be found on the DTF's 191 minidisk.

A user must be defined on all nodes involved in Process execution.

Default: None

RUNTASK.RESTART=YES | NO

determines whether a RUN TASK program should execute at restart if CONNECT:Direct is unable to determine whether the program has run.

This initialization parameter corresponds to the node where the RUN TASK step executes. For example, if the RUN TASK step is executing on the SNODE, then the coding of the RUNTASK.RESTART parameter on the SNODE determines whether the RUN TASK program executes at restart.

Default: NO

SECURITY.EXIT=(modname, DATASET | ALL, PSTKT) | OFF

specifies the name of the CONNECT:Direct exit which performs security checking.

modname, or module name, can be 1–8 alphanumeric characters long, with the first character alphabetic.

These sample programs DMGACFx, DMGRACF, and DMGVMSEC, which Sterling Commerce ships as part of the CONNECT:Direct for VM/ESA sample library, may not meet your normal security requirements and should be modified accordingly.

DATASET specifies that the exit is invoked only for file security; the CONNECT:Direct Authorization Facility is used for access (signon) security.

ALL specifies that the exit is invoked for file and access security.

PSTKT indicates that the local DTF RACF security has been defined to accept RACF PassTicket passwords.

OFF specifies that there is no security; all requests are valid.

If the SECURITY.EXIT parameter is not specified or is commented out of the initialization parameters file, customized security is not performed and the CONNECT:Direct Authorization Facility is used.

For the first installation of CONNECT:Direct for VM/ESA, specify SECURITY.EXIT=OFF until a security exit is installed.

A user must be defined on all nodes involved in Process execution.

Default: CONNECT:Direct Authorization Facility

SECURITY.NOTIFY=YES | NO | HOLD

specifies whether CONNECT:Direct should send a message to users informing them of security failures on Processes they have submitted.

YES specifies CONNECT:Direct to send a message to users informing them of security failures on their submitted Processes. If you specify SECURITY.NOTIFY=YES and NOTIFY=%USER or NOTIFY=*userid* on the PROCESS statement, a security failure will cause a TSO notification to be sent to the user specified in the NOTIFY parameter of a SUBMIT or PROCESS statement.

NO specifies CONNECT:Direct should not send a message to users informing them of security failures on their submitted Processes.

HOLD specifies CONNECT:Direct to place Processes in the Hold queue with a status of HE if the other node returns an error during performance of security checking.

The following scenarios could occur with this parameter:

- SECURITY.NOTIFY=NO and a Process has NOTIFY=*userid* specified. If a stage 2 security error occurs on the SNODE, the userid is not notified. The userid is notified of all other errors or normal completion. All messages and return codes are in the Statistics File.

- SECURITY.NOTIFY=YES and a Process does not specify NOTIFY. The user is not notified of any errors or normal completion. All messages and return codes are in the Statistics File.

- SECURITY.NOTIFY=YES and a Process has NOTIFY=*userid* specified. If a stage 2 security error occurs on the SNODE, the userid is notified. The userid is also notified of all other errors or normal completion. All messages and return codes are in the Statistics File.

Default: NO

SECURITY.TYPE = RACF|VMSECURE|ACF2

depends upon the type of security subsystem in use and specifies that file access is monitored by one of the following:

RACF for Resource Access Control Facility (RACF) by IBM

VMSECURE for VMSECURE by Sterling Software

ACF2 for CA-ACF2 by Computer Associates International, Inc.

Default: None

SFS.SERVER.VMID=userid

specifies the name of the Shared File System (SFS) Server machine. The name of the Server machine can be up to 8 alphanumeric characters.

Note: You must specify this initialization parameter to use SFS files in the COPY Process.

STAT.ARCH.CONFIRM = YES | NO

indicates whether or not CONNECT:Direct requires confirmation on whether the contents of a statistics file pair have been archived before erasing them and reusing the file pair to record new information.

YES specifies that CONNECT:Direct requires confirmation before reusing the file. The DMSTARRT and DMSTARBT utilities provide archive confirmation. You can invoke these utilities from an archive process or an archive batch job, respectively.

If archive confirmation has not occurred at the time a file is to be switched to and therefore erased, CONNECT:Direct issues a WTOR requesting operator permission to overwrite the file. DTF activity effectively halts until you enter a response to the WTOR. An affirmative response causes an immediate file pair switch. A negative response disables the statistics logging function, but the DTF remains active.

Note: Sterling Commerce recommends that you specify the STAT.SWITCH.SUBMIT initialization parameter if you code the STAT.ARCH.CONFIRM=YES.

NO specifies CONNECT:Direct to erase the file contents at the time of a pair switch regardless of whether indication that the file was archived has been received.

Default: NO

STAT.BUFFER.ESDSDATA = number of ESDS data buffers
STAT.BUFFER.KSDSINDX = number of KSDS index buffers
STAT.BUFFER.KSDSDATA = number of KSDS data buffers

These three parameters specify the number of buffers VSAM allocates for the statistics clusters. CONNECT:Direct uses the values when generating VSAM access method control blocks (ACBs) for the statistics

files. This provides a means of tuning VSAM performance for statistics file access in the DTF. CONNECT:Direct specifies separate buffers for the index and data components for the key sequenced clusters. Each buffer is the size of the control interval of the specified component.

Default: STAT.BUFFER.ESDSDATA = 6
STAT.BUFFER.KSDSINDX = 6
STAT.BUFFER.KSDSDATA = 6

STAT.ERROR = ABEND | DISABLE

specifies the action of the DTF for certain types of errors which can occur in the Statistics Facility, such as VSAM errors or repeated abends.

STAT.ERROR=ABEND specifies that the DTF will abend with U3400. STAT.ERROR=DISABLE specifies that the Statistics Facility will be disabled but the DTF will remain active. The DTF will operate normally, however, no statistics records will be written.

When an abend occurs within the Statistics Facility, a dump is written to OPERATNS RDR and recovery is attempted. After five recovery attempts, the DTF abends with U3400 or the Statistics Facility is disabled, depending on the value specified for the STAT.ERROR parameter.

Default: ABEND

STAT.EXCLUDE = (record type list)

specifies what record types to exclude from the statistics log. The system does not pass excluded records to the statistics exit. The 2-character identifiers specify the record types in the list. See the *Using CONNECT:Direct Exits* chapter for a complete list of record type identifiers.

Records can also *selectively* be excluded by the Statistics exit. See the *Using CONNECT:Direct Exits* for information on the Statistics exit. The recording of specific record types can also be turned on and off during DTF execution using the STATISTICS ON/OFF API command.

This example excludes member records from the statistics log.

```
STAT.EXCLUDE = (MC)
```

Statistics records are often useful or indispensable in debugging problems. The exclusion of records from the statistics log can make problem determination by the Sterling Commerce Customer Services staff difficult. Do not exclude the following record types:

- CT Copy Termination
- PS Process Submit
- PT Process Termination
- RJ Run Job
- RT Run Task
- SW Submit within Process
- WO Write To Operator

Default: None

STAT.INIT = WARM | COLD

specifies whether or not to erase the contents of the statistics files defined to the DTF at initialization time.

WARM specifies that the system will not erase the contents at DTF initialization. In this case, statistics from prior DTF executions are available in the new execution.

COLD specifies that the system will erase all pre-existing records. Only records generated during the current execution are available.

Default: WARM

STAT.QUEUE.ELEMENTS = statistics record queue size

specifies the size of the queue that holds statistic records to be written. When a CONNECT:Direct task needs to write a statistic record, it queues the record to be written to the statistics facility asynchronously. The statistics facility then processes the queue and writes the statistics record. This parameter controls the size of this queue. When the queue becomes full, tasks that need to write a statistics record must wait until a slot in the queue becomes available. You can use this parameter as a statistics tuning device and as a method of controlling the number of waiting tasks. The maximum size of the queue is 999 records.

Default: 100

STAT.SWITCH.SUBMIT=dsn [member]

allows a site to name a sequential data set or a member of a PDS that contains a Process to be submitted at statistics file pair switch time. Use this feature to submit a Process that archives the statistics file pair that has just filled. Alternatively, the Process may submit a batch job which will in turn archive the statistics records.

Note: The STAT.SWITCH.SUBMIT parameter is identical in format to the DSN parameter of the CONNECT:Direct SUBMIT statement. See the *CONNECT:Direct Process Guide* for information on the SUBMIT statement.

If you code the STAT.ARCH.CONFIRM parameter as YES, then it is recommended that you specify the STAT.SWITCH.SUBMIT parameter also.

The Process submitted is associated with the security ID named in the STAT.USER parameter. CONNECT:Direct internally generates a SUBMIT command to submit the Process, and specifies a single symbolic parameter, &EDSN. The symbolic parameter &EDSN specifies the data set name of the entry sequenced cluster just filled. Therefore, the DTF supplies to the archive Process the name of the ESDS cluster to archive.

You can make archived statistics records available to the SELECT STATISTICS command by copying them to a VSAM entry sequenced cluster, and then use the DMSTBKEY utility to recreate the associated index information in a VSAM key sequenced cluster.

Default: None

STAT.SWITCH.TIME = (hh:mm:ss , ...)

specifies times of day to perform a statistics file switch. The STAT.SWITCH.TIME is in twenty-four hour clock format. You may specify up to four times in this parameter. The system initiates a switch whenever one of the named times occurs regardless of whether the currently active files are full. If you do not specify the STAT.SWITCH.TIME parameter, switching occurs whenever a file pair becomes full or in response to the API command STATISTICS SWITCH.

Default: None

STAT.TPREC = (start_time, end_time, snaps_per_hour)

specifies for CONNECT:Direct to create a statistics record that contains the number of processes and the amount of data sent, received, or both, for a node. The statistics record can be used for load balancing and tracking trends.

Valid values for snaps_per_hour are 1–60. If you specify a value of 1, CONNECT:Direct will take 1 snapshot per hour; if you specify a value of 60, CONNECT:Direct will take 1 snapshot per minute, or 60 per hour.

Default: 00:00:00, 24:00:00, 0

STAT.USER = (userid, [password])

specifies the security ID under which the statistics log is written and any archive Process or batch job will run. Use this parameter when implementing a stage 2 security exit.

A system task (a separate TCB) does the writing of the statistics files to minimize the impact of statistics logging on the throughput of the DTF. File pair switching and archive Process submission is also done by this task. Such processing is done in the background within the DTF, and therefore, has less impact on other activity. CONNECT:Direct for VM/ESA creates this task using the security userid from the STAT.USER parameter. The system also propagates the userid to the archive Process and to any batch jobs the archive Process submits.

If your site is running with full Stage 1/Stage 2 security implemented, it is not necessary to supply the password with this parameter.

The first parameter, userid, specifies the security ID that CONNECT:Direct will pass to a security exit. It can contain 1–8 characters.

The second parameter, password, specifies the current security password. The security exit uses this parameter to validate the current security password. It can contain 1–8 alphanumeric characters.

Note: Certain CONNECT:Direct statistics records are written with the STAT.USER ID in their userid field. For example, the S2 records that contain information about the statistics logging Process are written with this ID.

Because userid is one of the indexed statistics record fields, specifying a unique ID can facilitate the rapid retrieval of these records through the SELECT STATISTICS command when the TYPE and USER selection criteria are specified.

Default: If this parameter is not specified, or if the Stage 2 security exit is not implemented, the statistics logging task runs with the security ID of the DTF job, and with the userid of NDM. In this case, the TP and S2 records are written with NDM in their userid fields.

STATISTICS.EXIT=modname

specifies the name of the CONNECT:Direct statistics exit module the user can invoke to complement the CONNECT:Direct statistics gathering functions. This program can be used to log CONNECT:Direct information, to perform IBM system management facilities (SMF) functions, and to log custom information.

The supplied CONNECT:Direct sample statistics exit, STATEXIT, can be found on the DTF's 191 minidisk.

Default: None

SUBMIT.EXIT=modname

specifies the name of the module responsible for controlling changes to CONNECT:Direct parameters, such as Process name, priority, class, and secondary node. The module name can be from 1–8 alphanumeric characters long, with the first character alphabetic.

The supplied CONNECT:Direct sample exit, SUBMEXIT, is located on the DTF's 191 minidisk.

Default: None

TAPE.DETACH = YES|NO

specifies whether CONNECT:Direct will detach the tape drive used in a CONNECT:Direct Process step when the step completes.

Default: NO

TAPEMOUNT.EXIT = module name

specifies the name of the tape mount exit. For **VMTAPE**, the module name must be DMGVMTMX. DMGVMTMX is the name of the tape mount exit in the distribution load library that supports **VMTAPE**. DMGVMTMX is not modifiable as it is not distributed in source form. Refer to the *CONNECT:Direct for VM/ESA Tape Exit* appendix in the *CONNECT:Direct for VM/ESA Installation Guide* for more information on using **VMTAPE** with CONNECT:Direct for VM/ESA.

Default: None.

TAPE.PREMOUNT=YES | NO

specifies whether the CONNECT:Direct for VM/ESA tape pre-mount message is displayed.

Default: NO

TAPE.RETPD = dddd

specifies the number of days the tape is retained. This optional CONNECT:Direct initialization parameter can be used to specify a default retention period for newly created tapes. *dddd* is the number of days. Values range from 0–9999.

Default: 0

TCP=IBM | NO

specifies whether the TCP/IP connection modules are loaded during initialization and if so, what kind.

IBM specifies the IBM TCP/IP support.

NO causes no modules for the TCP/IP connection to be loaded during initialization.

Default: NO

TCP.NAME=name of TCP/IP service machine task

specifies the name of the TCP/IP service machine task. Use this parameter for IBM TCP/IP support only. The name can be from 1–8 alphanumeric characters long.

Default: TCPIP

TCP.PORTNUM=port#

specifies the TCP/IP server port number. The number can range from 1–65000.

Default: 1364

TCP.TIMER=wait time

specifies the amount of time, in seconds, a Process waits on a read of TCP data before the Process is cancelled and put in timer retry status. The number can range from 0–65535. A value of zero (0) indicates that no timer is used.

Sterling Commerce recommends you use a value of 60 if you have a requirement to prevent Processes from waiting indefinitely because of a lost connection.

Default: 0

TCQ=WARM | COLD

specifies how the TCQ is initialized. If WARM is specified, CONNECT:Direct will use the TCQ as it exists.

If COLD is specified, CONNECT:Direct reinitializes the TCQ and any Processes left on the TCQ are lost.

Default: WARM

TRANS.SUBPAS=YES|NO

specifies if the submitter's password is sent to the receiving node for use if the receiving node submits within a Process back to the submitter's node.

Default: YES

UPPER.CASE=YES|NO

specifies if console messages display in uppercase letters.

Default: NO

V2.BUFSIZE=maximum buffer size for this transmission

specifies the maximum buffer size that CONNECT:Direct uses for LU6.2 data transmission. The number ranges from 3712–65536. (This parameter was previously called TCP.BUFSIZE.)

Default: 4096

WTMESSAGE=NO | YES | (YES,nnn)

specifies whether a WTO message is generated when a Process is placed on the timer retry queue.

YES specifies that message SVTM110I is written to the console each time a Process is placed on the timer queue.

NO disables this feature.

(YES,nnn), the frequency subparameter, specifies that some, but not all, of the SVTM110I messages are written to the console. The variable *nnn* can be any number from 1–512. The default is 1, which specifies that the message will be written every time. If you specify 2, every second message is displayed, and so on.

CONNECT:Direct uses this parameter in conjunction with the MAXRETRIES initialization parameter when attempting to establish a lost session.

Default: NO

WTRETRIES=hh:mm:ss

specifies the amount of time in hours, minutes, and seconds between attempts to re-establish a node-to-node session. CONNECT:Direct uses this parameter in conjunction with the MAXRETRIES initialization parameter when attempting to establish a lost session.

Default: 00:03:00

VMTAPE.MACHINE.ID

identifies the VMTAPE service machine id where the VMTAPE exit (DMGVMTMX) will send tape services request to. This parameter is required if the VMTAPE service machine id is something other than VMTAPE.

Default: None

VMTAPE.TIMEOUT

is a timeout value used by VMTAPE to cancel the outstanding tape mount request if the tape is not mounted within this time frame. The parameter is specified in minutes. The valid range of 1–99.

Default: 30

VSAM.FM = filemode of VSAM files

specifies the filemode of the VSAM files used with CONNECT:Direct.

Default: B

Initialization Parameters for VSAM Files

The following is a list of the CONNECT:Direct initialization parameters for CONNECT:Direct VSAM files. Each initialization parameter (except for LU1.SCRIPT.DSN) must be specified. *CONNECT:Direct Quick Reference for MVS, VM/ESA, VSE/ESA, and MSP* also contains the file initialization parameters and their default values.

AUTHDSN=dsn

specifies the file name of the CONNECT:Direct VSAM Authorization file.

Default: None

CKPTDSN=dsn

specifies the file name of the CONNECT:Direct VSAM Checkpoint/restart file.

Default: None

MSGDSN=dsn

specifies the file name of the CONNECT:Direct VSAM Message file.

Default: None

NETDSN=dsn

specifies the file name of the CONNECT:Direct VSAM Network Map file.

Default: None

STAT.ARCH.DIR = archive directory file name

specifies the data set name of the directory of statistics archive files. Use the directory to maintain information about the files containing archived statistics records. This information includes the date/time range covered by the records in each file, and can be useful in locating the archive file containing records for a specific date/time. When this parameter is omitted, the archive directory functions are unavailable.

Default: None

STAT.DSN.BASE=dsname base
STAT.FILE.PAIRS=number

STAT.DSN.BASE specifies the high-level qualifier(s) for the statistics files cluster names. Use any valid MVS data set name qualifiers for this parameter. The high-level qualifier may range from 1–37 characters.

STAT.FILE.PAIRS indicates the number of file pairs to use. You must specify at least two file pairs. The number of file pairs ranges from 2–20.

The two parameters specify the statistics file pair list. During DTF initialization, CONNECT:Direct uses these two values to develop the data set names for the statistics files. The low-level qualifier, ESDS*nn*, is added to the base data set name to form the names of the ESDS clusters. In ESDS*nn*, *nn* is the number that identifies the position of the file pair in the list. CONNECT:Direct uses KSDS*nn* as the qualifier to form the names of the KSDS clusters.

The following example uses both STAT.DSN.BASE and STAT.FILE.PAIRS to specify the statistics file pair list.

```
STAT.DSN.BASE = CD.STATS              /* STATISTICS DSNAME BASE  */
STAT.FILE.PAIRS = 3                   /* NUMBER OF PAIRS         */
```

The example in the previous figure generates the following file pair list.

```
CD.STATS.ESDS01      /* FIRST FILE PAIR ... ESDS  */
CD.STATS.KSDS01      /* FIRST FILE PAIR ... KSDS  */
CD.STATS.ESDS02      /* SECOND FILE PAIR ... ESDS */
CD.STATS.KSDS02      /* SECOND FILE PAIR ... KSDS */
CD.STATS.ESDS03      /* THIRD FILE PAIR ... ESDS  */
CD.STATS.KSDS03      /* THIRD FILE PAIR ... KSDS  */
```

Default: None

TCQDSN=dsn

specifies the file name of the CONNECT:Direct VSAM Transmission Control Queue (TCQ).

Default: None

TCXDSN=dsn

specifies the file name of the CONNECT:Direct VSAM Transmission Control Queue space map (TCX).

Default: None

TYPEDSN=dsn

specifies the file name of the CONNECT:Direct VSAM Type file.

Default: None (parameters optional)

Glossary

A

ADJACENT.NODE

Entry in the Network Map. Adjacent node entries define nodes in the network with which the local CONNECT:Direct may communicate. Each entry specifies a locally used CONNECT:Direct name, its associated network communications name, and session control parameters for these nodes.

API (Application Program Interface)

This CONNECT:Direct component accepts commands from the Interactive User Interface (IUI), Batch Interface, the Operator Interface, or user-written program and places them in a format so that the user's request can be executed by the DTF. If there are errors, the API returns a message to the user. If there are no errors, the API sends the command to the DTF using a VTAM session.

APPLID

The name specified in the ACB macro that identifies the application program to VTAM. For CONNECT:Direct, these applids correspond to a DTF node name or an API applid.

API POOL

Identifies the applids to be used for API communication with the DTF.

API POOL Segregation

To separate the pools of APPLIDs for use by the individual API types (BATCH, CICS, and TSO).

API System ID(s)

The System Identifier (SYSIDs or SMF IDs) of the CPUs (up to sixteen) that will be sharing a copy of the CONNECT:Direct DTF utilizing the SDF (Shared DASD Facility) option of CONNECT:Direct.

Authorization Facility

CONNECT:Direct facility that grants access to CONNECT:Direct commands.

Authorization File

CONNECT:Direct authorization file used to control access to CONNECT:Direct and identify commands that can be executed by userid. This file can also be used in conjunction with security exit interfaces with the secured point of entry feature.

B

Batch Interface

CONNECT:Direct interface that allows users to request CONNECT:Direct services from a batch job stream through control statements passed to a CONNECT:Direct-supplied program, DMBATCH.

C

Checkpoint/Restart

Eliminates the need to retransmit an entire file in the event of a transmission failure. A value on the COPY statement or in the initialization parameter, CKPT.MODE and CKPT, specifies the checkpoint interval. If a copy procedure is interrupted, CONNECT:Direct will restart that copy at the last checkpoint.

Command Line Interface

CONNECT:Direct interface that allows users to submit CONNECT:Direct Processes and commands from their native command line environment.

Commands

Are used to initiate and monitor activity within the CONNECT:Direct system and can be issued from the TSO/ISPF IUI, the operator console, a batch job, or a user application program.

D

DTF (Data Transmission Facility)

The nucleus component of CONNECT:Direct. The DTF controls information distribution to other CONNECT:Direct nodes in the network. Start-up parameters that govern the overall activity of the DTF are defined within the initialization parameters.

E

ESF (Extended Submit Facility)

An optional CONNECT:Direct for MVS feature. The ESF allows users to queue data transfer requests to a CONNECT:Direct node that is not active. This allows users to submit work to CONNECT:Direct, even if the CONNECT:Direct DTF is down.

I

InterCONNECT Option (ICO)

The InterCONNECT option is an optional feature of CONNECT:Direct and CONNECT:Mailbox which provides an automatic, secure way to route application-produced distribution files from a Direct supported node to a Mailbox node for distribution, automatically distribute Mailbox batches to a Direct node upon arrival, and provide complete notification of success or failure at each step of the process.

IUI (Interactive User Interface)

The IUI interface is an ISPF screen and dialog component that allows users to define and submit CONNECT:Direct Processes as well as issue

CONNECT:Direct commands that monitor and control administrative and operations activity. An IUI is also available for a CICS environment with the optional product, CONNECT:Direct for CICS.

L

LOCAL.NODE

Entry in the Network Map. The local node entry defines the logical CONNECT:Direct name of the local CONNECT:Direct DTF and its associated communications name. The local node entry also contains the name of the transmission queue and the SUPERUSR ID password, if specified.

M

Modal Statements

CONNECT:Direct modal statements (IF THEN, EIF, ELSE, EXIT, and GOTO) allow you to alter the sequence of CONNECT:Direct Process execution based on completion of a previous Process step.

N

Network Map

VSAM file identifying all valid CONNECT:Direct nodes and applids in the network. There is one Network Map (Netmap) associated with each CONNECT:Direct node. There is one entry in that netmap for each of the other CONNECT:Direct nodes to which the local CONNECT:Direct node can initiate a session. The netmap entries also contain the rules or protocol to which the nodes will adhere when communicating.

Node

Any site in a network from which information distribution can be initiated.

O

Online Messages

Completion and error messages that are displayed online.

Operator Interface

Allows CONNECT:Direct commands to be issued from the MVS operator console. This interface also allows tailoring of CONNECT:Direct commands through a command list (CLIST) facility.

P

Parallel Sessions

Capability of having two or more concurrently active sessions between the same set of two LUs. With parallel session support, CONNECT:Direct allows multiple, concurrent file transfers between two CONNECT:Direct nodes.

PNODE (Primary Node)

CONNECT:Direct node on which the Process is being submitted. The primary node may also be referred to as the controlling or source node, but should not necessarily be interpreted as the sending node since PNODE can be the receiver. In every Process, there is one PNODE and one SNODE specified. The submitter of a Process is always the PNODE.

Process

A series of statements (which can be predefined and stored in a library) submitted through the API to initiate CONNECT:Direct activity, such as copying files, running jobs, and so on.

Process Statements

Are used to build a CONNECT:Direct Process. They contain instructions for transferring files, running operating system jobs, executing programs, or submitting other CONNECT:Direct Processes. Process statements include COPY, RUN JOB, RUN TASK, SUBMIT, SYMBOL, and Modals (conditional logic).

R

Retry Interval

Installation parameter that specifies the interval, in minutes, that the retries mentioned in the Max Retries parameter will be performed.

S

SDF (Shared DASD Facility)

An optional CONNECT:Direct for MVS product. Allows up to sixteen CPUs in a shared DASD complex to use a single copy of CONNECT:Direct. This allows applications and operators at each of the CPUs to initiate CONNECT:Direct activity. If the CPU containing the single copy of CONNECT:Direct also contains ESF, ESF is available to all other SDF nodes in the complex.

Segregation

To separate CONNECT:Direct APPLIDs by type. Also see API APPLID POOL Segregation.

Session Classes

Installation parameter that specifies the Process class groupings, priorities and number of Processes that can be concurrently executed on this CONNECT:Direct node.

SNA (Systems Network Architecture)

A network architecture designed to provide compatibility among a wide variety of hardware and software products so that they can be used to build complex networks. It defines protocols, standards, and message formats to which different hardware and software products must conform.

SNODE (Secondary Node)

The CONNECT:Direct node that interacts with the primary node (PNODE) during process execution. The secondary node (SNODE) may also be referred to as the participating, target, or destination node. In every Process, there is one PNODE and one SNODE.

SOLVE Option

The CONNECT:Direct for MVS SOLVE Option is an optional component of CONNECT:Direct. It is a callable application programming interface allowing SOLVE:Netmaster users to submit commands to and retrieve responses from CONNECT:Direct. The interface integrates automated data transfer with production applications on a variety of computer platforms running CONNECT:Direct.

SOLVE Option IUI

The SOLVE Option IUI provides an interactive interface to CONNECT:Direct for SOLVE:Netmaster users.

Statistics Files

A pair of VSAM data sets that hold CONNECT:Direct statistics records to document the history of a CONNECT:Direct Process.

Statistics Facility

CONNECT:Direct facility that records all CONNECT:Direct activities.

SYMBOL Statement

CONNECT:Direct Process statement that allows you to build symbolic substitution values.

Symbolics

Symbolic parameters are supported within CONNECT:Direct Processes. This allows one predefined Process to be used for multiple applications. For example, the file names for a COPY operation could be passed to the Process by the user submitting the Process.

T

TCP/IP

TCP/IP, Transmission Control Protocol/Internet Protocol, is a set of network standards that specify the details of how computers communicate, as well as a set of conventions for interconnecting networks and routing traffic.

TCQ (Transmission Control Queue)

A VSAM relative record data set (RRDS) used to hold all Processes that have been submitted to CONNECT:Direct.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y