# Connect:Direct®
# for z/OS

## Administration Guide

**Version 4.8**

Sterling Commerce
An IBM Company

*Connect:Direct for z/OS Administration Guide*
**Version 4.8**
**First Edition**



---

Sterling Commerce, Inc.

4600 Lakehurst Court Dublin, OH 43016-2000 *
614/793-7000

# Contents

# Chapter 1

# About Connect:Direct for z/OS

Connect:Direct is peer-to-peer file-based integration middleware optimized for assured delivery, high-volume and secure data exchange within and between enterprises. It is optimized for high performance and moves files containing any type of data across multiple platforms, disparate file systems, and disparate media.

Connect:Direct enables businesses to:

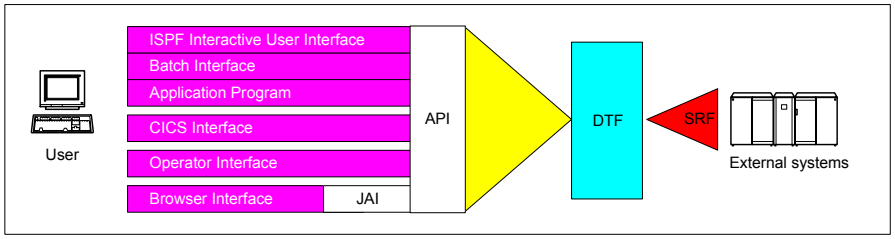
- Move large amounts of data internally and externally
- Share information with business partners
- Schedule business information-related application activities
- Automate data distribution
- Control and audit network activities
- Maintain network security
- Use one common command structure for information management with platform-specific interfaces

## Connect:Direct for z/OS Components

The major components of Connect:Direct for z/OS are:

- Data Transmission Facility (DTF), which executes user commands and Processes
- Application Program Interface (API), which enables user interfaces to communicate with the DTF
- User interfaces, such as the Interactive User Interface (IUI), browser end user interface (EUI), the batch interface, and the CICS interface, which interacts between the user and the API.
- Sysplex Requester Facility (SRF), which enables external CPUs to access a DTF without requiring their own DTF.

The following illustration shows these components.




**Note:** SRFs are only required when the non-DTF system does not share DASD with the DTF system.

# Connect:Direct for z/OS Configurations

Connect:Direct for z/OS can be used in either of the following configurations:

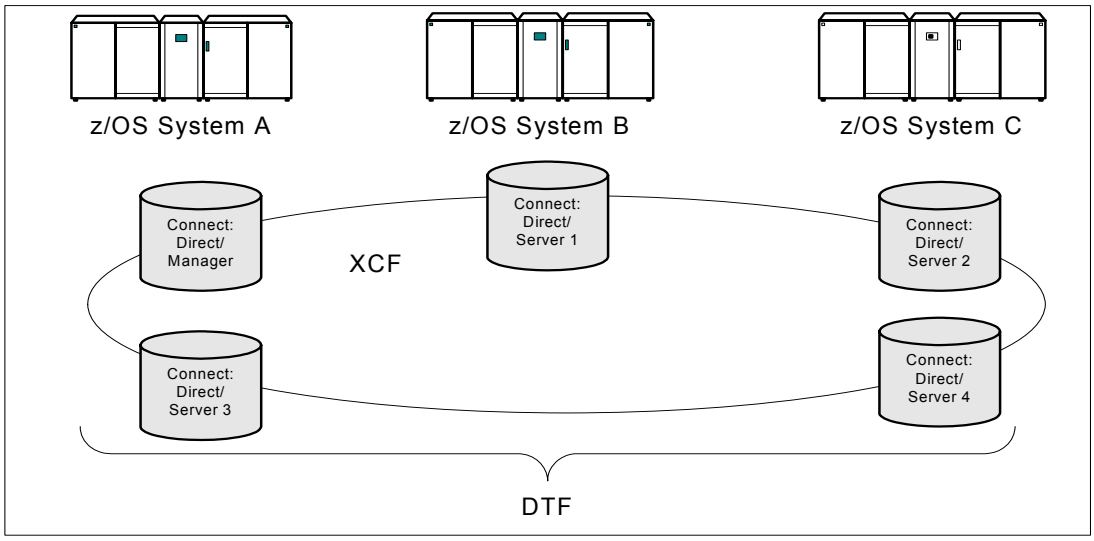
- Connect:Direct Stand-alone Server is a stand-alone Connect:Direct system operating in an IBM z/OS environment.
- Connect:Direct/Plex is a Connect:Direct system operating in an IBM z/OS sysplex or parallel sysplex environment. A Connect:Direct/Plex consists of:
  - Connect:Direct Manager, which manages the Connect:Direct/Plex. This includes handling:
    - Workload distribution and balancing
    - Processing queues
    - IUI, API, and SRF components
    - System-wide parameters
    - Network maps (netmaps)
    - Files that control user authorization, file attributes, and statistics logging
    - Secure+ parameters
  - One or more Connect:Direct Servers, which execute Processes assigned by the Connect:Direct Manager. Each Connect:Direct Server can be on a different z/OS system, or there can be multiple servers in one system. Individual servers can independently join or leave the Connect:Direct/Plex.

  The Connect:Direct Manager and all Connect:Direct Servers form the DTF in a Connect:Direct/Plex environment.

  A Process runs on any Connect:Direct Server in a Connect:Direct/Plex environment. You can balance workload dynamically among the Connect:Direct Servers. You can also direct Processes to specific servers (by specifying the PLEXCLASS keyword in the PROCESS

statement). For example, a Process that requires a tape drive can be directed to a Connect:Direct Server running on a computer with tape drives.

The Connect:Direct Manager communicates with the Connect:Direct Servers through the IBM z/OS Cross-System Coupling Facility (XCF), as in the following illustration.



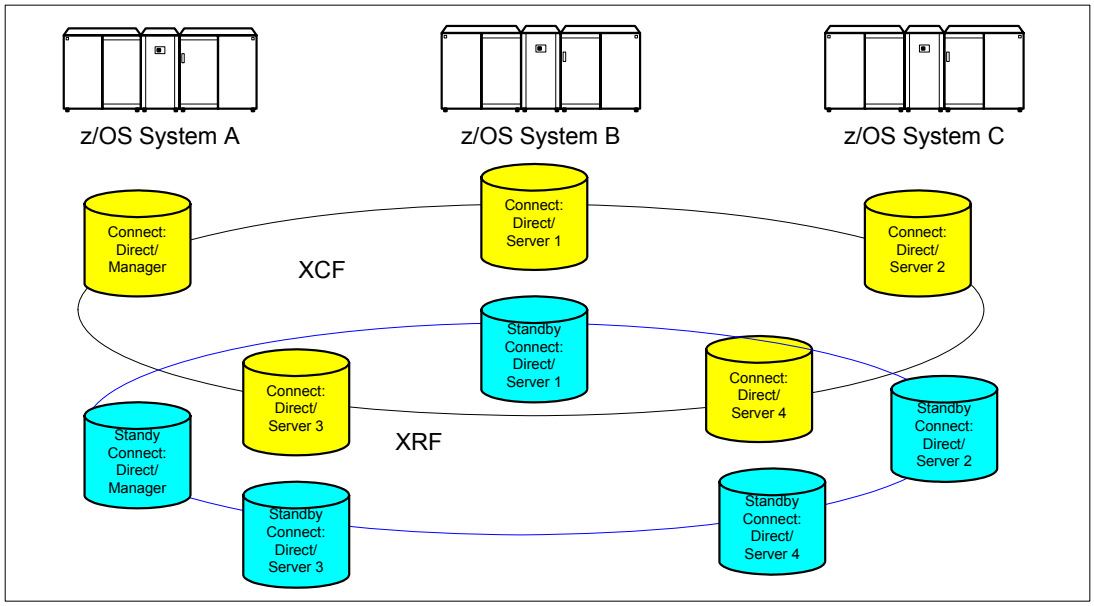

Connect:Direct/Plex operation is transparent; it appears as a single local node to other systems that communicate with it.

**Note:** For more information about XCF, see the IBM parallel sysplex documentation library. No additional XCF setup is required to use Connect:Direct/Plex.

Connect:Direct runs as a single system image in either the Connect:Direct Stand-alone Server or Connect:Direct/Plex environments. It starts as a single job. The Connect:Direct system shares parameters, the network map, and processing queues throughout the Connect:Direct system.

Both Connect:Direct Stand-alone Server and Connect:Direct/Plex can use the z/OS Extended Recovery Facility (XRF) to support extended recovery. If you use extended recovery in a Connect:Direct Stand-alone Server, a standby Connect:Direct system waits to assume the work of a failing system.

If you use extended recovery in a Connect:Direct/Plex environment, each active member has a standby member ready to assume work if the active member fails, as shown in the following illustration.




If you are using the dynamic VIPA resources, the standby managers or servers do not have to reside in the same z/OS system as the active manager or server they are backing up.

If you are not using the dynamic VIPA resources, the standby managers or servers must reside in the same z/OS system as the active manager or server they are backing up.

# Automatically Processing Files from a Watched Directory

Connect:Direct File Agent is a feature of Connect:Direct which provides unattended file management. File Agent monitors *watched* directories to detect new files. When File Agent detects a new file, it either submits a default Process or evaluates the file using rules to override the default Process and to determine which Process to submit. You create rules to submit different Processes based on the following properties:

- Specific or partial file names
- File size
- System events

You create the Processes used by File Agent on Connect:Direct; you cannot create them using File Agent.

To achieve optimum performance, configure File Agent to communicate with the Connect:Direct node where it is installed. File Agent can be installed on UNIX, Windows, and z/OS operating systems. For information to help you plan how to implement File Agent, see the *Managing Files*

*with Connect:Direct File Agent* chapter in your Connect:Direct administration guide or getting started guide. The Connect:Direct File Agent Help contains instructions for configuring File Agent.

## Connect:Direct Secure+ Option

Connect:Direct Secure+ Option provides enhanced security for Connect:Direct and is customized for one of the following security protocols:

- Transport Layer Security (TLS)
- Security Sockets Layer (SSL)
- Station-to-Station (STS)

If you are using TLS or SSL, you can also use the Sterling External Authentication Server application to validate certificates that are passed during a session.

Connect:Direct for z/OS may be installed with or without IBM's SMP/E product. It is strongly recommended that the SMP/E install option be used.

Although Connect:Direct Secure+ Option is installed at the same time as the base Connect:Direct product, it must be purchased as a separate product. Contact your Sterling Commerce representative to learn more about Connect:Direct Secure+ Option.

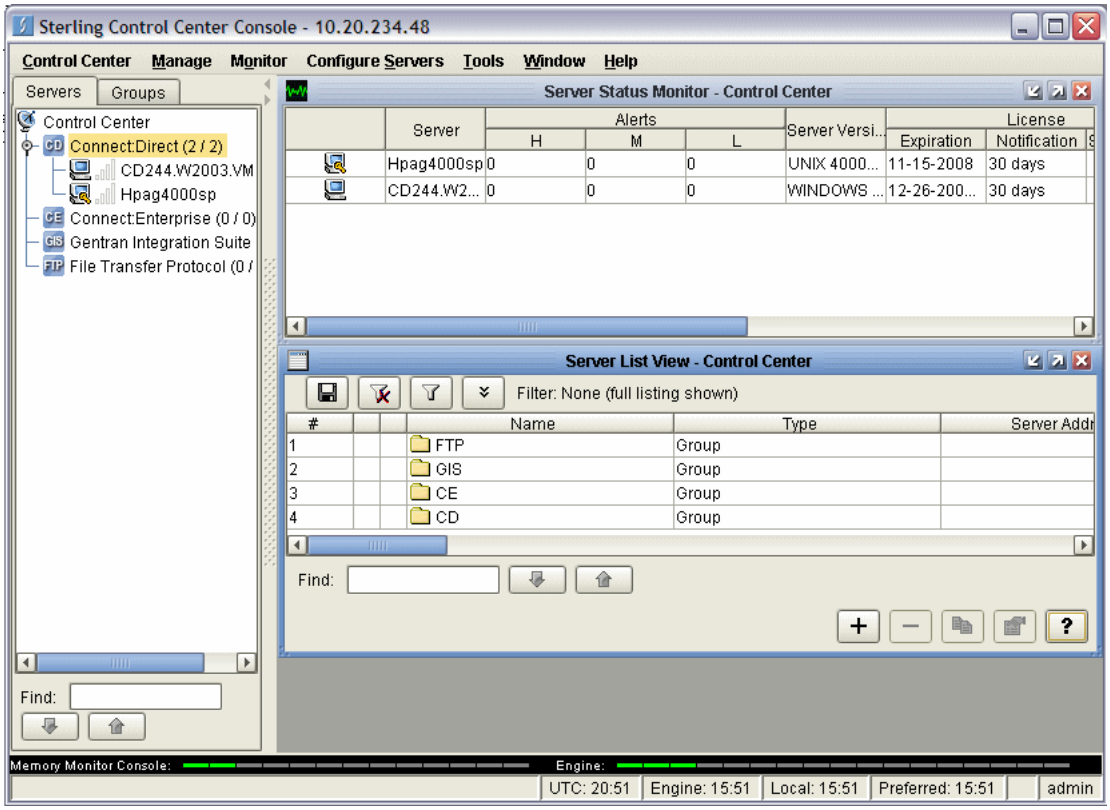
For a complete description of Connect:Direct Secure+ Option along with a discussion of security concepts, external authentication, and other considerations, see the *Connect:Direct Secure+ Option for z/OS Implementation Guide*. For the latest information on hardware and software requirements, installation, and other special considerations related to Connect:Direct for z/OS, see *Connect:Direct for z/OS Release Notes*.

## Sterling Control Center

Sterling Control Center is a centralized management system that provides operations personnel with continuous enterprise-wide business activity monitoring. Control Center lets you monitor these types of servers:

- Connect:Direct for z/OS
- Connect:Direct for UNIX
- Connect:Direct for Windows
- Connect:Direct HP NonStop
- Connect:Direct Select
- Connect:Direct for OS/400 (iSeries)
- Connect:Enterprise for z/OS
- Connect:Enterprise for UNIX
- Gentran Integration Suite (GIS) servers (including GIS clusters)

- FTP servers that use xferlog format




Sterling Control Center enables you to:

- Manage Connect:Direct for UNIX, Windows, or z/OS servers. You can manage these types of configuration objects:
  - Functional authorities
  - Initialization parameters
  - Netmap nodes
  - Netmap modes
  - Netmap communication paths
  - Secure+ nodes
  - Secure+ key certificates
  - Secure+ trusted certificates
  - Secure+ cipher suites
  - User proxies

You can also compare versions of the above configuration objects for a given server, do searches on configuration objects, make templates to simplify the creation of new configuration objects, and

do audits of changes to configuration objects. (For more on using Control Center to configure servers, see the *Sterling Control Center Configuration Management Guide*.)

- Monitor multiple servers
  - Group individual servers into server groups for a single view of system-wide activity. Group server groups into larger groups, and display a list view of servers and server groups.
  - View status and statistics on active or completed processing
  - Suspend, release, and handle Connect:Direct Processes on z/OS, Windows, HP NonStop, and UNIX platforms
  - Stop Connect:Direct servers on z/OS, Windows, HP NonStop, OS/400 (iSeries), and UNIX platforms
  - Pause and resume monitoring for a Connect:Direct server
- Monitor service levels
  - View information about active and completed Processes across servers within your network
  - Receive notification of data delivery events that occur or do not occur as scheduled
  - Define rules based on processing criteria that can generate an alert, send an e-mail notification, generate a Simple Network Management Protocol (SNMP) trap to an Enterprise Systems Manager (ESM), run a system command, or issue a Connect:Direct server command
  - Monitor for alerts about conditions such as a server failure or a Process that starts late
  - Create service level criteria (SLCs) that define processing schedules, monitor Processes and file transfers for compliance with the processing schedules, and generate alerts when schedules are not met
- Analyze key operational metrics through reports to document and analyze processing activity
- Create customized reports based on criteria you define, and schedule the reports to run and be delivered automatically by email
- Validate user authenticity for console to engine connections using one or more of four authentication methods, including password validation, host name identification, Windows domain, and TCP/IP address (or three methods in the case of the Web console, which does not support domain authentication)
- Identify additional Connect:Direct servers (through Node Discovery) that may need to be monitored based on communications with a currently monitored server

Sterling Control Center enhances operational productivity and improves quality of service by:

- Monitoring and configuring, and managing licenses for, Connect:Direct servers (for Windows, UNIX, and z/OS) from a central point
- Ensuring that critical processing windows are met
- Reducing impact on downstream processing by verifying that expected processing occurs
- Providing proactive notification for at-risk business processes

- Consolidating information for throughput analysis, capacity planning, post-processing operational or security audits, and workload analysis
- Reducing the risk of errors associated with manual system administration, including eliminating individual server logon to view activity and the need to separately configure each server for error and exception notifications

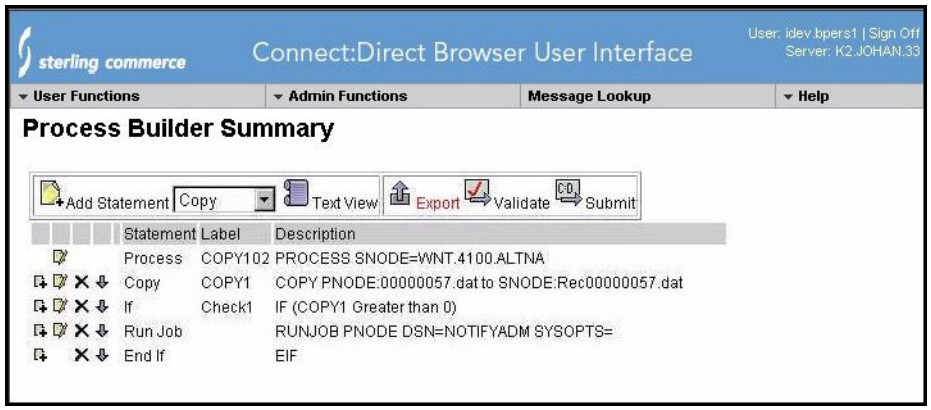
Sterling Control Center is available for purchase as a separate product. Contact your Sterling Commerce representative to learn more about Sterling Control Center.

## Connect:Direct Browser User Interface

Connect:Direct Browser User Interface allows you to build, submit, and monitor Connect:Direct Processes from an Internet browser, such as Microsoft Internet Explorer.

You can also perform Connect:Direct system administration tasks, such as viewing and changing the network map or initialization parameters, from Connect:Direct Browser. The specific administration tasks that you can perform depend on the Connect:Direct platform that your browser is signed on to and your security level.

Connect:Direct Browser is distributed on CD-ROM with Connect:Direct for z/OS, Connect:Direct for Windows, Connect:Direct for UNIX, and Connect:Direct for HP NonStop. It can also be downloaded from the Sterling Commerce Web site. Connect:Direct Browser is installed on a Web server and can be accessed by administrators and users through a URL. The following example shows the page used to graphically build a Process:




To learn more about Connect:Direct Browser, see the documentation on the Connect:Direct Browser CD-ROM or available online from the Sterling Commerce Documentation Library.

# Connect:Direct for z/OS Documentation

See the *Connect:Direct for z/OS Release Notes* for a complete list of the product documentation.

## About This Guide

The *Connect:Direct for z/OS Administration Guide* is for programmers and network operations staff who maintain Connect:Direct for z/OS.

Read the first two chapters of the book to gain general knowledge of Connect:Direct. These chapters introduce you to the basic components and general concepts, and summarize the administrative commands. The last chapter provides general information on troubleshooting and how to provide information to Sterling Commerce Customer Support when problems arise.

This manual assumes knowledge of the IBM z/OS operating system, Job Control Language (JCL), and Interactive Systems Productivity Facility (ISPF). If you are not familiar with the z/OS operating system, refer to the IBM z/OS documentation.

## Task Overview

Connect:Direct for z/OS administration includes the following activities:

| Activity | Description |
|---|---|
| Task administration | Tasks are the functions and requests processed by the DTF. System administrators can display task status and stop tasks from executing. |
| Initialization parameter administration | Initialization parameters define system-wide and member-specific defaults. System administrators can display and update initialization parameters. |
| Diagnostics | System administrators can run diagnostic traces to help solve problems. |
| Security maintenance | System administrators control security through several tools, including security exits which interface with third-party security packages, and the Authorization Facility which controls access to Connect:Direct functions. |
| Type file maintenance | System administrators maintain the Type file that contains file attribute information. Connect:Direct uses this information to create or access files. |
| Network map administration | The network map (NETMAP) identifies the local Connect:Direct node and the nodes it communicates with. It is created during system installation and maintained by the system administrator. |
| Exit maintenance | Connect:Direct provides exits to external user-written programs. The system administrator can implement and maintain the exits. |
| Message file maintenance | System administrators can update the Connect:Direct message library with user-defined messages. |
| Screen customization | System administrators can customize the IUI screens used to submit Processes. |

| Activity | Description |
|---|---|
| Statistic administration | The Connect:Direct statistics facility writes session statistics to VSAM log files. The system administrator can monitor, archive, and administer statistics files. |
| CICS administration | The system administrator can perform system maintenance tasks in a CICS environment. |
| Unattended file management | After determining the Connect:Direct server that Connect:Direct for z/OS most frequently communicates with and the directories that need monitoring, the system administrator can configure the File Agent component to enhance the automation accomplished with Connect:Direct Processes. |
| DBCS Support | Connect:Direct provides support, which enables the system administrator to translating ASCII and EBCDIC DBCS data so that users can copy files containing double-byte character set data. |
| Performance Tuning | System administrators can evaluate and improve the performance of the Connect:Direct system by reviewing and testing factors to help fine-tune the system and enhance overall efficiency of the system. |
| Troubleshooting | When users encounter problems, the system administrator can use a variety of diagnostic tools to help determine and resolve their problems, and gather information to provide Sterling Commerce Customer Support with supporting documentation. |

# Chapter 2

# Basic System Administration

This chapter provides instructions for basic system administration tasks and contains the following topics:

- The Administrative Options Menu
- Connect:Direct Native Command Structure
- Managing Tasks
- Displaying System Settings and Status
- Performing Administrative Tasks

Diagnostic tasks are discussed in *Chapter 17, Isolating Problems*.

**Note:** You can also use the Connect:Direct Browser to perform some of the procedures in this chapter. To learn more about the Connect:Direct Browser, see the *Connect:Direct Browser User Interface User's Guide* on the Connect:Direct Browser CD-ROM or online in the Sterling Commerce Documentation Library.

## The Administrative Options Menu

The Administrative Options Menu provides access to system administration functions. To access this menu, do one of the following:

- 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 z/OS User's Guide* for more information about using the **ADMIN** command from the Connect:Direct Primary Options Menu.

**Note:** The Connect:Direct Administrative Options Menu is not available to all users. Access to Connect:Direct functions is controlled through the User Authorization file, described in Chapter 4, *Maintaining User Authorization*.

The following figure is an example of the Connect:Direct Administrative Options Menu. To access a function, type the function abbreviation on the command line and press **ENTER**.

```
     View, Modify, Control, Delete, Secure+
_____________________________________________________________________________
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:  mm.dd.yyyy   *
    SU    - VIEW USER AUTHORIZATION RECORD            * TIME:   hh:mm        *
    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
    ARS   - ARS REPORTING FACILITY
    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
```

You can select the following options from the Connect:Direct Administrative Options Menu.

- To maintain the Type Defaults file that contains file attribute information used during Process submission, use the following options. For more information, Chapter 5, *Maintaining the Type File*.

| Option | Description |
|---|---|
| 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. |

- To maintain the User Authorization file that controls access to Connect:Direct functions, use the following options. For more information, see Chapter 4, *Maintaining User Authorization*.

| Option | Description |
|---|---|
| SU | Displays the Select User screen where you can examine the profile of a user in the Connect:Direct Authorization file. |

| Option | Description |
|---|---|
| IU | Displays the Insert/Update User screen where you can add a user to the system or change user privileges on the system. |
| DU | Displays the Delete User screen where you can remove a user from the Connect:Direct Authorization file. |

✦ To select and flush tasks., use the following options.

| Option | Description |
|---|---|
| TS | Accesses the Select Task screen where you can determine the task ID, task type, and task number for a task. For more information, see *Displaying Task Status* on page 32. |
| TF | Displays the Flush Task screen where you can remove a task from the execution queue. For more information, see *Removing Tasks from Execution* on page 35. |

✦ To initialize traces, suspend and resume sessions, modify initialization parameters, type native commands, terminate Connect:Direct, and access ARS, use the following options.

| Option | Description |
|---|---|
| MD | Displays the Modify screen where you can request traces and modify system functions. For detailed information on the MODIFY command and traces, see *Connect:Direct MODIFY Command* on page 368. For more information on the MODIFY SESSIONS command, see *Suspending and Resuming Processing on Individual Nodes* on page 47. For more information on the MODIFY INITPARMS command, see *Modifying Initialization Parameter Settings while Connect:Direct is Running* on page 40. |
| C | Displays the Native Command screen where you can type and execute any Connect:Direct command by providing it in native syntax. For more information, see *Connect:Direct Native Command Structure* on page 29. |
| SN | Displays the Stop Connect:Direct screen where you can stop the operation of Connect:Direct. For mor information, see *Stopping Connect:Direct* on page 44. |
| ARS | Displays the Connect:Direct - ARS REPORT OPTIONS menu. While Connect:Direct itself produces statistics, the Activity Reporting System (ARS) gives you access to more information and also provides sorting capabilities. For more information, see the *Connect:Direct for z/OS Facilities Guide*. |

✦ To view and maintain the network map and translate TCP/IP names, use the following options.

| Option | Description |
|---|---|
| NM | Displays the Select Network Map screen, where you choose to display or print, the defined Connect:Direct nodes from the network map file and translation of TCP/IP host names and network addresses. For more information on the SELECT NETMAP command and the SELECT TCPXLT command, see *The Network Map* in *Connect:Direct for z/OS User's Guide.* |
| UNM | Enables you to update the network map dynamically. For more information, see *Updating the Netmap through the IUI Interface* on page 159. |

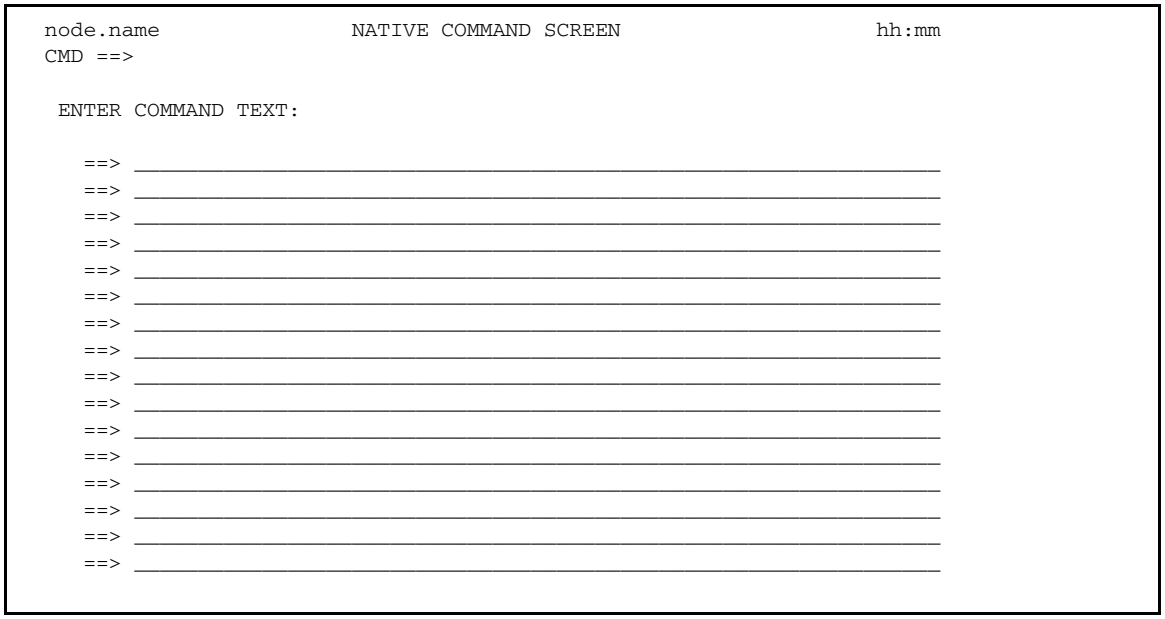
✦ To inquire about DTF internal status and perform statistics functions, use the following options.

| Option | Description |
|---|---|
| INQ | Displays the Inquire C:D Internal Status screen, from which you can request information about the:<br>◆ Statistics archive file directory (see *Viewing the Statistics Archive Directory through the IUI Interface* on page 294)<br>◆ Connect:Direct/Plex environment (see *Displaying Connect:Direct/Plex Status* on page 42)<br>◆ Statistics logging facility (see *Statistics Inquiry through the IUI Interface* on page 291)<br>◆ Connect:Direct APKEY file (see *Displaying the Asset Protection Key File Settings* on page 37)<br>◆ SNMP Trap Table (see *Displaying the SNMP Trap Table* on page 210)<br>◆ TCP Listen Status (see *Viewing a TCP Listen Status Report* on page 146)<br>◆ Current DEBUG settings (see *Displaying DEBUG Settings* on page 366)<br>◆ Connect:Direct initialization parameters settings (see *Displaying Initialization Parameter Settings* on page 39) |
| STAT | Displays the Statistics Command screen from which you can request functions related to the Statistics files, such as initiating statistics file pair switching, confirm statistics file archival, enable statistics recording, and disable statistics recording. For more information, see Chapter 12, *Administering Statistics*. |

# Connect:Direct Native Command Structure

You use the Native Command structure to build a more detailed list of parameters than you can from the command panels. You can type any Connect:Direct command or series of Connect:Direct commands using the Native Command structure.

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




Observe the following rules when you type your command:

- You must start keywords on the next line or break them by a separator (blank or comma).
- To use comments on the Native Command Screen, type an asterisk in the first column of the input line. Typing an asterisk 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.

## Examples

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

```
node.name               NATIVE COMMAND SCREEN                    hh:mm
CMD ==>

 ENTER COMMAND TEXT:

   ==> SUBMIT PROC=TEST2 ____________________________________________
   ==> ____________________________________________________________
   ==> *SELECT PROCESS WHERE (PNAME=TEST2) ___________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
```

To monitor the progress of TEST2, type an asterisk in column 1 of the first input line (before SUBMIT), delete the asterisk from the third input line (before SELECT), and press **ENTER**.

You can also submit a Process from the command line. In the following example, the Process TEST2 is submitted from the command line. The SELECT PROCESS (line 3) takes place just as in the previous example. The screen sample follows.

```
node.name                NATIVE COMMAND SCREEN                hh:mm
CMD ==> SUBMIT PROC=TEST2

 ENTER COMMAND TEXT:

   ==>_____________________________________________________________
   ==> ____________________________________________________________
   ==>  SELECT PROCESS WHERE (PNAME=TEST2) ___________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
   ==> ____________________________________________________________
```

# Managing Tasks

Connect:Direct tasks perform and manage work in a DTF. This section describes how to display task information and remove (flush) tasks.

The two categories of Connect:Direct tasks are:

- System tasks that operate the Connect:Direct DTF
- User tasks that manage work done within the DTF

The following table lists the Connect:Direct tasks and their functions:

| Type | Task | Function |
|---|---|---|
| **System** | Master (M) | Controls the dispatching and logon processing for the DTF |
| | Timer (T) | Performs timer services for the master task and Process-related timer functions |
| | Operator interface (C) | Enables you to communicate to the DTF through the operator console |
| | Extended Submit Facility Scan (W) | Scans the TCQ at predefined intervals and moves submitted Processes that are not on the current processing queue (PCQ) to the PCQ |
| **System** | Open/Close Task (O) | Manages the VTAM ACB open/close and TPEND exit |
| | TCP Task (U) | Monitors incoming TCP/IP session requests |
| | XCF Communication (Q) | Manages communications between Manager and Servers in a Connect:Direct/Plex |
| | TCP API Task (D) | Monitors incoming TCP/IP Connect:Direct API session requests |
| | LOGON (L) | Reserved for use during logon processing |
| | Statistics (A) | Controls status logging |
| | Session Creation TCA (F) | Manages Connect:Direct Processes and tasks |
| | Statistics Archive Submit Task (Z) | Submits the Statistics File Archive Process |
| | CTCA Server Task (Y) | Manages the CTCA tasks |
| | Connect:Direct/Plex Queue Manager Task (Q) | Manages the VTAM ACB open/close and TPEND exit |
| **User** | PNODE Task (P) | Manages the work related to a request that initiated the current session |
| | SNODE Task (S) | Manages the work related to a partner PNODE task |
| | IUI Task (I) | Manages the requests from a session with an IUI user |

| Type | Task | Function |
|---|---|---|
| | Background (Batch) Task (B) | Manages the request from a batch (DMBATCH) user |

## Displaying Task Status

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

| Label | Command | Parameters |
|---|---|---|
| (optional) | SELect TASK | **PRint \| Operator Table \| DISplay**<br>WHERE (SERVER = server name) |

### Required Parameters

The required SELECT TASK parameters are:

| Parameter | Description |
|---|---|
| PRint \| Operator Table \| DISplay | PRint specifies the command is output in hard copy to a printer.<br>TABLE specifies the command is output to a table.<br>DISplay specifies the command is output to the screen. |

### Optional Parameters

The optional SELECT TASK parameters are:

| Parameter | Description |
|---|---|
| WHERE (SERVER = server name) | This parameter specifies the name of the Connect:Direct/Plex member where the SELECT TASK command is performed. The server name parameter is the 1–8 character name assigned to a Connect:Direct/Server by the CDPLEX.SERVER initialization parameter.<br>This parameter only applies to a Connect:Direct/Plex environment. If this parameter is not specified in a Connect:Direct/Plex environment, the SELECT TASK is performed on the Connect:Direct/Manager. |

### Selecting a Task through the Batch Interface

To use the SELECT TASK command from the batch interface:

1. Place your commands in a batch job stream.
2. Submit the job while Connect:Direct is running.

3. Verify your results.

The following SELECT TASK command example sends output to the log printer:

```
SEL TASK PRINT
```

The following SELECT TASK command example is performed on a Connect:Direct/Server named SERVER3 and sends the output to your terminal in operator table format.

```
SEL TASK O WHERE(SERVER=SERVER3)
```

### Selecting a Task through the IUI

You must select the appropriate output for the SELECT TASK report. You can either display (in report or operator table format) or print the report.

1. Select option **TS** from the Administrative Options Menu.

   The SELECT TASK screen is displayed.

```
node.name                        SELECT TASK                              hh:mm
CMD ==>
                                                                      CMD: OPR

  CMD: O ... OPERATOR TABLE
       P ... PRINT REPORT    D ... DISPLAY REPORT


  SERVER => _________
```

2. Select one of the following display types.

| Option | Description |
|---|---|
| D | Displays the report on your screen and is captured in the TMPDSN specified in the SIGNON defaults of the user. |
| O or OPR | Displays the report to your screen in the operator table format. |
| P | Sends the report to a printer. |

3. If you are running in a Connect:Direct/Plex environment, type the member name on which you want to perform the SELECT TASK. If you leave this field blank in a Connect:Direct/Plex environment, the SELECT TASK is performed on the Connect:Direct/Manager.

   If you are running in a Connect:Direct/Stand-alone Server environment, leave this field blank.

4. Press **ENTER**.

If you selected Display from the Select Task screen, the following screen is displayed:

```
BROWSE--XXXXXXXX.XXXXXXX.XXXXX.XXXXXX.XXXXX--LINE 00000000 COL 001 080
COMMAND ===>                                          SCROLL ===> CSR
****************************** TOP OF DATA  ************************
=============================================================
            SELECT TASK for C:D/Plex Manager
=============================================================
 TASK TASK           XMIT          PNAME/
  ID  NUM  STATE     STATE                   PNUM
_____________________________________________________________________
  M   001  INACTIVE
_____________________________________________________________________
  T   002  TIMER
_____________________________________________________________________
  A   003  INACTIVE
_____________________________________________________________________
  Z   004  INACTIVE
_____________________________________________________________________
  C   005  MISC I/O
_____________________________________________________________________
  F   006  INACTIVE
_____________________________________________________________________
  U   007  SUBTASK   TCP MAIN TAS
```

If you selected the operator table format, the following screen is displayed:

```
------------------------------OPERATOR TABLE---------------      Row 1 to 20 of
22
==>                                                          SCROLL ===> PAGE
OPTION        TID    TASKNO   STATE      SUB-STATE    PNAM/UID    PNUM
-------------------------------------------------------------------------------
              M      1        INACTIVE
              T      2        TIMER
              A      3        INACTIVE
              Z      4        INACTIVE
              C      5        MISC I/O
              F      6        INACTIVE
              U      7        SUBTASK     TCP MAIN TAS
              U      8        ST RUNNG    TCP ACCEPT
              D      9        SUBTASK     TCP MAIN TAS
              D      10       API RUN     TCP ACCEPT
              O      11       INACTIVE
              I      13       VTAM I/O    RECEIVE     SJONES2
              I      14       RUNNING                 BSMITH1
              W      12       TIMER
              Q      37       WAIT4WRK
              Q      38       WAIT4WRK
              Q      39       WAIT4WRK
              Q      40       WAIT4WRK
              Q      41       WAIT4WRK
              Q      73       WAIT4WRK
```

You can perform the following operations from the Operator Table:

- Type **F** next to the task ID to flush and suspend nonsystem tasks.

- Type **P** to suspend a task.

Refresh the OPERATOR TABLE screen by typing **Q** in the command line and pressing **ENTER**.

## Removing Tasks from Execution

Use the FLUSH TASK command to remove a task from execution. Identify the task by its task number. You cannot flush system or IUI tasks.

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

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

| Label | Command | Parameters |
|---|---|---|
| (optional) | FLush TASK | **WHERE** ( |
| | | TASK = (tasknumber \| (list), |
| | | SERVER = server name |
| | | ) |
| | | FORCE |

### Required Parameter

The required FLUSH TASK parameter is:

| Parameter | Description |
|---|---|
| WHERE (TASK = (tasknumber \| (list), SERVER=server name) | (tasknumber \| (list) specifies the tasks to flush either by task number or a list of task numbers.<br><br>This parameter is required. SERVER=server name specifies the name of the Connect:Direct server where the FLUSH TASK is performed. The server name parameter is the 1–8 character name assigned to a Connect:Direct/Server by the CDPLEX.SERVER initialization parameter. This parameter is only valid in a Connect:Direct/Plex environment. If you omit this parameter in a Connect:Direct/Plex environment, the FLUSH TASK is performed on a Connect:Direct/Manager. |

### Optional Parameters

The optional FLUSH TASK parameter is:

| Parameter | Description |
|---|---|
| FORCE | This parameter specifies that the flush task is forced. Do not use the FORCE parameter when the task is executing on a LU6.2 session. The session terminates immediately and statistics for the task are not exchanged between the two nodes.<br><br>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 to complete), then it does not recognize the FLUSH TASK indicator, and the task is not flushed; otherwise, the program recognizes for the FLUSH TASK indicator and takes the appropriate action.<br><br>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 *Connect:Direct for z/OS User's Guide* for the actions taken for the specific STATE and SUBSTATE.<br><br>**Note:** Use the SELECT TASK command to determine the STATE and SUBSTATE of the task. |

## Removing Tasks from Execution through the Batch Interface

To use the FLUSH TASK 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 z/OS User's Guide.*
2. Submit the job while Connect:Direct is running.
3. Verify your results.

The following example shows the FLUSH TASK command force flushing three Connect:Direct tasks:

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

The following example shows the FLUSH TASK command flushing a Connect:Direct task running on a Connect:Direct/Server named OSGOOD:

```
FLUSH TASK WHERE (TASK=9,SERVER=OSGOOD)
```

### Removing Tasks from Execution through the IUI

You can flush a task from the system in the following ways:

- Flush a Task screen
- Operator Table of the SELECT TASK screen

Refer to the example of the SELECT TASK Operator Table beginning on page 34 for information about how to flush a task using the Operator Table.

To flush a task using the Flush a Task screen, perform the following steps:

1. Select option **TF** from the Administrative Options Menu to access the Flush a Task screen.

```
node.name                        FLUSH A TASK                              hh:mm
CMD ==>

       SERVER ==> ________

       TASK NUMBERS:
          ==> _______      ==> _______      ==> _______      ==> _______
          ==> _______      ==> _______      ==> _______      ==> _______

       FORCE: ('Y'-YES, 'N'-NO)  FORCE FLUSH A TASK ON A LU 6.2 SESSION MAY
          ==> _                  TERMINATE THE SESSION IMMEDIATELY AND NO
                                 STATISTICS OF THE TASK WILL BE EXCHANGED
```

2. If you are running in a Connect:Direct/Plex environment, type in the 1–8 character Connect:Direct/Plex server name. If you leave this field blank in a Connect:Direct/Plex environment, the Flush Task is performed on the Connect:Direct/Manager.

   Leave this field blank if you are running in a Connect:Direct/Stand-alone Server.

3. Type in the numbers of the tasks you want to flush.
4. In the FORCE field, type a **Y** if you want to force the flush. Type an **N** if you do not want to force the flush. The default is **N**.

   A list of the requested tasks is displayed to indicate a successful flush.

# Displaying System Settings and Status

This section describes how to display system settings, and includes the following topics:

- Displaying the Asset Protection Key File Settings
- Displaying Initialization Parameter Settings
- Modifying Initialization Parameter Settings while Connect:Direct is Running
- Displaying Connect:Direct/Plex Status

## Displaying the Asset Protection Key File Settings

The INQUIRE APFILE command displays the settings in the asset protection key (APKey) file.

**Note:** If necessary, see the *Connect:Direct for z/OS Release Notes* for instructions on how to obtain a temporary or permanent license management key (also known as the asset protection key). The APKey file is required to run Connect:Direct for z/OS.

### Command Format

The INQUIRE APFILE command has the following format.

| Label | Command | Parameter |
|---|---|---|
| (optional) | INQuire APFILE | |

## Issuing the INQUIRE APFILE Command through the IUI

To display the asset protection key file values from the IUI:

1. Select option INQ from the Connect:Direct Administrative Options Menu.

   The Inquire DTF Internal Status screen is displayed.
2. Select the IAPF option.
3. Press ENTER.

   The asset protection key file value settings are displayed, as in the following sample.

```
******************************** Top of Data ********************************
================================================================================
node.name         * Inquire APFILE *  Date: mm.dd.yyyy Time: hh:mm:ss
================================================================================

     APKEY File       : DEVSTG2.HOST.APKEY.FILE3
     Operating System : z/OS
     Product          : Connect:Direct
     Number of Servers: 0032
     Number of SRFS   : 0099
    ** Features **
     Secure-Plus      : Enabled
================================================================================

 APKEY FILE FOR C:D HOST DEVELOPMENT

 The following keywords are the only keywords processed by z/OS, everything e
 PRODUCT , OPERATING-SYSTEM , EXPIRATION-DATE , ACTIVATION-DATE
 CPU-ID , NUMBER-OF-SERVERS , SRFS , EMERGENCY-KEY
```

Scroll down to see additional asset protection key file values.

### Using the INQUIRE APFILE Command from the Batch Interface

To use the INQUIRE APFILE command from the Batch interface:

1. Place your commands in a batch job stream as demonstrated in the *Connect:Direct for z/OS User's Guide*.
2. Submit the job while Connect:Direct is running. The settings are displayed.

**Note:** 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.

## Displaying Initialization Parameter Settings

Use the INQUIRE INITPARM command to view the current global and local initialization parameter settings.

### Command Format

The INQUIRE INITPARM command has the following format and parameter.

| Label | Command | Parameter |
|---|---|---|
| (optional) | INQuire INITparm | WHERE (SERVER=server name) |

The INQUIRE INITPARM parameter is:

| Parameter | Description |
|---|---|
| WHERE (SERVER=server name) | This parameter is optional. This parameter specifies the Connect:Direct/Plex server initialization parameters you want to view. The server name parameter is the 1–8 character name assigned to a Connect:Direct/Server by the CDPLEX.SERVER initialization parameter. If you do not specify this parameter, the Connect:Direct/Manager initialization parameters are displayed. You do not need this parameter for a Connect:Direct/Stand-alone Server. |

### Using the INQUIRE INITPARM Command from the Batch Interface

To use the INQUIRE INITPARM command from the batch interface:

1. Place your commands in a batch job stream as demonstrated in the *Connect:Direct for z/OS User's Guide.*
2. Submit the job while Connect:Direct is running.

**Note:** 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.

3. Verify the results.

The following figure shows a partial sample report.

```
================================================================================
node.name         *INQUIRE INITPARM*  DATE: 06.23.2003  TIME: 12:42:20
                  C:D/Plex  *** MANAGER ** in Group SDWGRP
================================================================================
ABEND.CODES.NODUMP      => (SX37 SX13 U0728 SXD9 S9FC)
ABEND.RUNTASK           => ABEND.CODES.NODUMP
ALLOC.CODES             => (020C 0210 0218 0220 0234)
ALLOC.RETRIES           => 3
ALLOC.WAIT              => 00:00:30
ALLOCATION.EXIT         =>
CDPLEX                  => Yes
CKPT                    => 0
CKPT.DAYS               => 4
CKPT.MODE               => (RECORD   BLOCK   PDS   NOPDS   VSAM   VSAM)
CTCA                    => No
DATEFORM                => MDY
DEBUG                   => '00003001'
DESC.CRIT               => (2)
                        .
                        .
                        .
```

### Issuing the INQUIRE INITPARM Command through the IUI

To display the DTF initialization parameters from the IUI:

1. Select option **INQ** from the Connect:Direct Administrative Options Menu.

   The Inquire DTF Internal Status screen is displayed.

2. If you are running in a Connect:Direct/Plex environment and want to view the initialization parameters of a Connect:Direct/Server, type the server name in the Server field. If you want to view the initialization parameters of a Connect:Direct/Manager, leave the Server field blank.

   If you are running in a Connect:Direct/Stand-alone Server, leave this field blank.

3. Select the **IPRM** option.

4. Press ENTER.

   The current DTF initialization parameter settings are displayed.

## Modifying Initialization Parameter Settings while Connect:Direct is Running

You can update certain initialization parameters in the initialization parameter data set, and then use the MODIFY INITPARMS command to update Connect:Direct with the new initialization parameter settings without restarting Connect:Direct.

You cannot update local initialization parameters with the MODIFY INITPARMS command.

The following list contains the initialization parameters that you can update while Connect:Direct is running:

**Note:** The MODIFY INITPARMS command updates all parameters in the following list, after all parameters (local and global) are parsed for correct syntax.

You cannot update individual parameters with the MODIFY INITPARMS command.

| | | |
|---|---|---|
| ABEND.CODES.NODUMP | ABEND.RUNTASK | ALLOC.CODES |
| ALLOC.RETRIES | ALLOC.WAIT | APDSN |
| CDPLEX.WLM.GOAL | CKPT | CKPT.DAYS |
| CKPT.MODE | CRC | DESC.CRIT |
| DESC.NORM | DESC.TAPE | ECZ.COMPRESSION.LEVEL |
| ECZ.MEMORY.LEVEL | ECZ.WINDOWSIZE | ESF.WAIT |
| EXPDT | GDGALLOC | GDGENQ |
| INVOKE.ALLOC.EXIT | INVOKE.ALLOC.EXIT.ON.RESTART | MAX.AGE |
| MAX.AGE.TOD | MAXRETRIES | MAXSTGIO |
| MULTI.COPY.STAT.RCD | NETMAP.CHECK.ON.CALL | PDSE.SHARING |
| PDSENQ | PRTYDEF | REQUEUE |
| RESET.ORIGIN.ON.SUBMIT | REUSE.SESSIONS | ROUTCDE.CRIT |
| ROUTCDE.NORM | ROUTCDE.TAPE | RUNTASK.RESTART |
| SNMP | SNMP.DSN | SNMP.MANAGER.ADDR |
| SNMP.MANAGER.PORTNUM | STAT.EXCLUDE | SYSOUT |
| TAPE.PREMOUNT | TCQ.THRESHOLD | THIRD.DISP.DELETE |
| TRANS.SUBPAS | WTMESSAGE | WTRETRIES |

See Appendix A, *Global Initialization Parameters* for descriptions of these parameters.

### Command Format

The MODIFY INITPARMS command has the following format.

| Label | Command | Parameter |
|---|---|---|
| (optional) | MODify INITparms | |

This command has no parameters.

### Using the MODIFY INITPARMS Command from the Batch Interface

To use the MODIFY INITPARMS command from the batch interface:

1. Update the initialization parameters in the initialization parameter data set.
2. Place the MODIFY INITPARMS command in a batch job stream.
3. Submit the job while Connect:Direct is running. A message is displayed indicating the results of the refresh action.

**Note:** 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.

### Issuing the MODIFY INITPARMS Command through the IUI

To use the MODIFY INITPARMS command features through the IUI:

1. Update the initialization parameters in the initialization parameter data set.
2. Request option **MD** from the Connect:Direct Administrative Options Menu to access the MODIFY COMMAND screen.

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

 Server ==> _________         00000001     (Current DEBUG Settings)
   MODIFY DEBUG          ==> _________     (nnnnnnnn)
   MODIFY BITS.ON        ==> _________     (nnnnnnnn)
   MODIFY BITS.OFF       ==> _________     (nnnnnnnn)
   MODIFY DDNAME         ==> ___________   (ddname,nn)
   MODIFY CLOSE          ==> _________     (ddname)
   MODIFY MODDIR.TRACE   ==> ___           (YES)
   MODIFY DYN            ==> ____________________________________________________
   MODIFY SESSIONS       ==> _   (Quiesce or Resume)   NODE ==> ________________
   MODIFY NODE.TRACE.ON  ==> ( ________________ ________ )
   MODIFY NODE.TRACE.OFF ==> ________________
   MODIFY INITPARMS      ==> ___           (YES)
```

3. Type **YES** in the MODIFY INITPARMS field.
4. Press **ENTER**. A report is displayed indicating the results of the action.
5. Review the report and perform any corrections if necessary.
6. Press **ENTER** to clear the report.

## Displaying Connect:Direct/Plex Status

The INQUIRE CDPLEX command enables you to display Connect:Direct/Plex status information.

### Command Format

The INQUIRE CDPLEX command has the following format.

| Label | Command | Parameter |
|---|---|---|
| (optional) | INQuire CDPLEX | |

The INQUIRE CDPLEX command has no parameters.

## Using the INQUIRE CDPLEX Command from the Batch Interface

To use the INQUIRE CDPLEX command from the Batch interface:

1. Place your command in a batch job stream, as demonstrated in the *Connect:Direct for z/OS User's Guide*.
2. Submit the job while Connect:Direct is running. A report is displayed. The following figure shows a partial sample report.

**Note:** You must 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.

```
================================================================================
Inquire CDPLEX            10.30.2007 08:23:30
================================================================================
XCF Group Name    : TPXCFGRP   When Activated    : 10.30.2007 08:22:55
SYSPLEX Sys Name  : CSGB       JOB/STC Name      : CD$MGR
Active Servers    :    1       Maximum Servers   :   32

Server            : SERVER1    When Activated    : 10.30.2007 08:22:58
SYSPLEX Sys Name  : CSGB       JOB/STC Name      : CD$SRV1
Active Processes  :    0       Maximum Processes :  250
Server Supports   : CTCA   SNA   IPv4
Server PLEXCLASSES:(A        B        1        *       )
```

## Issuing the INQUIRE CDPLEX Command through the IUI

To display Connect:Direct/Server status from the IUI.

1. Select option INQ from the Connect:Direct Administrative Options Menu. The Inquire DTF Internal Status screen is displayed.
2. Select the IPLX option.
3. Press **ENTER**.

A report is displayed, as demonstrated in the preceding sample report.

# Performing Administrative Tasks

This section provides information about:

- Stopping Connect:Direct
- Suspending and Resuming Processing on Individual Nodes

**Note:** The task of starting Connect:Direct is discussed during the installation procedure in *Connect:Direct for z/OS Installation Guide.*

## Stopping Connect:Direct

The STOP CD command stops Connect:Direct through one of five types of shutdowns:

- Force
- Immediate
- Step
- Quiesce
- Run Task Immediate

This command is usually used for system maintenance.

If you are running Connect:Direct/Plex, you can shut down the individual Connect:Direct/Servers or the entire Connect:Direct/Plex environment. You can also specify if the extended recovery standby system processes the work performed by the system being shut down.

### Command Format

The STOP CD command has the following format and parameters.

| Label | Command | Parameters |
|---|---|---|
| (optional) | STOP CD | [Force \| Immediate \| Quiesce \| Runtaskimm \| Step] |
| | | CDPLEX \| WHERE (SERVER=server name) |
| | | RECOVER |

The parameters for the STOP CD command are:

| Parameter | Description |
|---|---|
| Force | Stops Connect:Direct through a user 4095 ABEND, and produces a dump. Use this option only when problems occur. |

| Parameter | Description |
|---|---|
| Immediate | Terminates all active transmissions immediately after any executing Run Task Processes complete. Connect:Direct writes the statistics record, closes the files, and shuts down. All Processes resume execution when Connect:Direct is reinitialized. If a Process is set for checkpointing and Connect:Direct stops with this parameter, Connect:Direct starts from the last checkpoint and resumes transferring data when the Process resumes.<br>**Note:** You can change how the Immediate parameter interprets the shutdown command by using the IMMEDIATE.SHUTDOWN initialization parameter. If IMMEDIATE.SHUTDOWN=I (the default), an immediate shutdown functions as described in the preceding paragraph. However, if IMMEDIATE.SHUTDOWN=R, an immediate shutdown functions as a runtaskimm shutdown, terminating any executing Run Task Processes before shutting down Connect:Direct.<br>Refer to *IMMEDIATE.SHUTDOWN = I \| R \| (I, nnn \| 60) \| (R, nnn \| 60)* on page 394 for more information on the IMMEDIATE.SHUTDOWN initialization parameter. |
| Quiesce | Enables all active transmissions to run until all executing Process steps complete. No new transmissions are started, and no additional Processes are accepted. All interactive sessions are terminated except for the issuer of the STOP CD command. All active Processes must complete and then you must sign off before Connect:Direct stops. QUIESCE is the default. |
| Runtaskimm | Terminates any Run Task Processes before stopping Connect:Direct. After the Processes are terminated, Connect:Direct writes the statistics record, closes the files, and shuts down. This parameter is provided because the Immediate parameter does not terminate a Run Task until it reaches an interrupt point, such as a checkpoint. A long-running Run Task could delay Connect:Direct shutdown until it completes. When Connect:Direct restarts, if the RUNTASK.RESTART initialization parameter is YES, the checkpoint records for the terminated Run Task restart the Run Task. The only exception to this rule is if the Process is a PNODE/SNODE Process running under SNA intended to run on the PNODE. In this case, the Process restarts, but ends immediately without re-executing the Run Task program. |
| Step | Enables all active transmissions to run until the current Process step of each executing Process finishes. Connect:Direct then writes the statistics records, closes the files, and shuts down. All Processes resume execution when Connect:Direct is reinitialized. |
| CDPLEX | Shuts down the entire Connect:Direct/Plex environment. You cannot use this parameter in a Connect:Direct/Stand-alone Server. |
| WHERE (SERVER=server name) | Specifies which Connect:Direct/Server in a Connect:Direct/Plex environment to shut down. The server name parameter is the 1–8 character name assigned to a Connect:Direct/Server by the CDPLEX.SERVER initialization parameter. You cannot use this parameter in a Connect:Direct/Stand-alone Server.<br>Use this parameter if you only want to shut down a particular Connect:Direct/Server, but leave the rest of the Connect:Direct/Plex environment running. (Use the INQUIRE CDPLEX command described on page 42 to find the name of a server.)<br>**Note:** When you shut down a Connect:Direct/Plex environment, you must specify CDPLEX. If you are shutting down a server, you must also specify WHERE(SERVER=). |

| Parameter | Description |
|---|---|
| RECOVER | Specifies if the extended recovery standby system continues processing work from the system that is shutting down. |

### Stopping Connect:Direct through the Batch Interface

To use the STOP CD command from the batch interface:

1. Place your command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify the results.

The following example stops Connect:Direct, and terminates all transactions immediately.

```
STOP CD  I
```

The following example stops an entire Connect:Direct/Plex environment after all Processes are complete.

```
STOP CD  Q  CDPLEX
```

The following example force stops a Connect:Direct/Server named WALTER, but continues processing on the extended recovery standby system.

```
STOP CD  F CDPLEX WHERE(SERVER=WALTER)  RECOVER
```

### Stopping Connect:Direct through the IUI

Perform the following steps to issue the STOP CD command through the Connect:Direct IUI:

1. Access the STOP Connect:Direct screen by selecting option **SN** from the Connect:Direct Administrative Options Menu.

```
node.name               STOP Connect:Direct                              hh:mm
CMD ==>


         Q ==> Continue active transmissions until the end of process

         S ==> Continue active transmissions until the end of a step

         I ==> Immediately stop all active transmissions
                 (wait for RUN TASKS to complete)

         R ==> Immediately stop all active transmissions
                 (do not wait for RUN TASKS to complete)

         F ==> Force Connect:Direct to stop via an ABEND


                  EXTENDED.RECOVERY and C:D/Plex Options:

 Server ==>  ________ (C:D/Plex server name or '*' for entire C:D/Plex)

 Recover =>  ___      (Yes|No) Should EXTENDED.RECOVERY standby takeover?
```

2. Type one of the five options on the command line.

   STOP CD parameter descriptions are listed beginning on page 44. QUIESCE (**Q**) is the default.

3. If you want to shut down a Connect:Direct/Server in Connect:Direct/Plex environment, type the server name in the SERVER field.

   If you want to shut down the entire Connect:Direct/Plex environment, leave the SERVER field blank.

4. If you want the extended recovery standby system to continue processing work, type Yes in the RECOVER field. Type No or leave the field blank if you do not want the extended recovery standby system to continue processing work.

5. Press **ENTER**.

A shutdown message is displayed for Immediate, Quiesce, and Step shutdowns. No message is displayed for Force shutdowns.

## Suspending and Resuming Processing on Individual Nodes

You can use the MODIFY SESSIONS command to suspend processing on individual nodes, and to resume processing on any suspended nodes.

You can suspend processing on up to a maximum of 15 nodes. You can suspend processing on an individual node because of problems with the node but leave other nodes operating. You can suspend processing if you know that a node will be down for some time.

The MODIFY SESSIONS command has the following format and parameters.

| Label | Command | Parameters |
|---|---|---|
| (optional) | MODIFY | SESSIONS= Quiesce \| Resume<br>(WHERE(NODE=node name \| *)) |

The following table describes the parameters of the MODIFY command.

| Parameter | Description |
|---|---|
| SESSIONS = Quiesce \| Resume<br>(WHERE(NODE=node name \| *)) | 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 Processes that normally execute are placed in the WAIT queue.<br>**Resume** terminates a quiesce state and returns Connect:Direct to normal operation.<br>The WHERE(NODE=) parameter enables you to suspend or resume processing on one or more nodes, to a maximum of 15 nodes. Use this parameter if you want to suspend processing on specific nodes because of problems, but want other nodes to continue processing. You can also use it if you know that a node will be down for some time.<br>The node name subparameter is the 1–16 character local node name specified in the network map of the affected node. You can also specify a partial node name followed by an asterisk (*). For example, the following command suspends processing on all node names that begin with NODE.CHICAGO.<br>SESSIONS=QUIESCE (WHERE(NODE=NODE.CHICAGO*))<br>If SESSIONS=RESUME, you can specify "*" as the node name. Specifying "*" as the node name resumes processing on all individually suspended nodes. Following is an example.<br>SESSIONS=RESUME (WHERE(NODE=*))<br>If you omit the WHERE(NODE=) parameter, the command applies to the entire Connect:Direct system. However, a system-wide RESUME command does not override the processing of any individually suspended nodes. You must issue the SESSIONS=RESUME command with WHERE(NODE=*) to resume processing on individually suspended nodes.<br>**Note:** If the command is issued on an SNODE to quiesce processing with a PNODE, the session with the PNODE is established. However, as soon as the PNODE node name is determined, the session is terminated. No processing of data occurs. |

The following sections describe how to suspend or resume processing on individual nodes.

### Suspending or Resuming Processing on a Node through the Batch Interface

To suspend or resume processing on a node from the batch interface:

1. Place the MODIFY SESSIONS command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify the results by issuing an INQUIRE DEBUG command.

**Note:** For more information about the INQUIRE DEBUG command, see *Displaying DEBUG Settings* on page 366.

The following example suspends processing on all nodes that begin with NODE.CHICAGO.

```
SESSIONS=QUIESCE (WHERE(NODE=NODE.CHICAGO*))
```

The resulting output from the INQUIRE DEBUG command follows.

```
================================================================================
NODE.NEWYORK        *INQ DEBUG/QUIESCE* DATE: 2003.04.01  TIME: 14:51:46
================================================================================

DEBUG            => '00000000'
SESSIONS QUIESCED=>  NODE.CHICAGO1
SESSIONS QUIESCED=>  NODE.CHICAGO2
```

### Suspending or Resuming Processing on a Node through the IUI

Perform the following procedure to suspend or resume processing on a node through the IUI:

1. Request option MD from the Connect:Direct Administrative Options Menu to access the MODIFY COMMAND screen.
2. Type Q in the MODIFY SESSIONS field to suspend processing.

   Type R in the MODIFY SESSIONS field to resume processing on a suspended node.
3. Type the 1–16 character node name in the NODE field.
4. Press **ENTER**.

   The bottom of the screen displays the nodes in the quiesce state.

```
NODE.TRACE.ON----DEBUG
NODE.CHICAGO   QUIESCED
```

# Chapter 3

# Implementing Security

Connect:Direct provides a range of security options to meet diverse security requirements, ranging from no security support to controlling access to all data. These options can be part of Connect:Direct, part of interfaces to other security software, sample exits, or available from user-customized exit routines.

This chapter describes how to plan and install Connect:Direct security, and includes the following topics:

- Overview of Security Options
- Implementing Security Exits
- Connect:Direct Secure Point-of-Entry
- Trusted Node Security
- Security System Requirements
- Configuring Firewall Navigation
- Troubleshooting Security Errors

**Note:** All sample exits provided in Connect:Direct define the proper AMODE and RMODE settings within the source member themselves. All user exits should be link-edited with AMODE=ANY and capable of executing in 31-bit mode. Each user exit should preserve the mode in which it was invoked and return to the caller in the proper mode. Modules written to execute in 31-bit mode can be link-edited with RMODE=ANY or RMODE=24. Check the source for the sample exits to see how Connect:Direct defines the proper AMODE and RMODE settings.

# Overview of Security Options

Connect:Direct for z/OS provides the following security features:

| Security Option | Description |
|---|---|
| Security exits | Secures signon processing, job streams, and application programs. Connect:Direct provides four security exits and includes samples in the sample library. See *Security Exits* on page 53 for more information. |
| SECURITY.EXIT initialization parameter | Specifies a stage 2 security exit. This exit is invoked during signon and Process start and data set access. Signon or file access requests are passed directly to the security exit for authorization checking. See page 410 or more information about the SECURITY.EXIT parameter. |
| Connect:Direct Authorization Facility | Provides signon security and assigns Connect:Direct functional authority if you do not specify or comment out the SECURITY.EXIT initialization parameter. You can use this facility if your installation does not have a security package. See *Connect:Direct Functional Authority* on page 63 for more information.<br>**Note:** The Connect:Direct Authorization Facility provides no data set access security checking. Chapter 4, *Maintaining User Authorization* describes the User Authorization file in detail. |
| Connect:Direct Secure Point-of-Entry | Secures the entry of an outside user to your system. Point-of-entry processing occurs before security exits are called. See *Connect:Direct Secure Point-of-Entry* on page 78 for more information. |
| Trusted Node Security | Enables you to enforce more restrictive security parameters on specific nodes in your network. For example, each adjacent node can be defined as internal or external in its relationship to the local node of that network map. See *Trusted Node Security* on page 80 for more information. |

Connect:Direct supports the following security options:

| Security Option | Description |
|---|---|
| Connect:Direct Secure+ Option | Provides enhanced security for Connect:Direct and is available as a separate component. It uses cryptography to secure data during transmission. You select the security protocol to use with the Secure+ Option product. See the *Connect:Direct Secure+ Option for z/OS Implementation Guide* for more information about Connect:Direct Secure+ Option. |
| CA-ACF2 | External security package that secures files, users, and Connect:Direct functions. |
| RACF (Resource Access Control Facility) | External security package that secures files, users, and Connect:Direct functions. |

| Security Option | Description |
|---|---|
| CA-TOP SECRET | External security package that secures files, users, and Connect:Direct functions. |
| Firewall Navigation | Enables you to control access to a Connect:Direct system running behind a firewall. See *Configuring Firewall Navigation* on page 86 for more information. |

## Security Exits

Connect:Direct provides the following security exits:

- Stage 1 signon security exit
- Stage 2 security exit
- Run Job security exit
- Run Task security exit

The Connect:Direct sample library provides the following security exit routines for use with CA-ACF2, RACF, and CA-TOP SECRET. The High-Level Assembler is required to assemble the sample security exits.

| Exit | Description |
|---|---|
| DMCXSIGN | Stage 1 signon security exit interface |
| DMGACFRJ | RUN JOB security exit |
| DMGACFRT | RUN TASK security exit |
| DMGRACRJ | RUN JOB security exit for SAF or RACF |
| DMGRACRT | RUN TASK exit |
| DMGSAF | Security Exit Stage 2 |
| DMGSAFRT | RUN TASK security interface |

The DMGSAFWA macro provides maps of the security and interface work area used by the security exits. This area allows for information that can be passed between the exits. Connect:Direct has two major processing flows that invoke security exits, the SIGNON command sequence and the Process execution sequence. This section describes how security exits are invoked during these two Processes.

### SIGNON Command Sequence

The SIGNON command sequence is the first flow through which a Connect:Direct terminal user, console operator, or batch application gains access to Connect:Direct functions. During this sequence, one or more of the following control points is invoked:

- Stage 1 signon security exit

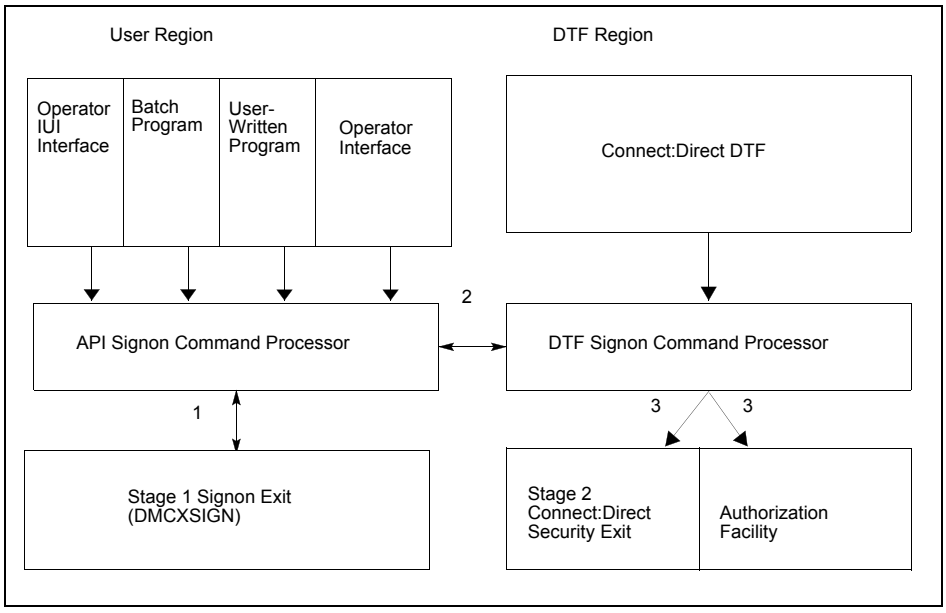
- ✦ Connect:Direct Authorization Facility
- ✦ Stage 2 security exit

### Security during Signon Command

When you execute a SIGNON command through the batch, interactive, or operator interface, security control points exist in the Connect:Direct user region (or API) and the Connect:Direct DTF region.

The stage 1 signon security exit is the initial control point, as shown in the following figure. This optional control point is a user exit that gains control in the region of the user. The exit can inspect and modify the SIGNON command parameters.

The next control point occurs in the DTF region and can be a stage 2 security exit or the Connect:Direct Authorization Facility.




The following SIGNON command flow traces the security flow. The step numbers correspond to the steps in the illustration.

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. For a listing of UICB fields, refer to the chapter on the application programming interface in *Connect:Direct for z/OS User's Guide*.

   If you specify 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 verifies the USERID and PASSWORD that are coded on the SIGNON command for system entry validation and all subsequent security calls.

If you do not specify a password on the SIGNON command, Connect:Direct extracts the USERID from the security system control block that is built for this address space (when the TSO user logged on to TSO or when the BATCH job began execution) and puts that USERID into the UICB.

**Note:** It is important to remember that the stage 1 exit keys off the password, not the user ID. So, if you do not specify a password but do specify a user ID, the stage 1 exit ignores that user ID and overlays it with the address space user ID that is picked up from the security system control block.

When the user ID is moved to the UICB, the exit fills in a special password of IUI, BATCH, or STC, depending upon what environment the signon comes from (Connect:Direct cannot access the password for the address space user ID), and control returns to Connect:Direct.

The benefit of running with a stage 1 signon exit is that Connect:Direct batch jobs do not need hardcoded passwords in their SYSIN data streams.

**Note:** The sample stage 1 exit is shipped, the dummy passwords of IUI, BATCH, and STC are coded in the exit. Change these passwords for each installation to avoid the chance that another site is 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 being assigned by the stage 1 exit. All calls to the security system for verifications verify authorizations by user ID only.

Regardless of how your system is implemented, this processing flow verifies the authority of the requesting 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.

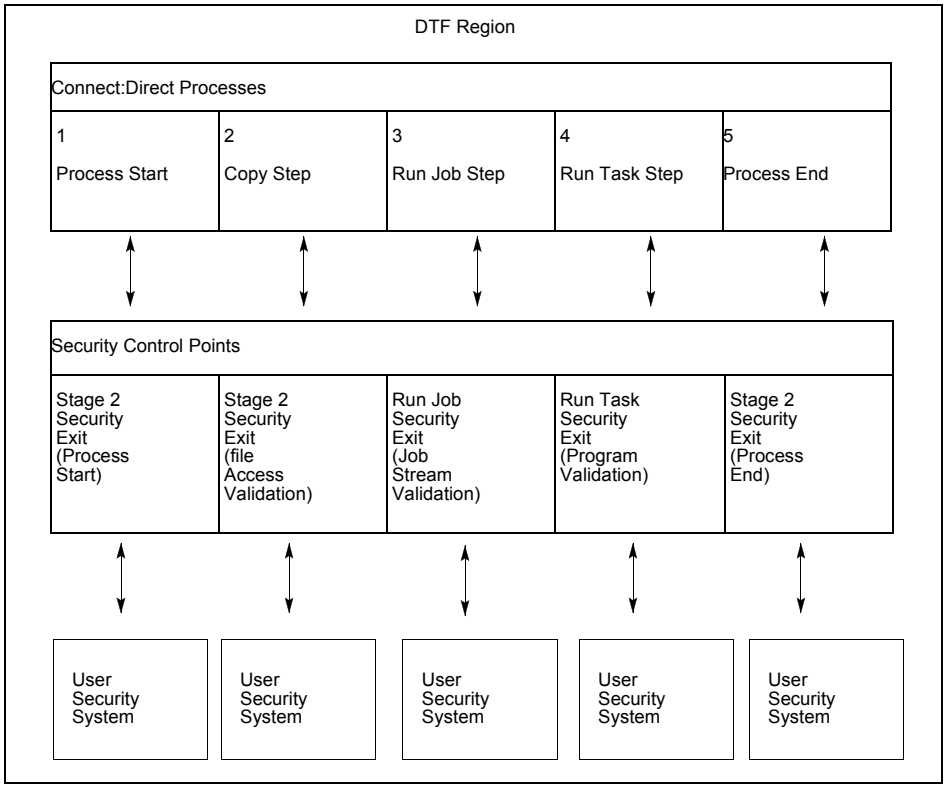
### Process Execution Sequence

The Process execution sequence is the second flow through which Connect:Direct services execute a user request. During this sequence, 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.

### Security during Process Execution

When Connect:Direct executes a Process for a user, several DTF security control points exist, as shown in the following figure:




Refer to the numbers in the illustration as you trace the following Process flow:

1. Process start—This point in the stage 2 security exit gains control whenever a Process begins initial execution or restart execution, and enables 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 or RUN JOB statement is encountered. It enables verification of the access of the user to read or write the file defined in the COPY statement.

   With the RUN JOB statement, the exit enables verification of the user to read the file containing the job stream to be submitted.
3. Run Job—This exit point enables job stream validation and gains control when the following conditions exist:
   - RUN JOB statement is encountered during Process execution
   - RUN.JOB.EXIT initialization parameter is specified
4. Run Task—This exit point enables program validation and gains control when the following conditions exist:
   - RUN TASK statement is encountered during Process execution

- RUN.TASK.EXIT initialization parameter is specified

5. 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, Run Job, and Run Task exit functions are entered for every occurrence of the associated statement in a Connect:Direct Process.

# Implementing Security Exits

This section describes how to implement each type of security exit, and includes information for specific security environments such as, RACF, CA-ACF2, and CA-TOP SECRET.

**Note:** You must have the High-level Assembler to assemble the sample exits.

***Caution:*** To avoid out-of-storage ABENDS in Connect:Direct for z/OS, examine all user exits to verify that all obtained storage is freed. For each GETMAIN that an exit issues, the exit must also issue a corresponding FREEMAIN to avoid accumulating storage; also, if an exit opens a file, you may need to issue a FREEPOOL after the file is closed.

## Stage 1 Signon Security Exit

This control point enables verification of the format and contents of the SIGNON command. The following requirements and restrictions apply:

- Implement the Connect:Direct stage 1 signon exit as an executable load module.
- Name the load module DMCXSIGN.
- Link-edit the module with the attribute AMODE=ANY.
- Link-edit the module as NORENT and NOREUS. Do not specify NCAL.
- Link-edit the module with an authorization code of 1.
- The module must come from an authorized library.
- For the TSO IUI, the module must come from a library in the LNKLST or ISPLLIB.

**Note:** *Do not put the module in a STEPLIB*. The only time a STEPLIB works under ISPF is when ISPLLIB is not allocated.

- For DMBATCH and DMCHLAPI, the module must come from a library in the LNKLST or STEPLIB. Refer to *Special Considerations* on page 260.
- The Connect:Direct SAMPLIB contains a sample source module exit called DMCXSIGN. Edit this module and modify the variable &SECTYPE to reflect the security system in use. Assemble and link-edit the exit.

For RACF or CA-TOP SECRET, use the character string RACF for &SECTYPE. For CA-ACF2, use the character string ACF2.

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.

### Signon Errors

If you are receiving signon errors about the stage 1 exit, allocate the special DDNAME APISECUR in the DMBATCH job for special diagnostic output using one of the following methods.

If you are using the IUI to route trace output to the screen, issue the following TSO command.

```
TSO ALLOC F(APISECUR) DA(*)
```

To route the output to a data set, issue the following TSO command.

```
TSO ALLOC F(APISECUR) SHR DSN('data-set')
```

To route the output to spool, issue the following TSO command:

```
TSO ALLOC F(APISECUR) SYSOUT(*)
```

You must preallocate the data set with the following DCB attributes.

```
DSORG=PS
RECFM=VBA
LRECL=121,
BLKSIZE=125 or greater
```

### Sample SIGNON Panel

The Connect:Direct SAMPLIB contains a member called SGNONPAN that you can use to replace the existing signon panel (DMISGNON) if you implement the stage 1 security exit.

## Stage 2 Security Exit

This control point applies 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 and file access verification. Although you can use it for many different purposes, the stage 2 security exit is designed to provide the interface to your security system. You can also use it to invoke an exit to test new applications and customer connections. For more information, see *Process Exit for Testing (NDMPRCXT)* on page 249.

The following requirements and restrictions apply:

- The Connect:Direct stage 2 security exit is implemented as an executable load module.

- The name of the load module is user-defined, but it cannot conflict with any Connect:Direct load modules.
- Specify the SECURITY.EXIT initialization parameter to activate the stage 2 security exit. This parameter also specifies whether the exit is used for ALL security checking or just DATASET access validation.
- You must link-edit the module as re-entrant and reusable and place it in a load library that the Connect:Direct DTF can access. Do not specify NCAL. For more information, see the *Special Considerations* on page 260.
- To prevent a remote node's security from using Signon dummy passwords, you can use the initialization parameter, REMOTE.DUMMY.PASSWORD. See *REMOTE.DUMMY.PASSWORD=[ YES | INTERNAL ]* on page 406 for more details.
- Because information passed to the exit by Connect:Direct is located above 16 megabyte line, you must link-edit the module with AMODE 31 to make it capable of executing in 31-bit mode.

### Considerations for Systems with UNIX System Services

The following considerations apply to systems with UNIX System Services:

- Access to HFS files is controlled by UNIX System Services. The user ID under which the DTF runs must have UPDATE authority to the BPX.SERVER facility. In addition, the submitter ID/password, the PNODEID/password, or SNODEID/password must be valid. UNIX System Services enables or denies access based on the UNIX permission rules.
- Connect:Direct can also check HFS access without requiring a password. To use a password length of zero, you must set up a RACF profile BPX.SRV.userid in class SURROGAT and make sure that the Connect:Direct started task userid has READ access to this profile. For more information, see *Defining a Surrogate for User IDs with No Password* on page 74.
- UNIX System Service (BPX) calls are executed in the Connect:Direct IUI under the TSO or Batch User ID. BPX calls require that a user ID has an OMVS segment defined to it within the external security product, such as RACF, ACF2 or CA-TOP SECRET. The BPX calls are used to resolve the TCP/IP name or address for reporting purpose in Select Statistics. See *Special Considerations* in *Connect:Direct for z/OS Release Notes* for more detailed information on adding an OMVS segment to a user ID.

### Sample Source Modules

The Connect:Direct SAMPLIB contains sample source modules for several release levels of z/OS security systems. These sample routines 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. Samples are provided for CA-ACF2, RACF, and CA-TOP SECRET. You can accommodate other systems by using the sample code as a model.

DMGSAF is a sample exit routine for all security software supporting the RACROUTE interface. It uses the z/OS Security Access Facility (SAF). The TCB Extensions Feature (TCBSENV) must be present for correct operation.

### Parameters

You can edit the parameters in the DMGSECUR macro to select the appropriate parameters. The parameters are described in each source module and summarized in the following table. The

parameters related to functional authority levels are listed separately in *Connect:Direct Functional Authority* on page 63.

| Parameter | Description |
|---|---|
| TYPE=[SAF] | Identifies the type of exit. |
| STAGE1=[YES,NO] | Identifies whether the stage 1 signon exit is implemented. |
| NOPASS=[YES,NO] | Specifies if data set validity calls to the security subsystem are to be made without using passwords. |
| SECSYS=[ACF,TSS,RACF] | Identifies the security system package installed on your z/OS system. |
| NEWPASS=[YES,NO] | Specifies whether the security system password of a user can be changed at Connect:Direct signon time. |
| PNODEID=[YES,NO] | Specifies if this exit enables security override of a PNODE user ID if one is used in a Connect:Direct Process statement. |
| SNODEID=[YES,NO] | Specifies if this exit enables an incoming node to use a SNODEID. |
| RESTRICT=[YES,NO] | (ACF2 only) Indicates if a user can specify a restricted ID (an ID with no ACF2 password) to access Connect:Direct.<br>When running a stage 1 signon exit, this parameter has no meaning. The stage 1 exit inserts a dummy password into the user security record. In an environment with a stage 1 exit, all data set validity calls to the security subsystem are made with a NOPASS option. Therefore, the security subsystem does not differentiate between a restricted ID and an ID with a valid password. |
| PROTECTD=[NO,YES] | (RACF only) Indicates if a user can specify a protected ID (an ID with no RACF password) to access Connect:Direct. The User IDs are defined to RACF with the NOPASSWORD and NOIDCARD parameters. A protected ID cannot be used in situations requiring a password.<br>Specifying SECSYS=RACF, PROTECTD=YES, allows users to access Connect:Direct using User IDs without passwords when the Connect:Direct DTF is started with a protected User ID. Therefore, Processes may specify PNODEID=\|SNODEID without specifying a password.<br>Specifying SECSYS=RACF, PROTECTD=NO generates no support for RACF Protected User IDs. |
| NPFAIL=[YES,NO] | Specifies whether to refuse a request to copy a file that is not protected. This parameter is only valid for RACF and CA-TOP SECRET users. |
| TEST=[YES,NO] | Specifies if a special debugging option is to be used. If the exit is assembled with TEST=YES specified, and a DD name of SECURITY is added to the Connect:Direct startup job stream, the exit produces a trace of information passed to the exit by Connect:Direct, information passed to the security system, and the feedback from those calls.<br>You can direct the SECURITY DD to SYSOUT or a disk file with attributes of RECFM=VBA, LRECL=121, and BLKSIZE=125 or greater. |

| Parameter | Description |
|---|---|
| CICSID=name | Specifies the dummy USERID name for establishing the initial session between the CICS Interface and a Connect:Direct DTF. Use this parameter for security when using the CICS interface. |
| PASSTK=[YES\|NO] | Results in the generation of a routine in the Stage 2 exit. This routine generates a RACF PassTicket for PNODE processing and receives a PassTicket for SNODE processing. If PASSTK=NO is coded, no RACF PassTicket processing is done. For more information, see *Generating RACF PassTickets*. |
| PROCEXIT=[NDMPRCXT,NO] | Specifies if the NDMPRCXT exit (Process Exit for testing) is to be used or not. To invoke the exit, specify NDMPRCXT. To prevent the NDMPRCXT exit from being invoked, specify NO. For more information, see *Setting Up and Using the NDMPRCXT Exit* on page 251. |
| UID=local ID | Specifies the local identifier which appears in NDMLOG output along with the PTF maintenance listing. |

## Generating RACF PassTickets

A RACF PassTicket is a temporary one-time password that is good for only a short period of time. The generation of the PassTicket requires a Userid and an Application Profile Name. To validate the PassTicket, the same Userid and same Application Profile Name must be used. The Application Profile Name must be defined to RACF as the name of a PTKTDATA profile. Connect:Direct allows the specification of a PassTicket Application ID in the AUTH file.

To identify a node as capable of generating PassTickets, the third parameter in the SECURITY.EXIT initialization parm must specify PSTKT as shown in the following example:

```
SECURITY.EXIT=(module name,DATASET|ALL,PSTKT)
```

If a session is established with another Connect:Direct for z/OS node that also supports PassTicket generation, a PassTicket is generated under the following conditions:

- The PNODE is PassTicket capable.
- The SNODE is PassTicket capable.
- SNODEID= is specified without a password.
- The AUTH file contains an entry for this SNODEID/SNODE and PassTicket information is defined. The Application Profile Name is passed to the Stage 2 security exit to generate the PassTicket.
- The PassTicket is generated using the Application Profile Name and the SNODEID userid.

A generated PassTicket is passed to the SNODE as the Security Password for the SNODEID, along with an indication that a PassTicket is being used. When the SNODE receives a session start with an indication that a PassTicket is being used, it attempts to retrieve the Application Profile Name by looking in the AUTH file for an entry for the SubmitterID/PNODE with the PassTicket information defined. The Application Profile Name and SNODEID userid are used to validate the PassTicket.

PassTickets can also be used to access HFS files.

### Return Codes

The following table describes the valid return codes from the stage 2 exit for signon, Process start, or security delete.

| RC | Description |
|---|---|
| 0 | No error |
| 8 | Insufficient access authority; an SAFB008I is issued |
| 20 | Security system inactive (ACF only); an SAFB020I is issued |

If none of the return codes in the previous table are returned, Connect:Direct issues the message SAFB003I.

**Note:** If SQMSGYES is on, Connect:Direct does not overlay the message ID set by the exit, and the Process ends with the message set by the exit.

The valid return codes for the data set create security call are:

| RC | Description |
|---|---|
| 0 | No error |
| 8 | Insufficient access authority; an SVSA908I ABEND is issued |
| 12 | Invalid data in SQCB; a U2250 ABEND is issued |
| 16 | No storage available for GETMAIN; a U2251 ABEND is issued |
| 20 | Security system inactive; Connect:Direct performs a STOP IMMEDIATE |
| 24 | ADJ node not allowed to send (RACF100I) or receive (RACF101I) and the node executing the exit is PNODE |
| 28 | ADJ node not allowed to send (RACF100I) or receive (RACF101I) and the node executing the exit is SNODE |

After control is returned from the exit to the DTF, the return code is set to 8 if the exit was run from PNODE and to 12 if the exit was run from SNODE.

If none of the return codes in the previous table are returned, Connect:Direct ends abnormally with a U2252 ABEND.

## Connect:Direct Functional Authority

When you sign on to a Connect:Direct running with security (or when a Process you submit begins executing), you are assigned a 20-byte authorization bit mask (ABM) based on a recommendation by the stage 2 security exit or the Connect:Direct Authorization file. The ABM describes your unique functional authority within Connect:Direct.

Connect:Direct provides four standard security levels described in the following table. The ADMVOL, OPRVOL, DBAVOL, and GENVOL parameters indicate the volumes on which these data sets reside. If you do not specify volume names in the DMGSAF stage 2 security exit, the exit provides default volume names for monitoring by your security subsystem.

**Note:** To add new functional authority levels or change the privileges in the standard functional authority levels, see *Customizing Levels of Connect:Direct Functional Authority* on page 66.

The following table describes the functional authority parameters defined in the DMGSAF exit (for other parameters in this exit, see *Parameters* on page 59):

| Parameter | Description |
|---|---|
| ADMDSN=file name<br>ADMVOL=volser | Specifies full administrator authority. The specified user is authorized to execute all Connect:Direct Process language statements and commands. |
| DBADSN=file name<br>DBAVOL=volser | Specifies DB2 Data Base Administrator. |
| OPRDSN=file name<br>OPRVOL=volser | Specifies operator authority. The specified user is authorized to delete, change, display, flush, and submit Processes; stop Connect:Direct; start and stop traces; and display, add, delete, and update type. |
| GENDSN=NULLFILE\|<br>filename<br>GENVOL=volser | Specifies general authority. The specified user is authorized to delete, change, display, and flush his own Processes, submit Processes, and display, add, delete, and update Type.<br>If NULLFILE is coded, a user who logs on to Connect:Direct without specific administrator or operator authorization is, by default, classified as a general user. The following is a sample User Authorization screen, showing commands available to a general user. |

If a bit in one of these standard ABMs is set to one, you are authorized to perform the Connect:Direct command that is associated with that bit, according to the security levels.

For example, if you have the authority to read the ADMDSN, you are given the administrator bit mask that enables you to perform administrator functions. If you do not have ADMDSN authority, OPRDSN read authority is checked, and so on, according to the sequence described in *Functional Authority Validation Sequence* on page 66.

To assign Connect:Direct functional authority, define four data sets or resources on your system to correspond to the administrator, operator, database administrator, and general user data sets.

You can specify Connect:Direct functional authority to individual users by verifying access to one of the named resources. These resource names refer to Connect:Direct functional authority grouped

by the four categories. Connect:Direct users are given access to the particular resource that corresponds to their level of authority.

In addition, you can modify the standard ABMs provided by Connect:Direct to changed the default privileges for a particular functional authority level. See *Customizing Levels of Connect:Direct Functional Authority* on page 66. In addition, you can expand the number of functional authority levels by creating authorization bit masks for new user-defined levels. See *Defining Additional Levels of Functional Authority* on page 71.

### Example Functional Authority Profiles

In the sample screens below, YES next to a command means that the security level is authorized to execute the command, NO means that the security level is not authorized to execute the command, and SUB means that the security level is authorized to execute the command only if the Process was submitted by the particular user.

The following example shows the User Authorization screen for the administration authority (ADMDSN=file name, ADMVOL=volser).

**Note:** You can access the User Authorization screen through the Connect:Direct Primary Options Menu. This menu and its options are discussed in the chapter on the Interactive User Interface in *Connect:Direct for z/OS User's Guide*. The User Authorization screen lists all the commands a particular user is authorized and not authorized to execute.

```
node.name                      USER AUTHORIZATION                          hh:mm
CMD ==>

           AUTH  COMMAND                        AUTH  COMMAND
         --------------------------          --------------------------
         1) YES - CHANGE PROCESS             15) YES - SELECT TASK
         2) YES - DELETE PROCESS             16) YES - SELECT TYPE
         3) YES - DELETE TYPE                17) YES - SELECT USER
         4) YES - DELETE USER                18) YES - SUBMIT PROCESS
         5) YES - FLUSH PROCESS              19) YES - SUBMIT WITHIN PROC
         6) YES - FLUSH TASK                 20) YES - SUSPEND PROCESS
         7) Y/Y - INSERT/UPDATE TYPE         21) YES - STAT COMMAND
         8) Y/Y - INSERT/UPDATE USER         22) YES - EVENT COMMAND
         9) YES - MODIFY (TRACE)             23) YES - VIEW PROCESS
        10) YES - STOP Connect:Direct        24) YES - PERFORM CRC OVERRIDES
        11) YES - UPDATE NETWORK MAP         25) NO  - CONFIRM DELETE
        12) YES - SELECT NETWORK MAP         26) YES - CONFIRM DEL OFF
        13) YES - SELECT PROCESS             27) YES - UPDATE APKEY
        14) YES - SELECT STATISTICS
```

The following example shows the User Authorization screen for the DB2 data base administrator authority (DBADSN=file name, DBAVOL=volser).

```
node.name                    USER AUTHORIZATION                          hh:mm
CMD ==>

          AUTH  COMMAND                        AUTH  COMMAND
        --------------------------          --------------------------
        1) SUB - CHANGE PROCESS             15) SUB - SELECT TASK
        2) SUB - DELETE PROCESS             16) YES - SELECT TYPE
        3) YES - DELETE TYPE                17) NO  - SELECT USER
        4) NO  - DELETE USER                18) YES - SUBMIT PROCESS
        5) SUB - FLUSH PROCESS              19) YES - SUBMIT WITHIN PROC
        6) SUB - FLUSH TASK                 20) SUB - SUSPEND PROCESS
        7) Y/Y - INSERT/UPDATE TYPE         21) NO  - STAT COMMAND
        8) N/N - INSERT/UPDATE USER         22) NO  - EVENT COMMAND
        9) NO  - MODIFY (TRACE)             23) SUB - VIEW PROCESS
       10) NO  - STOP Connect:Direct        24) YES - PERFORM CRC OVERRIDES
       11) NO  - UPDATE NETWORK MAP         25) NO  - CONFIRM DELETE
       12) NO  - SELECT NETWORK MAP         26) NO  - CONFIRM DEL OFF
       13) SUB - SELECT PROCESS             27) NO  - UPDATE APKEY
       14) SUB - SELECT STATISTICS
```

The following example shows the User Authorization screen for the operator authority (OPRDSN=file name, OPRVOL=volser).

```
node.name                    USER AUTHORIZATION                          hh:mm
CMD ==>

          AUTH  COMMAND                        AUTH  COMMAND
        --------------------------          --------------------------
        1) YES - CHANGE PROCESS             15) YES - SELECT TASK
        2) YES - DELETE PROCESS             16) YES - SELECT TYPE
        3) YES - DELETE TYPE                17) NO  - SELECT USER
        4) NO  - DELETE USER                18) YES - SUBMIT PROCESS
        5) YES - FLUSH PROCESS              19) YES - SUBMIT WITHIN PROC
        6) YES - FLUSH TASK                 20) YES - SUSPEND PROCESS
        7) Y/Y - INSERT/UPDATE TYPE         21) NO  - STAT COMMAND
        8) N/N - INSERT/UPDATE USER         22) NO  - EVENT COMMAND
        9) YES - MODIFY (TRACE)             23) YES - VIEW PROCESS
       10) YES - STOP Connect:Direct        24) YES - PERFORM CRC OVERRIDES
       11) NO  - UPDATE NETWORK MAP         25) NO  - CONFIRM DELETE
       12) NO  - SELECT NETWORK MAP         26) NO  - CONFIRM DEL OFF
       13) YES - SELECT PROCESS             27) NO  - UPDATE APKEY
       14) YES - SELECT STATISTICS
```

The following example shows the User Authorization screen for the general user authority (GENDSN=NULLFILE|filename, GENVOL=volser).

```
node.name                    USER AUTHORIZATION                          hh:mm
CMD ==>

           AUTH  COMMAND                        AUTH  COMMAND
         --------------------------         --------------------------
         1) SUB - CHANGE PROCESS            15) SUB - SELECT TASK
         2) SUB - DELETE PROCESS            16) YES - SELECT TYPE
         3) YES - DELETE TYPE               17) NO  - SELECT USER
         4) NO  - DELETE USER               18) YES - SUBMIT PROCESS
         5) SUB - FLUSH PROCESS             19) YES - SUBMIT WITHIN PROC
         6) SUB - FLUSH TASK                20) SUB - SUSPEND PROCESS
         7) Y/Y - INSERT/UPDATE TYPE        21) NO  - STAT COMMAND
         8) N/N - INSERT/UPDATE USER        22) NO  - EVENT COMMAND
         9) NO  - MODIFY (TRACE)            23) SUB - VIEW PROCESS
        10) NO  - STOP Connect:Direct       24) YES - PERFORM CRC OVERRIDES
        11) NO  - UPDATE NETWORK MAP        25) NO  - CONFIRM DELETE
        12) NO  - SELECT NETWORK MAP        26) NO  - CONFIRM DEL OFF
        13) SUB - SELECT PROCESS            27) NO  - UPDATE APKEY
        14) SUB - SELECT STATISTICS
```

## Functional Authority Validation Sequence

The security-checking sequence follows:

1. When the stage 2 security exit is called to determine Connect:Direct functional authority for a user (at signon or Process start), it first checks with the security subsystem (that is, CA-ACF2, RACF, or CA-TOP SECRET) to determine if that user is allowed to read the Administrator data set. If so, the authority of the user is set as an Administrator.
2. If the user is not allowed to read the Administrator data set, the exit checks to see if the user can read the Operator data set. If yes, the user is given Operator authority.
3. If the user is not allowed to read the Operator data set, the exit checks to see if the user can read the Data Base Administrator data set. If so, the user is given Data Base Administrator authority.
4. If the user is not allowed to read the Data Base Administrator data set, and the stage 2 exit includes GENDSN=NULLFILE, the user is given General User authority. If you specify a data set name for GENDSN, the exit either assigns the user General User authority if the user can read the data set, or disables the Connect:Direct function requested (signon or Process execution) if the user cannot read the data set.

## Customizing Levels of Connect:Direct Functional Authority

The privileges set for each of the four standard Connect:Direct functional authority levels are the default privileges provided in the base product. This section describes how to change the privileges in the standard functional authority levels or add new functional authority levels.

You can modify the Connect:Direct stage 2 security exit macro, DMGSECUR, to change the functions a user can perform in a particular authorization level. The Connect:Direct SAMPLIB contains a macro called DMABMFLG that describes each of the 20 bytes of functional authorization.

The following is the generic 20 byte mask that is mapped by a dummy section (DSECT) in the DMABMFLG macro along with a definition of each byte, the general function that the bits represent, and the specific settings:

| Byte | Function | Setting |
|---|---|---|
| BYTE00 | Reserved for future use | |
| BYTE01 | Display, Add, Update, and Delete User Commands | ADDUSR–Add user<br>UPDUSR–Update user<br>DELUSR– Delete user<br>DSPUSR–Display user |
| BYTE02 | Reserved for future use | |
| BYTE03 | Reserved for future use | |
| BYTE04 | Display, Add, Update, and Delete Network Map Commands | ADDNET–Add network map<br>UPDNET–Update network map<br>DELNET–Delete network map<br>DSPNET–Display network map |
| BYTE05 | Change and Delete Process Commands | CHGPRC–Change Process<br>DELPRC–Delete Process |
| BYTE06 | Display Process, Statistics, and Traces, Flush Process, and Use Stats commands | DSPPRC–Display Process<br>DSPSTA–Display Statistics<br>DSPTRC–Display Trace<br>FLSPRC–Flush Process<br>STATCMD–Use Statistics Commands |
| BYTE07 | Start/Stop Connect:Direct, Start/Stop Traces, Modify Init parms, Suspend/Resume Sessions, Use Event Services, and Update APKey commands | STPNDM–Start/Stop Connect:Direct<br>SSTRAC–Start/Stop Traces and Modify Initparms<br>EVENTCMD–Use Event Services Commands<br>UPDKEY–Update license key |
| BYTE08 | Reserved for future use | |
| BYTE09 | Display, Add, Update, and Delete Type commands | ADDTYP–Add type<br>UPDTYP–Update type<br>DELTYP–Delete type<br>DSPTYP–Display type |
| BYTE10 | Use COPY, RUN JOB, MODALS, and SUBMIT Statements, and View Process and CRC Override commands | GCOPY–Use COPY statement<br>GRUNJ–Use RUN JOB statement<br>GMODALS–Use MODAL statement<br>GSUBMIT–Use SUBMIT statement<br>VIEWPR–View Process<br>GOVCRC–Perform CRC overrides |

| Byte | Function | Setting |
|---|---|---|
| BYTE11 | Use Submit within a Process and RUN TASK statements, display Confirm Delete prompt, and turn Confirm Delete prompt off for a session.<br>**Note:** Note: The Confirm Delete function also includes the Flush and Suspend commands, that is, the user is prompted to confirm before the Flush and Suspend Commands in addition to the Delete command. | GSUB–Use Submit within a Process statement<br>GRUNT–Use RUN TASK statement<br>GCDEL–Display Confirm Delete, Flush, and Suspend prompts<br>GCDELOFF–Turn off Confirm delete prompt off for session |
| BYTE12 | General User Functions– Select, Delete, Flush, Change, and View Process, and Display Statistics, and Display Plex environment (the last command for an Administrator only).<br>**Note:** The General User functions enable you to restrict applying each command to Processes associated with a submitter ID. | GDSPPRC–Display Process<br>GDELPRC–Delete Process<br>GDFLSPRC–Flush Process<br>GDSPSTA–Display Statistics<br>GCHGPRC–Change Process<br>GVIEWPR–View Process<br>DSPPLX–Display Plex Environment |
| BYTE13 | Reserved for future use | |
| BYTE14 | Reserved for future use | |
| BYTE15 | Reserved for future use | |
| BYTE16 | Reserved for future use | |
| BYTE17 | Reserved for future use | |
| BYTE18 | Reserved for future use | |
| BYTE19 | Reserved for future use | |

The sample exit macro DMGSECUR contains authorization bit masks for the four standard Connect:Direct authority groups. The default settings shown in the following ABMs are in the DMGSECUR macro in the $CD.SAMPLIB data set. The DMABMFLG bit mask contains all possible functions for each byte whereas the bit masks for a particular Connect:Direct authority group may contain only a subset of the available functions. For example, BYTE 10 (DBA10) in the DB2 data base authority level authorization bit mask (DBAABM) below does not contain the View Process function (VIEWPR) while BYTE 10 in the ABM for the Operator authority level does. (Bytes reserved for future use are not shown.)

The following example shows the authorization bit mask for the Administrator authority level (ADMABM).

```
ABYTE1   DC    AL1(ADDUSR+UPDUSR+DELUSR+DSPUSR)
ABYTE4   DC    AL1(ADDNET+UPDNET+DELNET+DSPNET)
ABYTE5   DC    AL1(CHGPRC+DELPRC)
ABYTE6   DC    AL1(DSPPRC+DSPSTA+FLSPRC+STATCMD)
ABYTE7   DC    AL1(STPNDM+SSTRAC+EVENTCMD+UPDKEY)
ABYTE9   DC    AL1(ADDTYP+UPDTYP+DELTYP+DSPTYP)
ABYTE10  DC    AL1(GCOPY+GRUNJ+GMODALS+GSUBMIT+VIEWPR+GOVCRC)
ABYTE11  DC    AL1(GSUB+GRUNT)
ABYTE12  DC    AL1(DSPPLX)
```

The following example shows the authorization bit mask for the Operator authority level (OPERABM).

```
OPER1    DC    XL1(00)              NULL - Not Set
OPER4    DC    XL1(00)              NULL - Not Set
OPER5    DC    AL1(CHGPRC+DELPRC)   DELETE/CHANGE PROCESS
OPER6    DC    AL1(DSPPRC+DSPSTA+FLSPRC) DISPLAY/FLUSH PROCESS
*                                        DISPLAY STATISTICS
OPER7    DC    AL1(STPNDM+SSTRAC)   STOP START-STOP TRACE
OPER9    DC    AL1(ADDTYP+UPDTYP+DELTYP+DSPTYP)
*                                   DISPLAY/ADD/DELETE TYPE
OPER10   DC    AL1(GCOPY+GRUNJ+GMODALS+GSUBMIT+VIEWPR+GOVCRC)
*                                   COPY/RUN JOB/MODALS/SUBMIT
OPER11   DC    AL1(GSUB+GRUNT)      REMOTE SUBMIT/RUN TASK
```

The following example shows the authorization bit mask for the DB2 data base authority level (DBAABM).

```
DBA1     DC    XL1(00)              NULL - Not Set
DBA4     DC    XL1(00)              NULL - Not Set
DBA9     DC    AL1(ADDTYP+UPDTYP+DELTYP+DSPTYP)
*                                   DISPLAY/ADD/DELETE TYPE
DBA10    DC    AL1(GCOPY+GRUNJ+GMODALS+GSUBMIT+GOVCRC)
*                                   COPY/RUN JOB/MODALS/SUBMIT
DBA11    DC    AL1(GSUB+GRUNT)      REMOTE SUBMIT/RUN TASK
DBA12    DC    AL1(GDSPPRC+GDELPRC+GFLSPRC+GDSPSTA+GCHGPRC+GVIEWPR)
*                                   DISPLAY/CHANGE/FLUSH/STATS FOR
*                                   SUBMITTERS PROCESS ONLY
```

The following example shows the authorization bit mask for the General User authority level (GUSRABM).

```
GUSR1    DC    XL1(00)              NULL - Not Set
GUSR4    DC    XL1(00)              NULL - Not Set
GUSR9    DC    AL1(ADDTYP+UPDTYP+DELTYP+DSPTYP)
*                                   DISPLAY/ADD/DELETE TYPE
GUSR10   DC    AL1(GCOPY+GRUNJ+GMODALS+GSUBMIT+GOVCRC)
*                                   COPY/RUN JOB/MODALS/SUBMIT
GUSR11   DC    AL1(GSUB+GRUNT)      REMOTE SUBMIT/RUN TASK
GUSR12   DC    AL1(GDSPPRC+GDELPRC+GFLSPRC+GDSPSTA+GCHGPRC+GVIEWPR)
*                                   DISPLAY/CHANGE/FLUSH/STATS FOR
*                                   SUBMITTERS PROCESS ONLY
```

To change the bits in any given authorization byte, locate the bit labels in the DMABMFLG macro and update the DMGSECUR macro. To implement any changes made and put your new exit into effect, you must stop and restart Connect:Direct.

### Example 1–Broadening Privileges for General Users

To authorize general users to perform a SELECT PROCESS command and a SELECT STATISTICS command for Processes submitted with any user ID rather than just with the ID of the submitter, perform the following steps:

1. Look in the DMABMFLG macro for the bits that allow the user to perform these two commands. These bits are located in BYTE06 of DMABMFLG.
2. Find the bits that allow these two commands only for the ID of the submitter. These bits are located in BYTE12 of DMABMFLG.
3. Locate the label in the DMGSECUR macro that indicates the General User authorization bit mask (GUSRABM). General user BYTES 06 and 12 are currently set to the following values.

```
GUSR6  DC XL1'00'
GUSR12 DC AL1(GDSPPRC+GDELPRC+GFLSPRC+GDSPSTA+GCHGPRC+GVIEWPR
```

4. To allow general users to perform SELECT PROCESS and SELECT STATISTICS commands for any user ID, remove GDSPPRC and GDSPSTA from BYTE12 and copy the DSPPRC and DSPSTA bits from BYTE06 in the DMABMFLG authorization bit mask and put them in BYTE06 in the GUSRABM, changing these bytes to the following values:

```
GUSR6   DC  AL1(DSPPRC+DSPSTA)
GUSR12  DC  AL1(GDELPRC+GFLSPRC+GCHGPRC+GVIEWPR)
```

5. Reassemble and link-edit your security module that uses the DMGSECUR macro.
6. To put the new exit into effect, stop and restart Connect:Direct.

### Example 2–Forcing the Confirm Prompt for General Users

If a user has the authority to delete, flush or suspend a Process, the default setting allows the user to perform the action automatically. As soon as the user enters the command, it is executed instantly. However, you can modify this default privilege and require a user to confirm the action before it is executed. In addition, you can specify whether a user can turn off the Confirm Delete/Flush/Suspend Command prompt for a particular session after the prompt displays at least one time.

The following sample procedure shows you how to turn on the Confirm Delete/Flush/Suspend Command prompt for users in the general user authority category but at the same time allow them to turn off the prompt for a particular session.

1. Locate BYTE11 in the DMABMFLG macro. Two of the four bits, GCDEL and GCDELOFF, turn on the Confirm Delete/Flush/Suspend Command prompt and if turned on, permit a user to turn off the Confirm Delete/Flush/Suspend Command prompt temporarily for the current session. (The other two bits pertain to the Submit within a Process and RUN TASK commands.)

2. Locate the label in the DMGSECUR macro that indicates the General User authorization bit mask setting (GUSRABM). General user BYTE 11 is currently set to the following values.

```
GUSR11 DC AL1 (GSUB+GRUNT)
```

3. To ensure that the Confirm/Delete/Suspend Command prompt displays for all users in the general user category, add GCDEL to change BYTE11 as follows:

```
GUSR11 DC AL1 (GSUB+GRUNT+GCDEL)
```

4. To let users in the general user category turn off the Confirm/Delete/Suspend Command prompt for a particular session, add GCDELOFF to change BYTE11 as follows:

```
GUSR11 DC AL1 (GSUB+GRUNT+GCDEL+GCDELOFF)
```

5. Reassemble and link-edit your security module that uses the DMGSECUR macro.
6. To put the new exit into effect, stop and restart Connect:Direct.

## Defining Additional Levels of Functional Authority

Connect:Direct provides additional authorization bit masks (US0DSN through US9DSN) that you can use to expand the number of functional authority levels beyond the standard four levels. The following example shows the authorization bit mask for the user-definable ABM.

```
U0BYTE1  DC    XL1(00)              NULL - Not Set
U0BYTE4  DC    XL1(00)              NULL - Not Set
U0BYTE5  DC    AL1(CHGPRC+DELPRC)   DELETE/CHANGE PROCESS
U0BYTE6  DC    AL1(DSPPRC+DSPSTA+FLSPRC) DISPLAY/FLUSH PROCESS
*                                        DISPLAY STATISTICS
U0BYTE7  DC    AL1(STPNDM+SSTRAC)   STOP START-STOP TRACE
U0BYTE8  DC    XL1'00'              NOT USED
U0BYTE9  DC    AL1(ADDTYP+UPDTYP+DELTYP+DSPTYP)
*                                   DISPLAY/ADD/DELETE TYPE
U0BYTE10 DC    AL1(GCOPY+GRUNJ+GMODALS+GSUBMIT+VIEWPR+GOVCRC)
*                                   COPY/RUN JOB/MODALS/SUBMIT
U0BYTE11 DC    AL1(GSUB+GRUNT)      REMOTE SUBMIT/RUN TASK
```

To create your own authorization level, locate the user-definable ABM you want to use, and change the bytes. Add the data set and volume names to the DMGSAF stage 2 security exit. Assemble and link-edit the DMGSAF exit. To implement the new authorization level, into effect, you must stop and restart Connect:Direct.

### Example 1–Assigning Read-Only Authority to a User Authorization Level

To define a new security profile to allow read-only authority for one or more users, follow this sample procedure:

1. Modify the DMGSECUR macro by locating the USR0ABM label and making the following changes:
   a. Delete both the CHGPRC and DELPRC bits in BYTE05.
   b. Delete the FLSPRC bit in BYTE06.
   c. Delete both the STPNDM and SSTRAC bits in BYTE07.
   d. Delete all bits in BYTE09.
   e. Delete all bits in BYTE10 except for VIEWPR.
   f. Delete all bits in BYTE11.

   The USR0ABM should look like the following:

```
USR0ABM  DS    0XL20       DEFINES User Group Zero ABM Flags
U0BYTE0  DC    XL1'00'              NOT USED
U0BYTE1  DC    XL1'00'              NOT USED
U0BYTE2  DC    XL1'00'              NOT USED
U0BYTE3  DC    XL1'00'              NOT USED
U0BYTE4  DC    XL1'00'              NOT USED
U0BYTE5  DC    XL1'00'              DELETE/CHANGE PROCESS
U0BYTE6  DC    AL1(DSPPRC+DSPSTA)   Display Process
*                                   DISPLAY STATISTICS
U0BYTE7  DC    XL1'00'              STOP START-STOP TRACE
U0BYTE8  DC    XL1'00'              NOT USED
U0BYTE9  DC    XL1'00'              DISPLAY/ADD/DELETE TYPE
U0BYTE10 DC    AL1(VIEWPR)          View Process only
U0BYTE11 DC    XL1'00'              REMOTE SUBMIT/RUN TASK
U0BYTE12 DC    XL1'00'              NOT USED
```

2. Modify the DMGSAF example in $CD.SAMPLIB to assign a file name to the new US0DSN parameter and indicate which volume it resides on.

```
DMGSAF   DMGSECUR TYPE=SAF,                                          X
               .                                                     X
               .                                                     X
               ADMDSN=$CD.ADMIN,                                     X
               ADMVOL=VOLSER,                                        X
               OPRDSN=$CD.OPER,                                      X
               OPRVOL=VOLSER,                                        X
               DBADSN=$CD.DBA,                                       X
               DBAVOL=VOLSER,                                        X
               GENDSN=$CD.GUSER,                                     X
               GENVOL=VOLSER,                                        X
               US0DSN=$CD.NEW.USER.LEVEL,                            X
               US0VOL=VOLSER
```

3. Assemble and link-edit the DMGSAF module using the sample JCL in $CD.SAMPLIB(ASMSTG2).

```
//ASM      EXEC PGM=ASMA90,
//             PARM='OBJECT,NODECK,XREF(SHORT),RENT,USING(WARN(0),NOMAPX
//             ),FLAG(NOCONT),SYSPARM(GEN),NOTEST'
//SYSIN    DD DISP=SHR,DSN=connect.direct.samplib(DMG****)
//SYSLIB   DD DISP=SHR,DSN=connect.direct.samplib
//         DD DISP=SHR,DSN=SYS1.MACLIB
//         DD DISP=SHR,DSN=SYS1.AMODGEN
//         DD DISP=SHR,DSN=SYS1.AMACLIB
//         DD DISP=SHR,DSN=security.maclib
//SYSLIN   DD DISP=(,PASS),DSN=&&OBJ,
//            UNIT=SYSDA,SPACE=(CYL,(1,1)),
//            DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//************************
//*        LKED          *
//************************
//LKED     EXEC  PGM=IEWL,COND=(0,LT,ASM),
// PARM=('SIZE=(256K,13K),LIST,LET,XREF,AMODE=ANY,RMODE=24,RENT',
// 'REUS')
//SYSLIB   DD  DISP=SHR,DSN=connect.direct.linklib
//         DD  DISP=SHR,DSN=security.loadlib
//SYSLIN   DD  DISP=(OLD,DELETE),DSN=&&OBJ
//SYSLMOD  DD  DISP=SHR,DSN=connect.direct.linklib(DMG****)
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1),,CONTIG)
```

4. If necessary, update the Connect:Direct initialization parameter, SECURITY.EXIT, to specify the new exit.

```
SECURITY.EXIT = (mod-name,ALL)
```

5. Initialize Connect:Direct in the normal manner.

   When a user signs on to Connect:Direct, they are assigned an authorization bit mask that allows them to display and view Processes, and display statistics but they cannot submit or run a Process.

### Example 2–Defining a New Adminstrator Level

To define a new security profile giving the administrator all normal adminstrator functions except the ability to run Connect:Direct Processes, follow this procedure. Authorization BYTES 10 and 11 represent the functions that are to be disallowed. If you have used USR0DSN to define another level, use USR1DSN for this new profile.

1. Modify the DMGSECUR macro by locating the USR1ABM label and updating the various bytes as follows:

```
ADMABM   DC    0XL20
ABYTE0   DC    XL1'00'              NOT USED
ABYTE1   DC    AL1(ADDUSR+UPDUSR+DELUSR+DSPUSR)
ABYTE2   DC    XL1'00'              NOT USED
ABYTE3   DC    XL1'00'              NOT USED
ABYTE4   DC    AL1(ADDNET+UPDNET+DELNET+DSPNET)
ABYTE5   DC    AL1(CHGPRC+DELPRC)
ABYTE6   DC    AL1(DSPPRC+DSPSTA+FLSPRC+STATCMD)
ABYTE7   DC    AL1(STPNDM+SSTRAC+EVENTCMD+UPDKEY)
ABYTE8   DC    AL1(UPDNCR+DSPNCR)
ABYTE9   DC    AL1(ADDTYP+UPDTYP+DELTYP+DSPTYP)
ABYTE10  DC    AL1(VIEWPR)
ABYTE11  DC    XL1'00'
ABYTE12  DC    AL1(DSPPLX)
ABYTE13  DC    XL1'00'
ABYTE14  DC    XL1'00'              NOT USED
```

2. Modify the DMGSAF example in $CD.SAMPLIB to define the new US1DSN parameter with the new security profile as follows:

```
DMGSAF   DMGSECUR TYPE=SAF,                                    X
               .                                               X
               .                                               X
               ADMDSN=$CD.ADMIN,                               X
               ADMVOL=VOLSER,                                  X
               OPRDSN=$CD.OPER,                                X
               OPRVOL=VOLSER,                                  X
               DBADSN=$CD.DBA,                                 X
               DBAVOL=VOLSER,                                  X
               GENDSN=$CD.GUSER,                               X
               GENVOL=VOLSER,                                  X
               US0DSN=$CD.NEW.USER.LEVEL,                      X
               US0VOL=VOLSER,                                  X
               US1DSN=$CD.NEW.ADMIN,                           X
               US1VOL=VOLSER
```

3. Assemble and link-edit the DMGSAF module using the sample JCL in $CD.SAMPLIB(ASMSTG2).
4. If necessary, update the Connect:Direct initialization parameter, SECURITY.EXIT, to specify the new exit.
5. Initialize Connect:Direct in the normal manner.

## Defining a Surrogate for User IDs with No Password

Use the BPX.SERVER profile to set the scope of z/OS resources that the server can access when acting as a surrogate for its clients. BPX.SERVER UPDATE access lets the server establish a thread level (task-level) security environment for clients connecting to the server. When the RACF identity of the application server is granted UPDATE authority to BPX.SERVER in the RACF FACILITY class, the server can act as a surrogate for the client.

This procedure contains sample IBM RACF commands. For more information, refer to IBM RACF manuals. For more information about how to define SURROGAT in other external security products, such as ACF2 or CA-TOP SECRET, refer to the manuals of the specific vendor.

1. Make sure that the Stage 2 Security exit can verify if Stage 1 has set the dummy password in SQCB. The DMGSECUR macro contains label STG1NPW which includes the following instruction:

```
OI SQFLAG,SQDUMMY DUMMY PASSWORD USED P768101
```

2. Identify all user IDs that will access HFS without supplying their password.
3. To activate the SURROGAT class support in RACF, if it has not already been set up on your system, issue the following command:

```
SETROPTS CLASSACT(SURROGAT)
```

**Note:** You only have to activate this feature one time.

4. If you want to cache the SURROGAT profiles in storage to enable you to refresh and immediately put all RACF changes in effect immediately, issue the following command:

```
SETROPTS RACLIST(SURROGAT)
```

**Note:** If you do not use the RACLIST option, the changes made during this procedure will not take effect until the next IPL.

5. To create the SURROGAT class profile for a particular user, issue the following command:

```
RDEFINE SURROGAT BPX.SRV.UUUUUUUU UACC(NONE)
```

where *UUUUUUUU* is the user ID you are creating a profile for.

6. Repeat Step 5 for each user ID that requires HFS support without a password with a SURROGAT profile.

**Note:** To define all users in one command, you can specify BPX.SRV.* .

7. To give a user the authority to create a thread-level security environment for another user, issue the following command:

```
PERMIT BPX.SRV.UUUUUUUU CLASS(SURROGAT) ID(CDIRECT) ACCESS(READ)
```

where the Connect:Direct DTF user called *CDIRECT* is the user you are granting permission to create the security environment for another user called *UUUUUUUU*.

8. Repeat Step 8 for each user ID that requires HFS support without a password with a SURROGAT profile.

**Note:** To define all users in one command, you can specify BPX.SRV.* .

9. Verify that the Connect:Direct DTF User ID has sufficient access to HFS files along with both RACF access and UNIX System Services permissions.
10. If you are using the RACLIST option, issue the following command to refresh and put your changes in effect for the SURROGAT class:

```
SETROPTS RACLIST(SURROGAT) REFRESH
```

11. To check whether the Connect:Direct DTF Userid has been defined to the BPX.SRV.*uuuuuuuu* SURROGAT class profile, use the following RLIST command:

```
RLIST SURROGAT BPX.SRV.uuuuuuuu AUTHUSER
```

where *uuuuuuuu* is the user ID whose requests Connect:Direct needs to process.

The system displays the user ID (which should be the Connect:Direct DTF Userid) and access rights of the user ID that can act as a surrogate for *uuuuuuuu*.

***Caution:*** Be aware of the REMOTE.DUMMY.PASSWORD and Adjacent Node settings for Node to Node communication.

```
SAFB022I – DMABMB - Dummy password usage by Adjacent Node rejected.
           An attempt was made by an Adjacent Node to use a dummy
           password to authorize access to the Connect:Direct local
           node. If the Init Parm REMOTE.DUMMY.PASSWORD setting is
           INTERNAL, only Adjacent Nodes having the INTERNAL
           attribute in the Netmap may use a dummy password for
           this purpose.
```

## Run Job Security Exit

The Run Job security exit control point provides a standard interface for security verification of job streams before they are submitted to the job entry system. Specific implementation details include the following:

- The Connect:Direct Run Job exit is implemented as an executable load module.
- The name of the load module is user-defined and cannot conflict with any Connect:Direct load module names.
- Specify RUN.JOB.EXIT=(modname) in the Connect:Direct initialization parameters to activate the Run Job exit.
- You must link-edit the module as re-entrant and place it in a load library that the Connect:Direct DTF can access.

- Because information passed to the exit by Connect:Direct is located above the 16 megabyte line, you must link-edit the module with AMODE ANY to make it capable of executing in 31-bit mode.

For additional information about exits, see the *Special Considerations* section of Chapter 10, *Using Connect:Direct Exits*.

### Sample Run Job Security Exits

The Connect:Direct SAMPLIB contains a sample source module for the most used z/OS security systems. Sample exit routines are:

- DMGRACRJ for RACF and CA-TOP SECRET
- DMGACFRJ for CA-ACF2

The sample exits are designed to ensure that correct security information is coded on each JOB statement in the job stream.

- For RACF and CA-TOP SECRET, a check is made for a valid USER and PASSWORD on each JOB card. If not found, a USER=submitter keyword is added to each JOB card.
- For CA-ACF2, a JOBFROM=submitter keyword is added immediately following each JOB card to ensure that the correct security information is transferred to each submitted job:

If you use one of these exits without coding a value for the RUN.JOB.EXIT initialization parameter, Connect:Direct does not use the default for the RUNJOBID initialization parameter.

---

**Note:** Use the Run Job security exit to achieve *user propagation* for security checks when the job that executes is submitted by the user ID assigned to Connect:Direct rather than the user ID that submitted the job. **In most environments, this Exit is not needed.**

---

## Run Task Security Exit

The Run Task security exit control point provides a standard interface to verify that the user is authorized to run the specified program. Connect:Direct passes the exit 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-defined, but cannot conflict with any Connect:Direct load module names.
- Specify RUN.TASK.EXIT=(modname) in the Connect:Direct initialization parameters to activate the Run Task exit.
- You must link-edit the module as re-entrant and place it in a load library that the Connect:Direct DTF can access.
- Because information passed to the exit by Connect:Direct is located above the 16 megabyte line, you must link-edit the module with AMODE ANY to make it capable of executing in 31-bit mode.

For additional information about exits, see the *Special Considerations* section of Chapter 10, *Using Connect:Direct Exits*.

### Sample Run Task Security Exit

The Connect:Direct SAMPLIB contains a sample source module for the most used security systems. Sample exit routines are:

- DMGRACRT for RACF and CA-TOP SECRET
- DMGACFRT for CA-ACF2
- DMGSAFRT if using the Security Access Facility

You can use the sample exit as a model to implement specific requirements.

## Connect:Direct Secure Point-of-Entry

You need security on the local node because adjacent nodes need access to local nodes in order to transfer files. Connect:Direct administrators have three options for security on transfers initiated at a remote node:

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

Point-of-entry security secures the entry of an *outside* user to your system. It works with your current security setup (including all current exits) to provide additional security that addresses concerns about users from other nodes knowing a user ID and password combination on your system. Both data protection and Connect:Direct functional authority are accomplished with exits.

Point-of-Entry Processing is internal within Connect:Direct, and happens prior to calling the security exit for validations.

### Point-of-Entry Concept

When a Process is submitted by another node, the receiving Connect:Direct node has access to the user ID of the person who submitted the Process and the name of the node of the submitted Process.

For example, the local node is CD.HOUSTON, and a user SMITH submits a Process on CD.CHICAGO to copy a file to CD.HOUSTON. By placing an entry of SMITH/CD.CHICAGO into the local Connect:Direct authorization file, the security administrator for CD.HOUSTON can associate this user with a valid user ID and password on the local system. The Connect:Direct Authorization file has the following values:

```
USERID          = SMITH
NODE            = CD.CHICAGO
SECURITY ID     = JONES
SECURITY PSWD   = DALLAS
```

In this scenario, when user ID SMITH on node CD.CHICAGO submits a Process to run with node CD.HOUSTON, the functional authority and the data set validation for that Process are done under the authority of user ID JONES, which is a valid user ID on CD.HOUSTON. The user from the

Chicago node never needs to know the related valid user ID and password on the CD.HOUSTON node.

```
USERID          = SMITH
NODE            = CD.HOUSTON
SECURITY ID     = JONES
SECURITY PSWD   = DALLAS
```

## Optional Variations

Note the following variations:

- If the CD.HOUSTON node enables SNODEID overrides and user SMITH puts an SNODEID parameter in his Process, the Authorization file is not checked and the translation of the user ID and password is not done.

  **Note:** If the INVOKE.SPOE.ON.SNODEID initialization parameter is set to YES, then the Authorization file is checked and the user ID and password are translated.

  For example, if the incoming Process in the previous example is coded with SNODEID=(BROWN,PWB), even if it is submitted by SMITH from CD.CHICAGO, the CD.HOUSTON node validates security with the authority of user ID BROWN, not user ID JONES.

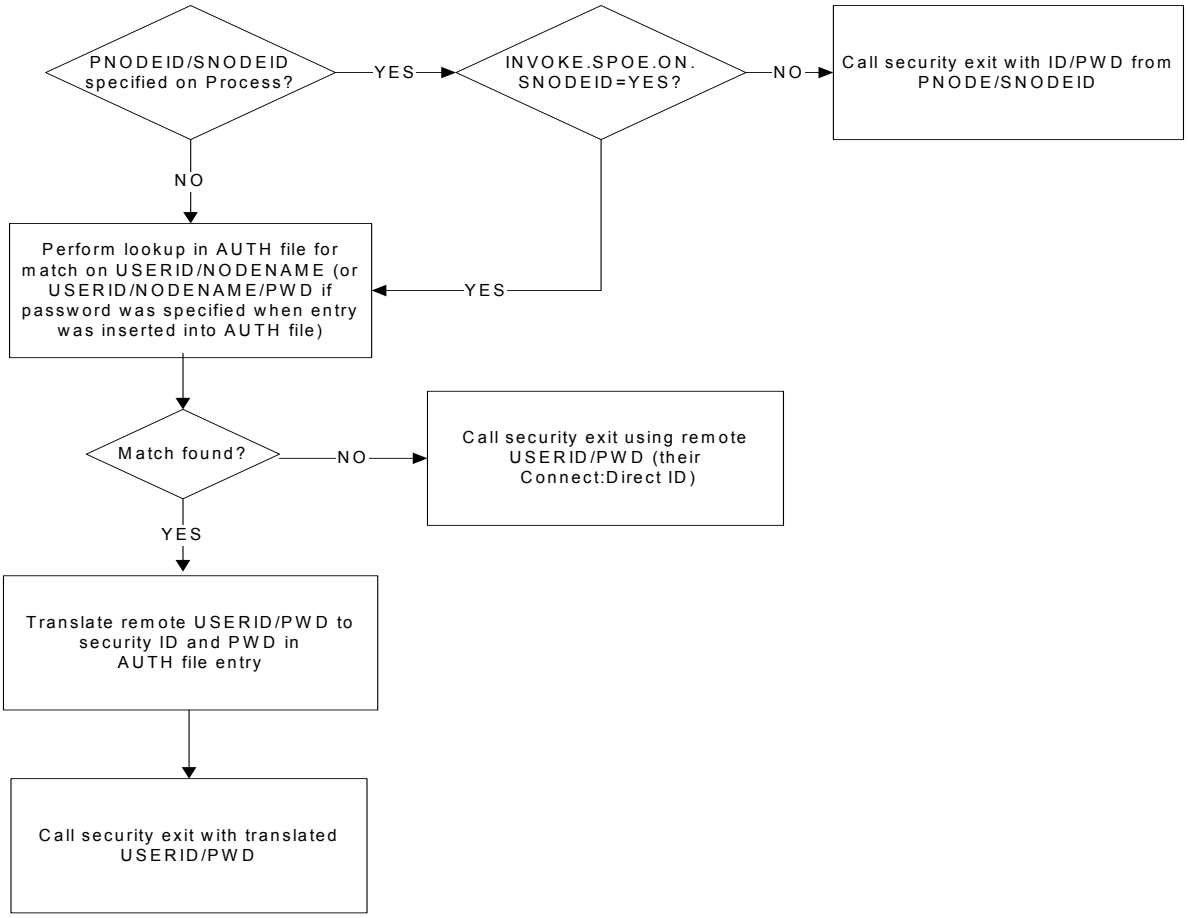
  **Note:** To produce a completely secure point-of-entry security system, disable SNODEID overrides. To disable SNODEID overrides, specify SNODEID=NO in your stage 2 security exit.

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

  For instance, you can specify JONES/DALLAS as the user ID and password for all users coming into your node from CD.CHICAGO. In addition, if you are running a stage 1 signon exit, you can specify the security password for all users as IUI, BATCH, or STC, and avoid the need to update the Authorization file as the password changes.

- Connect:Direct for OpenVMS is not able to pass an OpenVMS password with the OpenVMS user ID. If you are using secure point-of-entry with incoming OpenVMS nodes, you must leave the User Password field blank in your z/OS authorization file, or all incoming OpenVMS Processes will fail.

The following figure illustrates the flow of security checking for secure point-of-entry:




### Implementing Secure Point-of-Entry

These security checks are made automatically for every incoming Process, so no parameter is needed to activate point-of-entry security. To implement point-of-entry, add user ID and node name combinations to your Authorization file. For instructions on manipulating the Connect:Direct Authorization file, see Chapter 4, *Maintaining User Authorization*.

The security exit determines if a secure point-of-entry translation was performed on a user ID by checking the bit SQIDXLAT of the SQCB control block (DMFSQCB macro).

## Trusted Node Security

The Trusted Node Security feature enables you to enforce more restrictive security parameters when dealing with specific nodes in your network, enabling you to define each adjacent node in the network map as *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 administrator can code the security exit to 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:

| Parameter | Description |
|---|---|
| EXTERNAL\|EXT | Indicates that the node is external. |
| INTERNAL\|INT | Indicates that the node is internal. This value is the default. |

## Cross-Domain Signon Environment

The cross-domain signon environment is an extension of the Trusted Node Security feature. This feature enables 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. You can then modify the security exit 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.

## 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 enables you 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 descriptions follow:

| Parameter | Description |
|---|---|
| RECEIVE\|RECV | Indicates that when the adjacent node initiates a transfer, it is only allowed to receive data from this node. It is never allowed to send data to this node. |

| Parameter | Description |
|---|---|
| SEND | Indicates that when the adjacent node initiates a transfer, it is only allowed to send data to this node. It is never allowed to receive data from this node. |
| BOTH | Indicates that when the adjacent node initiates a transfer, it is allowed to both send and receive data from this node. This value is the default. |
| NONE | Indicates that when the adjacent node initiates a transfer, it is neither allowed to send or receive data from this node. |

### Examples

The following 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))
```

The following two bits are identified in the security exit:

| Bit | Description |
|---|---|
| SQSNODE | Identifies 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. |
| SQIDXLAT | 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. |

# 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 sections describe these considerations.

## CA-ACF2 Environment

When assembling both the stage 1 signon exit and the stage 2 security exit, you must provide the following data definition (DD) statements.

1. For the assembly step, ensure that the SYSLIB concatenation contains the following information:

```
//SYSLIB   DD     DSN=ACF.MACLIB
//         DD     DSN=$CD.SAMPLIB
//        DD     DSN=SYS1.MODGEN
//         DD     DSN=SYS1.MACLIB
```

2. Replace $CD with the appropriate high-level qualifier for your Connect:Direct data sets.
3. For the link-edit step, provide the following DD statements.

```
//SYSLIB   DD     DSN=SYS1.ACFMOD
//         DD     DSN=SYS1.ACFAMOD
//         DD    DSN=$CD.LINKLIB
```

**Note:** You must have the High-Level Assembler for correct assembly. Do not specify NOALIGN as an option. The correct option is ALIGN.

4. Specify the Connect:Direct DTF logon ID (LID) with the following attributes:

| LID Attribute | Comment |
|---|---|
| MUSASS | Required for ACF2. |
| NON-CNCL | Only required if you are not running DMGSAF. |
| NO-SMC | Required. |
| SECURITY | Only required if NEWPASS=YES is specified for the stage 2 security exit. |
| ACCOUNT | Only required if NEWPASS=YES is specified for the stage 2 security exit. |
| JOBFROM | Only required if the Run Job statement is allowed and the Run Job exit is active. |
| STC | Required if Connect:Direct is run as a started task. |
| RESTRICT | Optional. |
| PROMPT | Optional. Enables Connect:Direct to receive prompts from the operating system. For example, with the PROMPT attribute, Connect:Direct receives a prompt for the password of a password protected data set if it was not supplied in the COPY statement DSN=filename/password. |

If you are executing Connect:Direct as a started task, CA-ACF2 monitors started tasks and the Connect:Direct logon ID specifies STC=YES. See the GSO OPTS field STC/NOSTC in the *CA-ACF2 Administrator's Guide* for more information.

If you are using the program-pathing facility of CA-ACF2 that requires that the user logon ID be defined with the RESTRICT and SUBAUTH attributes, then the program name specified in the PROGRAM attribute for the user logon ID must be BPXPTATT for Connect:Direct authorization.

The SAF interface requires definitions (SAFDEF) for both BPXPTATT and DMGRUNT.

## RACF Environment

When assembling both the stage 1 signon exit and the stage 2 security exit, you must provide the following DD statements.

1. For the assembly step, ensure that the SYSLIB concatenation contains the following information:

```
//SYSLIB   DD     DSN=$CD.SAMPLIB
//        DD     DSN=SYS1.MODGEN
//         DD     DSN=SYS1.MACLIB
```

2. Replace $CD with the appropriate high-level qualifier for your Connect:Direct data sets.
3. For the link-edit step, provide the following DD statement.

```
//SYSLIB   DD     DSN=$CD.LINKLIB
```

**Note:** You must have Assembler H or the High Level Assembler for correct assembly. Do not specify NOALIGN as an option. The correct option is ALIGN.

4. Observe the following restrictions or requirements:
   - If the z/OS Task Control Block (TCB) Extension Feature is installed, give Connect:Direct update authority to Connect:Direct system files, such as the TCQ and network map. These prerequisites allow Connect:Direct to use the Security Access Facility (SAF) of z/OS.
   - If you are using the RACF PROGRAM ACCESS authority to set up access authority by program name and user ID, define the program name DMGATTIS to RACF for Connect:Direct.

### Program Access to Data Sets (PADS)

If your system has UNIX System Services, or if you use PADS functionality in your security system, include all data sets in the Connect:Direct JCL STEPLIB DD concatenation in your Program Control List (PCL). If any data set in the STEPLIB concatenation is not in the PADS list, the “dirty bit” will be turned on, and Connect:Direct initialization will fail and display message SITA997I. Use the following procedure:

1. Type the following command to display the data sets (libraries) in the PCL.

```
rlist program *
```

The access-controlled data sets are displayed.

```
CLASSNAME
----------------
PROGRAM  *

MEMBER     CLASS    NAME
---------  ------   ------
PMBR

DATA SET NAME                VOLSER     PADS CHECKING
--------------------------   -------    ----------------
CEE.SCEERUN                               NO
TCPIP.SEZALINK                            NO
USER01.HOST4100.LOADLIB                   NO
```

2. Define the Connect:Direct libraries to Resource Access Control Facility (RACF) PADS. The following screen is an example of a definition.

```
RDEFINE PROGRAM ** UACC(READ) ADDMEM +
                   ('$CD.LOADLIB'//NOPADCHK)
SETROPTS WHEN(PROGRAM) REFRESH
```

**Note:** Refer to the RACF documentation to verify RACF command formats and keywords.

## CA-TOP SECRET Environment

When assembling both the stage 1 signon exit and the stage 2 security exit, you must provide the following DD statements.

1. For the assembly step, ensure that the SYSLIB concatenation contains the following information:

```
//SYSLIB   DD     DSN=$CD.SAMPLIB
//         DD     DSN=SYS1.MODGEN
//         DD     DSN=SYS1.MACLIB
```

2. Replace $CD with the appropriate high-level qualifier for your Connect:Direct data sets.
3. For the link-edit step, provide the following DD statements.

```
//SYSLIB   DD   DSN=$CD.LINKLIB
```

**Note:** You must have Assembler H or the High-Level Assembler for correct assembly. Do not specify NOALIGN as an option. The correct option is ALIGN.

4. Add Connect:Direct as a CA-TOP SECRET Facility.
5. Observe the following restrictions or requirements:
   - If you are using CA-TOP SECRET Release 4 or later, issue the following commands.

```
TSS  CREATE(NDM)   NAME('...')   DEPT(...)
     MASTFAC(NDM)  FAC(STC)      PASSWORD(NOPW)
TSS  ADDTO(STC)    PROC(NDM)     ACID(NDM)
TSS  PERMIT(NDM)   DSN(NDM)      ACCESS(ALL)
```

Issue TSS MODIFY commands to obtain the following list of attributes.

```
NDM PGM=DMG        ID=your choice
ATTRIBUTES=ACTIVE, SHRPRF, ASUBM, MULTIUSER, NOXDEF,
      SIGN(M) NORNDPW NOAUDIT, RES, NOABEND,
      NOPROMPT, NOTSOC
```

- If you are using CA-TOP SECRET Release 4 or later, and if the z/OS TCB Extension Feature is installed, Connect:Direct only needs update authority to its system files, such as TCQ and network map. These prerequisites allow Connect:Direct to use the SAF of z/OS. Otherwise, the ACID referenced previously must provide full access authority to all files Connect:Direct accesses. Alternatively, you can identify Connect:Direct in the privileged program name table as having access to all files by setting bit 6 (bypass password checking) in the program properties table (IEFSDPPT) to **1**.

# Configuring Firewall Navigation

Firewall navigation enables controlled access to a Connect:Direct system running behind a packet-filtering firewall without compromising your security policies or those of your trading partners. You control this access by assigning a specific TCP or UDT source port number or a range of source port numbers with a specific destination address (or addresses) for Connect:Direct sessions.

Before you configure source ports in the Connect:Direct initialization parameters, you need to review the information in this section, especially if you are implementing firewalls for UDT.

To implement firewall navigation:

1. Coordinate IP address and associated source port assignment with your local firewall administrator before updating the firewall navigation record in the initialization parameters file.
2. Add the following parameters to the Connect:Direct initialization parameters file as needed, based on whether you are using TCP or UDT:
   - TCP.SRC.PORTS
   - TCP.SRC.PORTS.LIST.ITERATIONS
   - UDP.SRC.PORTS
   - UDP.SRC.PORTS.LIST.ITERATIONS

In a Connect:Direct/Plex environment, specify these parameters in the local initialization parameters file of the Connect:Direct/Plex member that communicates with an external firewall.

3. Reinitialize Connect:Direct for z/OS.
4. Coordinate the specified port numbers with the firewall administrator at the remote site. These ports must also be available for Connect:Direct communications on the firewall of your trading partner.

## Firewall Navigation

Firewall rules need to be created on the local firewall to allow the local Connect:Direct node to communicate with the remote Connect:Direct node. A typical packet-filtering firewall rule specifies that the local firewall is open in one direction (inbound or outbound) to packets from a particular protocol with particular local addresses, local ports, remote addresses, and remote ports. Firewall navigation differs between TCP and UDT; as a result, firewall rules for TCP and UDT should be configured differently.

### TCP Firewall Navigation Rules

In the following table, the TCP rules are presented in two sections: the first section applies to rules that are required when the local node is acting as a PNODE; the second section applies to rules that are required when the local node is acting as an SNODE. A typical node acts as a PNODE on some occasions and an SNODE on other occasions; therefore, its firewall will require both sets of rules.

| TCP PNODE Rules | | | |
|---|---|---|---|
| **Rule Name** | **Rule Direction** | **Local Ports** | **Remote Ports** |
| PNODE session | Outbound | Local C:D's source ports | Remote C:D's listening port |
| **TCP SNODE Rules** | | | |
| **Rule Name** | **Rule Direction** | **Local Ports** | **Remote Ports** |
| SNODE session | Inbound | Local C:D's listening port | Remote C:D's source ports |

### UDT Firewall Navigation Rules

UDT firewall rules are applied to the UDP protocol. The recommended default firewall rule for UDP packets is to block packets inbound to the local system *and* outbound from the local system to prevent the confusion that could occur due to the callback feature of UDT session establishment.

In the following table, the UDT rules are presented in two sections: the first section applies to rules that are required when the local node is acting as a PNODE; the second section applies to rules that are required when the local node is acting as an SNODE. A typical node acts as a PNODE on some occasions and an SNODE on other occasions; therefore, its firewall will require both sets of rules.

| UDT PNODE Rules | | | |
|---|---|---|---|
| **Rule Name** | **Rule Direction** | **Local Ports** | **Remote Ports** |
| PNODE Session Request | Outbound | Local C:D's source ports | Remote C:D's listening port |
| PNODE Session | Outbound | Local C:D's source ports | Remote C:D's source ports |
| **UDT SNODE Rules** | | | |
| **Rule Name** | **Rule Direction** | **Local Ports** | **Remote Ports** |
| SNODE listen | Inbound | Local C:D's listening port | Remote C:D's source ports |
| SNODE session | Inbound | Local C:D's source ports | Remote C:D's source ports |

## Firewall Configuration Examples

In the firewall configuration examples for TCP and UDT, the following IP addresses and source ports will be used:

**Note:** The IP addresses in the examples have been chosen to be distinctive and are not intended to be valid IP addresses.

- The **local node** has IP address 222.222.222.222 and listening port 2264. Its source ports for communicating with the remote node are 2000–2200.
- The **remote node** has IP address 333.333.333.333 and listening port 3364. Its source ports for communicating with the local node are 3000–3300.

**Note:** See *Session Establishment* on page 90 for a discussion of the differences between UDT and TCP session establishment.

### TCP Firewall Configuration Example

The Connect:Direct administrator configures the **local node** to listen on port 2264, and the following initialization parameter settings are used to configure the local node's source ports:

- TCP.SRC.PORTS = (333.333.333.333, 2000–2200)
- TCP.SRC.PORTS.LIST.ITERATIONS = 1

This configuration specifies to use a source port in the range 2000–2200 when communicating with the remote node's address 333.333.333.333 and to search the port range one time for an available port. The local node will act as both a PNODE and an SNODE when communicating with the remote node.

Based on this scenario, the firewall rules for the local node are the following:

| Rule Name | Rule Direction | Local Ports | Remote Ports |
|---|---|---|---|
| PNODE session request | Outbound | 2000–2200 | 3364 |
| SNODE session | Inbound | 2264 | 3000–3300 |

## UDT Firewall Configuration Example

The Connect:Direct administrator configures the **local node** to listen on port 2264, and the following initialization parameter settings are used to configure the local node's source ports:

- UDP.SRC.PORTS = (333.333.333.333, 2000-2200)
- UDP.SRC.PORTS.LIST.ITERATIONS = 1

This configuration specifies to use a source port in the range 2000–2200 when communicating with the remote node's address 333.333.333.333 and to search the port range one time for an available port. The local node will act as both a PNODE and an SNODE when communicating with the remote node.

Based on this scenario, the firewall rules for the local node are the following:

| Rule Name | Rule Direction | Local Ports | Remote Ports |
|---|---|---|---|
| PNODE session request | Outbound | 2000–2200 | 3364 |
| PNODE session | Outbound | 2000–2200 | 3000–3300 |
| SNODE listen | Inbound | 2264 | 3000–3300 |
| SNODE session | Inbound | 2000–2200 | 3000–3300 |

### Blocking Outbound Packets

The recommended default rule for outbound UDP packets from the local system is to block the packets. If you do not follow this recommendation, port usage may, at first sight, appear to violate the firewall's inbound rules.

An example will help illustrate this situation. Suppose that in the example in the previous section:

- The local node is the SNODE.
- The default outbound rule allows all outbound UDP packets from the local system.
- The "SNODE session" rule is accidently omitted.

Because of the callback feature of UDT session establishment, SNODE sessions are still likely to succeed on ports 2000–2200. This may cause confusion because ports 2000–2200 are blocked to inbound UDP packets.

If you use the recommended default outbound rule and apply the PNODE and SNODE rules described in the previous section, there will be no confusion about which port to use, and the UDT callback feature will function as designed, thus supporting reliability.

## Session Establishment

Session establishment differs between TCP and UDT; these differences affect how you set up firewall rules and configure the firewall navigation initialization parameters in Connect:Direct.

### TCP Session Establishment

A Connect:Direct TCP client contacts a Connect:Direct TCP server on its listening port. The Connect:Direct client scans the list of ports (specified using the **TCP.SRC.PORTS** initialization parameter) and looks for a port to bind to. The number of times Connect:Direct scans the list is specified using the **TCP.SRC.PORTS.LIST.ITERATIONS** initialization parameter. If Connect:Direct finds an available port, communication with the remote node proceeds.

### UDT Session Establishment

When a Connect:Direct UDT client contacts a Connect:Direct UDT server on its listening port to request a session, the UDT server responds with a different server port to use for the session. The client attempts to contact the server on the session port. The Connect:Direct client scans the list of ports (specified in the **UDP.SRC.PORTS** initialization parameter) and looks for an available port to bind to. The number of times Connect:Direct scans the list is specified using the **UDP.SRC.PORTS.LIST.ITERATIONS** initialization parameter. If the Connect:Direct client finds an available port, communication with the remote Connect:Direct server proceeds. If a session cannot be established after a certain time interval, the server atempts to contact the client.

### Common Problems in Establishing a Session

The following message indicates that Connect:Direct cannot find an idle port.

```
SCPA001I - TGT.ADDR=nnn.nnn.nnn.nnn, TCP.SRC.PORTS exhausted
```

If this message occurs frequently, increase the pool of available ports.

You may need to reserve ports in TCP/IP to ensure they are available for firewall navigation. Reserve ports with the PORT statement in IBM TCP/IP. An example follows:

```
PORT
   5000 TCP connect-jobname   NOAUTOLOG  ; Connect:Direct Firewall pool
   5001 TCP connect-jobname   NOAUTOLOG  ; Connect:Direct Firewall pool
   5002 TCP connect-jobname   NOAUTOLOG  ; Connect:Direct Firewall pool
   5003 TCP connect-jobname   NOAUTOLOG  ; Connect:Direct Firewall pool
   5004 TCP connect-jobname   NOAUTOLOG  ; Connect:Direct Firewall pool
   5005 TCP connect-jobname   NOAUTOLOG  ; Connect:Direct Firewall pool
```

A pool of available ports that is too small can affect performance because the outbound connections are limited.

# Troubleshooting Security Errors

Security errors can show up at signon, at Process start, or at any step of a Process. In general, a return code of 8 means that the error occurred on the PNODE, and a return code of C means the error occurred on the SNODE. This section tells you how to determine the cause of security errors.

Many Connect:Direct security-related messages begin with the prefix RACF. This fact does not mean that RACF was necessarily involved with the failure. It is merely a naming convention for Connect:Direct message identifiers.

Often, it is helpful to run a security trace to determine exactly where and why a security failure occurred. See *Security Traces* on page 364, for information on security traces.

The following pages list possible security errors, error messages, probable causes, actions to take, and data to collect.

## Condition: Signon Denied

When you sign on from either batch or the IUI, you receive the a message that indicates the Stage 1 Signon Exit has failed.

| Error | Cause | Action | Collect |
|---|---|---|---|
| RACF097I | The Stage 1 Signon exit, DMCXSIGN, cannot be found.<br>In the Connect:Direct for z/OS IUI, verify that DMCXSIGN is in the LINKLIST or in ISPLLIB of your signon CLIST and not in a STEPLIB. | Review both the short text and long text Connect:Direct messages. If you are receiving the message during signon to the IUI, run a batch job after verifying that DMCXSIGN is available to the job (through STEPLIBs, the linklist, or globals). For either batch or interactive signon, allocate SECURITY as described in *Security Traces* on page 364. You will be able to view the progression of BLDLs, along with output showing where Connect:Direct looked for DMCXSIGN and the results from the search. | ◆ Output written to the SECURITY DD when you allocated SECURITY as described in *Security Traces* on page 364. |

### Condition: Lack Authority to Perform Connect:Direct for z/OSFunction

You attempt to perform a Connect:Direct for z/OSfunction but receive a message that says you are not authorized to perform that function.

| Error Messages | | | | | |
|---|---|---|---|---|---|
| SCBB001I | SCBC030I | SCBD001I | SCBE001I | SCBF001I | SCBF063I |
| SCBF064I | SCBG001I | SCBH001I | SCBI001I | SCBJ001I | SCBK005I |
| SCBL001I | SCBN001I | SCBO001I | SCBP001I | SCBR002I | SCBS001I |
| SCBT005I | SCBU003I | SCBV001I | SCBW001I | SCBX001I | SCBY001I |
| SCPA008I | SFIA002I | SFIA003I | SRJA014I | SRTA008I | SSUB100I |

| Cause | Action | Collect |
|---|---|---|
| If you are running a Stage 2 security exit, your user ID is defined using an authorization bit mask that does not include the function you are attempting. A security trace will show you the general category of Connect:Direct user assigned to your userid (administrator, operator, or general user). See *Security Traces* on page 364 for more information about how to initiate a security trace. If you are using the Connect:Direct authorization file, the functional authority of your userid does not include the function you are trying to perform. | Review both the short text and long text Connect:Direct messages. Have the Connect:Direct administrator at your site ensure that your userid has the authority necessary to perform the function, either by updating your userid record in the Connect:Direct authorization file or by assigning the authority. within the Stage 2 security exit | ◆ Output from a security trace showing the validation of your authority to perform the Connect:Direct function |

### Condition: Access Denied to File or Data Set on COPY Step

You are denied access to a data set or a file on a COPY step.

| Error | Cause | Action | Collect |
|---|---|---|---|
| RACF095I | The security subsystem either on your node (RC=8) or the remote node (RC=C) has denied your userid access to the data set. | Review both the short text and long text Connect:Direct messages. Ensure that your userid has the correct access to the data set. If you continue getting this message, run a security trace. See *Security Traces* on page 364 for more information about how to initiate a security trace. It might be necessary to use a PNODEID or SNODEID statement to send a valid userid and password to the security system. | ◆ Output from a security trace |

### Condition: User Record not Found in the Authorization Data Set

When you sign on to Connect:Direct or submit a Process to another node, you receive message SAFA002I, *The user record was not found in the Authorization Data Set.*

| Error | Cause | Action | Collect |
|---|---|---|---|
| SAFA002I | If you are using the Connect:Direct authorization file for security, be aware that the key to that file is a combination of userid and node name. For example, if you are signed on to node CDA with userid USERA and transmitting to node CDB (not using an SNODEID override), the authorization file on CDB must have an entry for the userid USERA and node CDA. | Review both the short text and long text Connect:Direct messages. Check the appropriate Connect:Direct authorization file and verify that the correct userid/node combination is specified. User records in the Connect:Direct authorization file can be added or modified with the Insert User or Update User commands. | ◆ None |

# Chapter 4

# Maintaining User Authorization

This chapter includes the following topics:

- Overview of the Authorization Facility
- Adding or Updating User Information in the Authorization File
- Deleting Users from the Authorization File
- Selecting User Information from the Authorization File

**Note:** You can also use the Connect:Direct Browser to perform some of the procedures in this chapter. To learn more about the Connect:Direct Browser, see the *Connect:Direct Browser User Interface User's Guide* on the Connect:Direct Browser CD-ROM or online in the Sterling Commerce Documentation Library.

## Overview of the Authorization Facility

The Connect:Direct Authorization Facility controls access to Connect:Direct functions. It is an alternative source of security information to the Stage 1 Signon and Stage 2 Security exits. If you use the Connect:Direct Authorization Facility, you must identify all Connect:Direct users in all nodes that execute Processes.

The following example shows how the Connect:Direct Authorization Facility is used. This example includes two Connect:Direct nodes, called SYSTEMA and SYSTEMB. Joe has access to SYSTEMA under the Connect:Direct user ID of JOEA and access to SYSTEMB under the Connect:Direct user ID of JOEB.

Joe requires two entries in the Connect:Direct Authorization Facility of each system, as illustrated in the following tables. These entries give him access to Connect:Direct on both systems and the authorization to move files between both systems.

| SYSTEMA Authorization FIle | | | |
|---|---|---|---|
| **Node** | **Connect:Direct User ID** | **Connect:Direct: Password** | **Authorized Functions** |
| SYSTEMA | JOEA | [pswd] | Y,Y,N,Y . . . . . |
| SYSTEMB | JOEB | [pswd] | N,Y . . . . . |

| SYSTEMB Authorization FIle | | | |
|---|---|---|---|
| **Node** | **Connect:Direct User ID** | **Connect:Direct: Password** | **Authorized Functions** |
| SYSTEMA | JOEA | [pswd] | Y,Y,N,Y . . . . . |
| SYSTEMB | JOEB | [pswd] | N,Y . . . . . |

The combination of logical node name and user ID is used to access the Authorization file on the remote node to obtain the user ID, password, and associated functional authority.

For example, if Joe sent a file from SYSTEMA to SYSTEMB, the combination of SYSTEMA and JOEA enables him to access the authorization file on SYSTEMB. This entry then determines what Connect:Direct functional authority Joe has on SYSTEMB when coming from SYSTEMA.

**Note:** The Connect:Direct password is optional, but if specified in the Authorization Facility, you must also specify it on the SIGNON command. You must make the password available at Process execution time through the signon or SNODEID override.

# Authorization File

The Authorization file contains user attribute default records. Each record defines which Connect:Direct features the user can access for each node.

Individual users can access the User Authorization screen to display information about their own authorization record. See the *Connect:Direct for z/OS User's Guide* for a screen example and field descriptions.

The following table describes the User Authorization file maintenance commands:

| Command | Description |
|---|---|
| INSERT USER | Inserts a User record in the Authorization file. |
| UPDATE USER | Updates a User record in the Authorization file. |
| DELETE USER | Deletes a User record from the Authorization file. |
| SELECT USER | Selects a User record from the Authorization file. |

You can execute these commands through the batch interface, the Interactive user interface (IUI), or the operator interface.

## Adding or Updating User Information in the Authorization File

The INSERT USER and UPDATE USER commands add or update a user in the Connect:Direct Authorization file. The commands have the following format and parameters. The required parameters and keywords are in bold print. (The NAME parameter is required only for INSERT USER.) Default values for parameters and subparameters are underlined.

| Label | Command | Parameters |
|---|---|---|
| (optional) | **INSert USER \| UPDate USER** | **USERID = (nodename, user ID)** |
| | | **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 |
| | | APKEY = Y \| N |
| | | CASE = Y \| N |
| | | CDEL = Y \| N<br>See Note. |
| | | CDELOFF = Y \| N<br>See Note. |

**Note:** Valid only in the Interactive user interface.

| Label | Command | Parameters |
|---|---|---|
| | | CHange = Y \| N |
| | | COPY = Y \| N |
| | | DELPR = Y \| N |
| | | EVENTCMD = Y \| N |
| | | FLUSH = Y \| N |
| | | GEN.CHG.PROCESS = Y \| N<br>See Note. |
| | | GEN.DEL.PROCESS = Y \| N<br>See Note. |
| | | GEN.FLS.PROCESS = Y \| N<br>See Note. |
| | | GEN.SEL.PROCESS = Y \| N<br>See Note. |
| | | GEN.SEL.STATISTICS = Y \| N<br>See Note. |
| | | GVIEW = Y \| N<br>See Note. |
| | | MAXSA = max signon attempts |
| | | MODALS = Y \| N |
| | | MODIFY = Y \| N |
| | | NSUBMIT = Y \| N |
| | | OVCRC = Y \| N |
| | | PASSword = initial password |
| | | PHone = 'phone number' |
| | | PTICDATA = (APPL profile name, secured signon key) |
| | | RESETSA |
| | | RUNJOB = Y \| N |
| | | RUNTASK = Y \| N |
| | | SECUREWR=Y\|N |
| | | SECURITY = (security id, security pswd) |
| | | SELNET = Y \| N |
| | | SELPR = Y \| N |

**Note:** Valid only in the Interactive user interface.

| Label | Command | Parameters |
|---|---|---|
| | | SELSTAT = Y \| N |
| | | STATCMD = Y \| N |
| | | STOPCD = Y \| N |
| | | SUBMIT = Y \| N |
| | | SUBMITTER.CMDS = ( Y \| N, Y \| N, Y \| N, Y \| N, Y \| N )<br>**Note:** This parameter is valid only in the batch interface. |
| | | UPDNET = Y \| N |
| | | VIEW PROCESS = Y \| N |

**Note:** Valid only in the Interactive user interface.

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

| Parameter | Description |
|---|---|
| USERID = (nodename, user ID) | Specifies the user node and user ID of the record being added or updated.<br>nodename specifies the user node of the User record. It is a 1–16 character alphanumeric string.<br>**user ID** specifies the user ID of the User record. The user ID can contain 1–64 characters of any kind. |
| 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, you must enclose the NAME in single quotation marks. This parameter is not required by the UPDATE USER command. |

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 table describes the authorization record parameters for the INSERT USER and UPDATE USER commands. You can authorize each user to add, alter, read, or remove a record. Specify the authorization by indicating the action (ADD, ALTER, READ, REMOVE) followed by the record type. If you do not specify an action for a Type or User record, the action defaults to No.

| Parameter | Description |
|---|---|
| ADD TYPE = Y \| N | Specifies whether the user is allowed to insert new records into the Type Defaults file. |

| Parameter | Description |
|---|---|
| ALTER TYPE = Y \| N | Specifies whether the user is allowed to update records in the Type Defaults file. |
| READ TYPE = Y \| N | Specifies whether the user is allowed to read records from the Type Defaults file. |
| REMOVE TYPE= Y \| N | Specifies whether the user is allowed to delete records from the Type Defaults file. |
| ADD USER = Y \| N | Specifies whether the user is allowed to insert new records into the Authorization file. |
| ALTER USER = Y \| N | Specifies whether the user is allowed to update records in the Authorization file. |
| READ USER = Y \| N | Specifies whether the user is allowed to read records from the Authorization file. |
| REMOVE USER = Y \| N | Specifies whether the user is allowed to delete records from the Authorization file. |

### Functional Authorization Parameters

The following table describes the functional authorization parameters for the INSERT USER and UPDATE USER commands:

| Parameter | Description |
|---|---|
| APKEY = Y \| N | Specifies whether the user is allowed to update the license management key (also known as the asset protection key file). |
| CASE = Y \| N | Specifies whether accounting data, user ID, password, and data set name parameters are case sensitive. This choice overrides the case designation selected at session signon, and is in effect only for this command. The default is the designation made at session signon. |
| CDEL = Y \| N | Specifies whether the Confirm Delete/Suspend/Flush Command prompt displays for a particular user. |
| CDELOFF = Y \| N | Specifies whether the user can turn off the Confirm Delete/Flush/Suspend Command prompt for the current session. If you do not change the default of No to Yes, the user will always see the Confirm Delete/Flush/Suspend Command prompt and will not be given this option. |
| CHange = Y \| N | Specifies whether the user is allowed to use the CHANGE PROCESS command. |
| COPY = Y \| N | Specifies whether the user is allowed to use the COPY statement. |
| DELPR = Y \| N | Specifies whether the user is allowed to use the DELETE PROCESS command. |

| Parameter | Description |
|---|---|
| EVENTCMD = Y \| N | Specifies whether the user is allowed to use the Event Services Support commands. |
| FLUSH = Y \| N | Specifies whether the user is allowed to use the FLUSH PROCESS and SUSPEND PROCESS commands. |
| GEN.CHG.PROCESS = Y \| N | Specifies 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 he or she submitted (valid only in the IUI). |
| GEN.DEL.PROCESS = Y \| N | Specifies 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 he or she submitted (valid only in the IUI). |
| GEN.FLS.PROCESS = Y \| N | Specifies 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 he or she submitted (valid only in the IUI). |
| GEN.SEL.PROCESS = Y \| N | Specifies 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 he or she submitted (valid only in the IUI). |
| GEN.SEL.STATISTICS = Y \| N | Specifies 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 he or she submitted (valid only in the IUI). |
| GVIEW = Y \| N | Specifies whether the user can view only Processes submitted with a matching USERID or can view all Processes regardless of who submitted them. |
| MAXSA = max signon attempts | Specifies the maximum number of signon attempts the user is allowed per hour. The range is 0–99. The default is 60. Zero (0) indicates no maximum number. (See the RESETSA parameter to see 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. |
| MODIFY = Y \| N | Specifies whether the user is allowed to request traces and modify initialization parameters. |
| NSUBMIT = Y \| N | Specifies whether the user is allowed to use the SUBMIT statement to submit a Process. |
| OVCRC = Y \| N | Specifies whether the user is allowed to use the CRC statement to override the initial CRC settings. |
| PASSword = initial password | Defines the initial password for the user ID. The password is a 1–64 character alphanumeric string. |
| PHone = ’phone number’ | Specifies the phone number of the user. Enclose the phone number in single quotation marks. The quotation marks allow for a space after the area code. |

| Parameter | Description |
|---|---|
| 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. For more information, see *Generating RACF PassTickets* on page 61. |
| RESETSA | Specifies that the signon attempt count is reset to 0. (See the MAXSA parameter to see how to set the signon attempt count.) This parameter enables 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. |
| RUNTASK = Y \| N | Specifies whether the user is allowed to use the RUN TASK statement. |
| SECUREWR = Y \| N | Specifies whether the user can update the Secure+ Option Parameters file. |
| SECURITY = (security ID, security pswd) | Specifies the security ID and security password to identify the file authorization of the user. Security support includes CA-ACF2, CA-TOP SECRET, and RACF.<br>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.<br>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. |
| SELPR = Y \| N | Specifies whether the user is allowed to use the SELECT PROCESS command. |
| SELSTAT = Y \| N | Specifies whether the user is allowed to use the SELECT STATISTICS command. |
| STAT COMMAND = Y \| N | Specifies whether the user is allowed to use the STATISTICS COMMAND command. |
| STOPCD = Y \| N | Specifies whether the user is allowed to use the STOP CD command. |
| SUBMIT = Y \| N | Specifies whether the user is allowed to use the SUBMIT statement to define and submit within a Process. |

| Parameter | Description |
|---|---|
| SUBMITTER.CMDS = (Y \| N  Y \| N  Y \| N  Y \| N  Y \| N) | Specifies whether the user is allowed to issue certain commands concerning the Processes that he or she submitted (valid only in the batch interface). These commands are:<br>◆ SELECT PROCESS<br>◆ DELETE PROCESS<br>◆ FLUSH PROCESS<br>◆ CHANGE PROCESS<br>◆ SELECT STATISTICS<br>For more information, see the IUI definitions for these commands in *Functional Authorization Parameters* on page 100. (The IUI definitions begin with GEN and the command name is abbreviated.) |
| UPDNET = Y \| N | Specifies whether the user is allowed to use the UPDATE NETMAP command. |
| VIEW PROCESS = Y \| N | Specifies whether the user is allowed to use the VIEW PROCESS command. |

### Inserting and Updating Users through the Batch Interface

To use the INSERT USER or UPDATE USER commands from the batch interface:

1. Place your command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify your results.

The following example shows a User record for user ID Smith being added to the Authorization file.

```
SIGNONUSERID=(user ID, password)
INSERT USERUSERID=(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 Y)
SIGNOFF
```

In the example definition, the user 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 submitted Processes

In the example, Smith cannot perform the Select Statistics command on any Processes, regardless of who submitted them. Smith's initial password is XYZZY, and Smith's phone number is (817) 555-5555.

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

```
UPDATE USERUSERID=(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 these updates, Smith can 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

### Inserting and Updating Users through the IUI

Use the Insert/Update/Select/Delete User Record screen (following) to insert, update, select, or delete a record. Select option **IU** from the Administrative Options Menu to access this screen.

```
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==> node.name_____             PHONE     ==> _______________
   PASSWORD==>
   SEC ID  ==> ____________________________________________________________
   SEC PASS==>
   MAX SIGNON ATTEMPTS===> 60        PASSTICKET DATA ==> ( _________ ,          )

 DO YOU WANT VALUES FOR THIS COMMAND TO BE CASE SENSITIVE? ==> NO_

 DEFINE USER FUNCTIONS: (RESPOND WITH 'Y'-YES OR 'N'-NO)

   FLUSH PROCESS   => _   SELECT NETMAP      => _   UPDATE NETMAP    => _
   INSERT USER     => _   SELECT PROCESS     => _   MODALS FUNCTION  => _
   DELETE USER     => _   SUBMIT PROCESS     => _   RUNTASK FUNCTION => _
   SELECT USER     => _   SUBMIT WITHIN PROC => _   INSERT TYPE      => _
   UPDATE USER     => _   RUNJOB FUNCTION    => _   DELETE TYPE      => _
   COPY FUNCTION   => _   CONTROL TRACING    => _   SELECT TYPE      => _
   CHANGE PROCESS  => _   STOP Connect:Direct=> _   UPDATE TYPE      => _
   DELETE PROCESS  => _   SELECT STATISTICS  => _   GEN.FLS.PROCESS  => _
   STAT COMMAND    => _   GEN.DEL.PROCESS    => _   RESET SIGNON     => N
   GEN.SEL.PROCESS => _   GEN.SEL.STATISTICS => _   VIEW PROCESS     => _
   GEN.CHG.PROCESS => _   EVENT COMMAND      => _   CRC OVERRIDES    => _
   GEN.VIEW.PROC   => _   CONFIRM DELETE     => _   CONFIRM DEL OFF  => _
   UPDATE APKEY    => _   SECURE+ ADMIN      => _
```

The DEFINE USER FUNCTIONS portion of the screen is scrollable. **More + or -** indicates additional data. Press **PF8** to scroll forward. Press **PF7** to scroll back. See the description of the INSERT and UPDATE USER command parameters on page 97 for the valid values of the fields, or press the **PF1** key for Help.

## Deleting Users from the Authorization File

The DELETE USER command removes a User record from the Connect:Direct Authorization file. The command has the following format and parameters. The required parameters and keywords appear in bold print.

| Label | Command | Parameters |
|---|---|---|
| (optional) | **DELete USER** | **WHERE (** |
| | | **USERID** = (nodename, user ID) \| (list) |
| | | **)** |
| | | CASE = YES \| NO |

| Parameter | Description |
|---|---|
| WHERE<br><br>(USERID = (nodename, user ID) \| (list) ) | This parameter specifies which user records to delete.<br>USERID = (nodename, user ID) \| (list)<br>This parameter specifies the record to delete from the Connect:Direct Authorization file.<br>nodename specifies the node ID of the User record that is searched. The nodename is a 1–16 character alphanumeric string.<br>The user ID parameter specifies the user ID of the User record. The complete user ID consists of the nodename and the user ID enclosed in parentheses and separated by a comma.<br>The list parameter specifies a list of user IDs. |

CASE is the only optional parameter.

| Parameter | Description |
|---|---|
| CASE = Yes \| No | This parameter specifies whether nodename and user ID parameters are case sensitive. This choice overrides the case designation selected at session signon and is in effect only for this command. The default is the designation made at session signon. |

### Deleting Users through the Batch Interface

To use the DELETE USER command from the batch interface:

1. Place your command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify your results.

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

You can delete a user in the Connect:Direct IUI from:

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

The Delete A User Record screen enables you to simultaneously delete up to four User records. To use the Delete A User Record screen:

1. Select option **DU** from the Administrative Options Menu to display the Delete A User Record screen.
2. Type the user ID and user node of the records to delete.
3. Press **ENTER**.
4. Verify your results.

To delete a User record using the Insert/Update/Select/Delete User Record screen (on page 105), select the **IU** option from the Administrative Options Menu. See the description of the DELETE USER command parameters on page 105 for the valid values of the fields on this screen, or press the **PF1** key for Help.

## Selecting User Information from the Authorization File

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

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

| Label | Command | Parameters |
|---|---|---|
| (optional) | **SELect USER** | **WHERE (** |
| | | **USERID** = (nodename, user ID) \| |
| | | (generic \| (list)) |
| | | EXCLUDE = (AUTH) |
| | | **)** |
| | | PRint \| TABle |
| | | CASE = YES \| NO |

WHERE is the only required parameter for the SELECT USER command and USERID is the only required subparameter.

| Parameter | Description |
|---|---|
| WHERE<br><br>(USERID = (nodename, user ID) \| (generic \| list)) | This parameter specifies which User records you want to examine.<br><br>USERID = (nodename, user ID) \| (generic \| list)<br>This parameter specifies the record to search for in the Connect:Direct Authorization file. This subparameter of the WHERE parameter is required. The complete user ID consists of the nodename and the user ID enclosed in parentheses and separated by a comma.<br>**nodename** specifies the node ID of the User record that is searched. Type a 1–16 character alphanumeric string. If the user node is not specified, nodename defaults to the Connect:Direct system that receives the command.<br>**user ID** specifies the user ID of the User record.<br>**generic** specifies generic selection of user IDs. To specify user nodes and user IDs generically, type a 1–7 character alphanumeric string with the first character alphabetic, plus an asterisk (*). For instance, if you specify a user ID of B*, examine records for BLACK, BRADFORD, and BROWN.<br>**list** specifies a list of user IDs. |
| EXCLUDE = (AUTH) | EXCLUDE = (AUTH)<br>This parameter specifies that the function-by-function authorization description is not included in the output. This subparameter of the WHERE parameter is not required. |

The following table describes the optional parameters for the SELECT USER command:

| Parameter | Description |
|---|---|
| PRint \| TABle | This parameter specifies the output destination.<br>**PRINT** specifies that the output of the SELECT USER 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.<br>**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 | This parameter specifies whether parameters associated with nodename and user ID are 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. |

### Selecting a User through the Batch Interface

To use the SELECT USER command from the batch interface:

1. Place your command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify your results.

The following command searches for user BILL at the local (default) node.

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

### Selecting a User through the IUI Interface

The Select a User Record screen enables you to simultaneously display user information and authorized functions for up to four users. You can also use the Insert/Update/Select/Delete User Record screen (option **IU**) to select users.

1. From the Administrative Options Menu, select option **SU** to display the Select a User Record screen and press **Enter**.
2. Type the user ID and user node for the records you want to view.
3. Indicate the output destination, the case sensitivity, and whether you want to exclude the function-by-function authorization description.

---

**Note:** See the description of the SELECT USER command parameters on page 107 for a description of the valid values for the fields or press **PF1** for Help.

---

4. Press **Enter**.

# Chapter 5

# Maintaining the Type File

This chapter includes the following topics:

- Description of the Type File
- Adding or Updating File Types
- Deleting Type Records
- Viewing Type Records

## Description of the Type File

The Type file contains file attribute records. Each file attribute record is associated with a Type key. The Type key is used by the TYPE parameter in the COPY statement to create new files or access existing files.

The following example illustrates a Copy Process using the TYPE parameter.

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

The Type file serves two purposes:

- Saves retyping parameters such as DCB, DISP, and SPACE within Processes for files with common attributes.
- Facilitates the use of previously-defined attribute specifications of non-z/OS systems. This usage is especially useful for remote users who are not familiar with z/OS data set organizations and allocation parameters.

The Type key that is referenced in the TYPE= parameter must be 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 in the COPY statement override similar parameters in the Type record. Use this functionality when you want to override a specific Type record subparameter.

## Type Keys

Connect:Direct for z/OS provides you with four predefined Type keys to communicate with other Connect:Direct nodes:

- TEXT
- DF
- DF2
- BINARY

The four Type keys contain file allocation information as defined in the following figure.

```
TYPE KEY => TEXT
    DISP=(RPL,CATLG,DELETE)
    DCB=(DSORG=PS,LRECL=255,BLKSIZE=2554,RECFM=VB)
    SPACE=(TRK,(10,10))
    UNIT=SYSDA

TYPE KEY => DF (Data File)
    DISP=(RPL,CATLG,DELETE)
    DCB=(DSORG=PS,LRECL=255,BLKSIZE=2554,RECFM=VB)
    SPACE=(TRK,(10,10))
    UNIT=SYSDA

TYPE KEY => DF2 (Data File 2)
    DISP=(RPL,CATLG,DELETE)
    DCB=(DSORG=PS,LRECL=80,BLKSIZE=3120,RECFM=FB)
    SPACE=(TRK,(10,10))
    UNIT=SYSDA

TYPE KEY => BINARY
    DISP=(RPL,CATLG,DELETE)
    DCB=(DSORG=PS,BLKSIZE=6144,RECFM=U)
    SPACE=(TRK,(10,10))
```

## Maintaining the Type File

You maintain the Type file by using the INSERT TYPE, UPDATE TYPE, DELETE TYPE, and SELECT TYPE commands. Type these commands through the IUI, Batch interface, or Operator interface.

# Adding or Updating File Types

Use the INSERT TYPE command to insert a new Type record into the Type file. Use the UPDATE TYPE command to update an existing Type record. These commands use the following format and parameters. The required parameters appear in bold print. Default values for parameters are underlined.

| Label | Command | Parameters |
|---|---|---|
| (optional) | **INSert TYPE** \| **UPDate TYPE** | **TYPEKEY**=typekey |
| | | DCB=(BLKSIZE = no. bytes |
| | | ,DSORG = (DA \| PS \| PO \| VSAM) |
| | | ,LRECL = no. bytes |
| | | ,RECFM = record format) |
| | | DISP=((NEW \| OLD \| MOD \| RPL \| SHR)<br>(,KEEP , CATLG , DELETE)<br>(,KEEP , CATLG , DELETE)) |
| | | DSNTYPE=PDS\|LIBRARY\|BASIC\|LARGE\|EXTPREF\|EXTREQ |
| | | AVGREC=(U\|K\|M) |
| | | DATACLAS=data_class_name |
| | | KEYLEN=bytes |
| | | KEYOFF=offset_to_key |
| | | LIKE=model_data_set_name |
| | | LRECL=bytes |
| | | MGMTCLAS=management_class_name |
| | | RECORG=(KS\|ES\|RR\|LS) |
| | | SECMODEL=(profile_name [,GENERIC]) |
| | | STORCLAS=storage_class_name |
| | | SPACE=(CYL \| TRK \| blk,<br>(prim, sec, (dir))<br>(,RLSE\|,(CONTIG\|,)(ROUND\|))<br>(ave_rec_len,(primary_rcds,<br>secondary_rcds)) |
| | | UNIT=unit type |
| | | VOL=SER = volume serial number |
| | | IOEXIT=exitname\|<br>(exitname[,parameter,parameter...]) |

## Required Parameter

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

| Parameter | Description |
|---|---|
| **TYPEKEY=typekey** | This parameter specifies the name associated with the entry being added or updated. The Type key is a 1–8 character alphanumeric string. The first character must be alphabetic. |

## Optional Parameters

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

| Parameter | Description |
|---|---|
| DCB=<br>([BLKSIZE=no.bytes,<br>DSORG=[DA\|PS\|PO\|<br>VSAM], LRECL=no.bytes,<br>RECFM=record format]) | This parameter specifies DCB information associated with the data set name on the COPY statement.<br>**BLKSIZE** specifies the length of the block in bytes.<br>**DSORG** specifies the file organization. File organizations supported are DA, PS, PO, and VSAM.<br>**LRECL** specifies the length of the records in bytes.<br>**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.<br>Any file attributes specified on the COPY statement take precedence over those in 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 in the COPY TO or the Type file, the attributes are taken from the FROM side (source). |

| Parameter | Description |
|---|---|
| DISP=<br>([NEW\|OLD\|MOD\|RPL\|SHR]<br>[,KEEP\|,CATLG\|,DELETE]<br>[,KEEP\|,CATLG\|,DELETE]) | This parameter specifies the default destination file status on the receiving node.<br>The first DISP subparameter specifies the status of the file. Only the OLD and RPL dispositions apply to VSAM files.<br><br>**NEW** (default) specifies that the Process step creates the destination file.<br>**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 can be a VSAM file, a SAM file, or a PDS (Connect:Direct for z/OS).<br>**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 is difficult to detect.<br>**RPL** specifies that the destination file replaces any existing file or allocates a new file. You can specify DISP=RPL for SAM or VSAM files. If the file is VSAM, you must define it with the REUSE attribute. You cannot specify RPL if VOL=SER is specified.<br>**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.<br><br>The second subparameter specifies the normal termination disposition. It does not apply to VSAM files.<br>**KEEP** specifies that the system keeps the file after the Process step completes. If DISP=(NEW,KEEP), you must also specify a volume serial number.<br>**CATLG** (default) specifies that the system keeps the file after the Process step completes and places an entry in the catalog.<br>**DELETE** specifies that the system deletes the file after the Process step completes.<br><br>The third subparameter specifies abnormal termination disposition. It does not apply to VSAM files. This subparameter has no default.<br>**KEEP** specifies that the system keeps the file after the Process terminates abnormally.<br>**CATLG** (default) specifies that the system keeps the file after the Process step terminates abnormally and places an entry in the catalog.<br>**DELETE** specifies that the system deletes the file after the Process if Connect:Direct terminates abnormally.<br>The third subparameter specifies abnormal termination disposition. It does not apply to VSAM files. This subparameter has no default. |

| Parameter | Description |
|---|---|
| DSNTYPE = LIBRARY \| PDS \| BASIC \| LARGE \| EXTPREF \| EXTREQ | Defines a specific data set organization for an SMS controlled data set of a PARTITIONed type.<br>**LIBRARY** specifies a partitioned data set extended (PDSE).<br>**PDS** specifies a partitioned data set.<br>**LARGE** specifies specifies a sequential data set which can contain more than 65535 tracks per volume.<br>**BASIC** specifies a sequential data set which can have no more than 65535 tracks per volume.<br>**EXTPREF** specifies that the extended attribute is preferred. If an extended format data set cannot be allocated, a data set is created without the attribute.<br>**EXTREQ** specifies that the extended attribute is required. If an extended format data set cannot be allocated, a data set is not created. |
| AVGREC=(U \| K \| M) | Requests that Connect:Direct allocate the data set in records. The primary and secondary space quantities represent the number of records requested in units, thousands, or millions of records. This parameter is mutually exclusive with the TRK, CYL, and ABSTR subparameters of the SPACE keyword.<br>**U** specifies a record request where primary and secondary space quantities are the number of records requested. It is a multiple of **1**.<br>**K** specifies a record request where primary and secondary space quantities are the number of records requested in thousands of records. It is a multiple of 1024.<br>**M** specifies a record request where primary and secondary space quantities are the number of records requested in millions of records. It is a multiple of 1,048,576. |
| DATACLAS=data_class_ name | Requests the data class for a new data set. The class selected must be previously defined by the SMS administrator. You can use this keyword with VSAM data sets, sequential data sets, or partitioned data sets.<br>**data_class_name** specifies the 1–8 character name of the data class to which this data set belongs. The name of the data class is assigned by the SMS administrator. |
| KEYLEN=bytes | This parameter specifies the length of the keys in the file. This keyword is not restricted as a subparameter of the DCB keyword to support use with VSAM KS data sets.<br>**bytes** specifies the length in bytes of the keys used in the file. You must specify this value with a decimal integer from 0–255 for non-VSAM data sets or 1–255 for VSAM data sets. |
| KEYOFF=offset_to_key | This parameter specifies the offset within the record to the first byte of the key in a new VSAM KS data set. The relative first byte of the record is byte 0. The range is 0–32760.<br>**offset_to_key** specifies the position of the first byte of the key in the record. |

| Parameter | Description |
|---|---|
| LIKE=model_data_set_name | Requests that allocation attributes for a new data set are copied from an existing cataloged data set. Any or all of the following attributes are copied to the new data set: RECORG or RECFM, LRECL, KEYLEN, KEYOFF, DSNTYPE, AVGREC, and SPACE. Any attributes specified for the data set override the values from the model data set. Neither EXPDT nor RETPD is copied from the model data set.<br>**model_data_set_name** specifies the name of the data set from which the allocation attributes are copied. |
| LRECL=bytes | This parameter specifies the length in bytes of the records in the new data set. This parameter is allowed outside of the DCB keyword to allow use with SMS VSAM data sets. Do not specify LRECL with RECORG=LS type data sets.<br>**bytes** specifies the length of the records in the data set. For a non-VSAM data set, this length is 1–32760 bytes. For VSAM data sets, this length is 1–32761 bytes. The LRECL must be longer than the KEYLEN value for VSAM KS data sets. |
| MGMTCLAS=<br>management_class_name | Determines to which of the previously defined management classes this new data set belongs. The attributes in this class determine such things as when a data set is migrated and when the data set is backed up.<br>**management_class_name** specifies the 1–8 character name of the management class to which this data set belongs. The name of the management class is assigned by the SMS administrator. |
| RECORG=(KS\|ES\|RR\|LS) | Defines the organization of records in a new VSAM data set. If RECORG is not specified, then SMS assumes that the data set is either a physical sequential (PS) data set or a partitioned (PO) data set.<br>**KS** specifies a VSAM key-sequenced data set<br>**ES** specifies a VSAM entry-sequenced data set<br>**RR** specifies a VSAM relative record data set<br>**LS** specifies a VSAM linear space data set |
| SECMODEL=<br>(profile_name,GENERIC) | Copies an existing RACF profile as the discrete profile for this new data set. The following information is copied along with the profile: OWNER, ID, UACC, AUDIT/GLOBALAUDIT, ERASE, LEVEL, DATA, WARNING, and SECLEVEL.<br>**profile_name** specifies the name of the model RACF profile, discrete data set profile, or generic data set profile that is copied to the discrete data set profile created for the new data set.<br>**GENERIC** identifies that the profile_name refers to a generic data set profile. |

| Parameter | Description |
|---|---|
| STORCLAS=<br>storage_class_name | This parameter specifies the storage class to which the data set is assigned. The SMS administrator must define the storage class name to the SMS system by the SMS administrator. The storage class defines a storage service level for the data set and replaces the UNIT and VOLUME keywords for non-SMS data sets. You cannot use JCL keywords to override any of the attributes in the storage class. You can use an Automatic Class Selection (ACS) routine to override the specified class.<br>**storage_class_name** specifies the 1–8 character name of the storage class to which this data set is assigned. |
| SPACE<br>= (CYL \| TRK \| blk, (prim, [sec], [dir]) [,RLSE \|, ] [,CONTIG \|, ] [,ROUND]) \| (ave_rec_len,[primary_rcds, secondary_rcds]) | This parameter specifies the amount of storage allocated for new files on the destination node. If SPACE is specified, the DISP of the destination file must be NEW. If SPACE is not specified and the DISP is either NEW or CATLG, space allocation defaults to the value obtained from the source file. The default is blk (blocks) with the ROUND option, which provides device-independent space allocation.<br>If the AVGREC keyword is specified, the allocation of the data set is done on a record size basis instead of TRK, CYL, or BLK. This restriction is also true when the AVGREC keyword is present in the COPY statement.<br>**CYL** specifies that space is allocated in cylinders.<br>**TRK** specifies that space is allocated in tracks.<br>**blk** specifies that space is allocated by the average block length of the data. The system computes the number of tracks to allocate. If the subparameter ROUND is also specified, the system allocates the space in cylinders. ROUND is preferred because allocation is performed on cylinders in a device-independent manner.<br>**prim** specifies the primary allocation of storage.<br>**sec** specifies the secondary allocation of storage.<br>**dir** specifies the storage allocated for the PDS directory.<br>**RLSE** releases the unused storage allocated to the output file.<br>**CONTIG** specifies that the storage for the primary allocation must be contiguous.<br>**ROUND** specifies that the storage allocated by average block length be rounded to an integral number of cylinders.<br>**ave_rec_len** specifies the average record length, in bytes, of the data. Connect:Direct computes the BLKSIZE and the number of tracks to allocate. The record length must be a decimal value from 1–65535.<br>**primary_rcds** specifies the number of records that the data set contains. Connect:Direct uses this number and the value of the AVGREC keyword to compute the primary space allocation.<br>**secondary_rcds** specifies the number of additional records to allocate space for when the primary space is exhausted. Connect:Direct uses this value and the AVGREC keyword to compute the number of tracks to allocate. |
| UNIT = unit type | This parameter indicates 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. |

| Parameter | Description |
|---|---|
| VOL = SER = volume serial number | This parameter specifies the volume serial number containing the file. If VOL=SER is not specified with the FROM parameter, you must catalogue the file. |
| IOEXIT = exitname \| (exitname [,parameter, parameter,...]) | This parameter indicates that a user-written program is given control to perform I/O requests for the associated data.<br>**exitname** specifies the name of the user-written program given control for I/O related requests for the associated data. The character length for IOEXIT is a variable of 1–510 characters.<br>**parameter \| (parameter,parameter,...)** specifies a parameter, or list of parameters, passed to the specified exit. A parameter consists of a data type followed by the value in single quotes, for example C'ABC'. For a full description of valid parameter formats, see the RUN TASK statement parameters described on the Connect:Direct Processes Web site at<br>http://www.sterlingcommerce.com/documentation/processes/processhome.html. |

## Inserting and Updating Type Files through the Batch Interface

To use the INSERT TYPE or UPDATE TYPE command from the batch interface, perform the following steps:

1. Place your command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify your results.

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

```
INSERT TYPE  TYPEKEY=NEWALLOC -
             DCB=(DSORG=PS) -
             DISP=(NEW,CATLG) -
             UNIT=3380
```

Connect:Direct users can then use the NEWALLOC Type key in a COPY command to allocate a new physical sequential file on a 3380 unit device that is cataloged on normal termination.

The following example updates a record in the Type file. When referring to the NEWALLOC Type key, the destination file is an existing PS file allocated on a 3390 disk pack:

```
UPDATE TYPE   TYPEKEY=NEWALLOC -
              UNIT=3390
```

## Inserting and Updating Type Files through the IUI

Use the Insert/Update Type Record screen to insert or update a Type record. To access this screen, select option **IT** from the Administrative Options Menu. The Insert/Update Type Record screen is displayed.

### Insert/Update Type Record Screen

The Insert/Update Type Record screen prompts you for the action to perform (Insert or Update) and the Type key. Type the requested information and press **ENTER**. Connect:Direct displays an error message if you attempt to insert an existing record or update a nonexistent record.

### Type Record Selection List Screen

The next screen displayed is the Type Record Selection List screen. Select either the General Dataset Attributes, IOEXIT Parameters, or the SMS/VSAM Attributes option by typing an **S** next to the appropriate entry and pressing **Enter**.

```
node.name              Type Record Selection List          hh:mm
CMD ==>

Operation   ==> UPDATE
Type Key    ==> X

       _    General Dataset Attributes
       _    IOEXIT Parameters
       _    SMS/VSAM Attributes

       Select the entries above to view and/or update the
       respective parameters.

       Enter the END command to perform the updates.
         -or-
       Enter the CANcel command to abandon your changes.
```

Type the END command from the Type Record Selection List or select another option to perform the update.

Type the CANcel command from the Type Record Selection List screen to terminate the update.

### Type Record General Dataset Attributes Screen

From the Type Record General Dataset Attributes screen, you can define or update the file attributes for the Type record. See *Optional Parameters* on page 114 for a description of the fields on this screen you can update for the Type record you are inserting or updating.

```
node.name          Type Record General Dataset Attributes          hh:mm
CMD ==>

(DISP=)     Initial file status      ==> ___     (SHR,NEW,OLD,MOD,RPL)
            Normal step termination  ==> _______ (KEEP,CATLG,DELETE)
            Abend step termination   ==> _______ (KEEP,CATLG,DELETE)

(DCB=)      DSORG ==> ____  LRECL ==> _____  BLKSIZE ==> _____  RECFM ==> ____

(SPACE=)    Allocation type    ==> __________ (CYL,TRK,Average blksize)
            Primary extent     ==> __________ (Required with CYL,TRK,BLK)
            Secondary extent   ==> _________  (Optional)
            Directory storage  ==> _________  (Optional)
            RLSE,CONTIG,ROUND  ==> ( _ , _ , _ )  ('Y'-Yes, 'N'-No)

(UNIT=)     Unitname          ==> ________

(VOL=SER=)  Volume serial      ==> ______
```

Type the END command to return to the Type Record Selection List.

### Type Record IOEXIT Parameters Screen

The Type Record IOEXIT Parameters screen (a TSO edit screen) enables you to define or update the IOEXIT parameters associated with the Type record. Refer to the IOEXIT parameter description on page 119 for valid values.

```
 node.name                 Type Record IOEXIT Parameters                  hh:mm
 Cmd ==>                                                  Scroll ===> PAGE

        IOEXIT Specifications (exitname,{parameter,parameter,...})
--------------------------------------------------------------------------------
 ****** ***************************** Top of Data ******************************
        RIOEX01,(C'MAXLEN',X'0120')









****** **************************** Bottom of Data ****************************
```

After defining or updating the IOEXIT specifications, type the END command (or use the equivalent PF key) to return to the Type Record Selection List.

### Type Record SMS/VSAM Parameters Screen

The Type Record SMS/VSAM Parameters screen enables you to define or update parameters related to SMS controlled data sets and VSAM files. See *Optional Parameters* on page 114 for a description of the fields on this screen you can update for the Type record you are inserting or updating.

```
 node.name              Type Record SMS/VSAM Attributes              hh:mm
CMD ==>


Operation   ==> UPDATE
Type Key    ==> NEWTYPE

(SMS=)     Data Class       ==> ________
           Management Class ==> ________
           Storage Class    ==> ________
           DSNTYPE          ==> ________  (PDS, LIBRARY, LARGE, BASIC, EXTREQ,
                                           EXTPREF)
(MODEL=)   Like Data Set Name ==> ____________________________________________
           Security Profile   ==> ____________________________________________
           Generic Profile    ==> ___   (YES or NO)

(SPACE=)   Average Record Value ==> _  (U, K or M)

(VSAM=)    Organization ==> __     (ES - ESDS, KS - KSDS, RR - RRDS, LS - LDS)
           Key Offset   ==> _____  (0 - 32760)
           Key Length   ==> ___    (1 - 255)
```

After defining or updating the SMS/VSAM attributes, type the END command to return to the Type Record Selection List.

## Deleting Type Records

Use the DELETE TYPE command to delete a Type record from the Type file. Use the following format and parameters. The required parameters and keywords are 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. No optional parameters exist for this command.

| Parameter | Description |
|---|---|
| WHERE<br>(TYPEKEY = typekey \| (list)) | This parameter specifies which records in the Type file to delete. You can specify one Type key or a list of Type keys.<br>**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.<br>**list** specifies multiple Type keys. A list of Type keys is specified by enclosing them in parentheses. |

## Deleting Types Files through the Batch Interface

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

1. Place your command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify your results.

For example, the following commands delete the records under the Type keys 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))
```

## Deleting Type Files through the IUI

To issue the DELETE TYPE command from 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.
3. Press **ENTER**.

   Connect:Direct displays a list of the deleted records. If the delete is unsuccessful, Connect:Direct displays a list of the records not deleted.
4. Press **PF3/END** to return to the Delete Data Set Type Defaults screen.

   This screen displays a message indicating if the delete request is successful.
5. Use the Delete Data Set Type Defaults message display to verify a successful delete request.

# Viewing Type Records

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 parameters. The required parameters and keywords are 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 parameter for the SELECT TYPE command.

| Parameter | Description |
|---|---|
| WHERE (TYPEKEY = typekey \| generic \| (list)) | This parameter specifies which records in the Type file to select.<br>TYPEKEY = typekey \| generic \| (list)<br>specifies the key or list of keys of the records to select.<br>**typekey** specifies the name associated with the record selected. You created the typekey name when originally adding the entry to the Type file. The typekey is a 1–8 character alphanumeric string, with the first character alphabetic.<br>**generic** specifies generic selection of type keys. 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, and SENDWK, a specification of SEND* provides information about those keys.<br>**list** specifies multiple type keys. A list of type keys is specified by enclosing them in parentheses. |

## Optional Parameters

The optional parameter is described below.

| Parameter | Description |
|---|---|
| PRInt \| TABle | Parameters specify the method of display for the output of the select.<br>**PRint** specifies that the output to a printer in tabular format. Output is routed to the destination specified in the PRINT keyword of the Connect:Direct SIGNON command.<br>**TABle** specifies that the output is stored in a temporary file in tabular format and is displayed upon successful completion of the command. This parameter is the default. |

## Selecting Type Records through the Batch Interface

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

1. Place your command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.

   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 => 1/19/1998

       DISP=(RPL,CATLG,DELETE)
       DCB=(DSORG=PS,LRECL=255,BLKSIZE=2554,RECFM=VB)
       SPACE=(TRK,(10,10))
       UNIT=SYSDA
```

3. Use the output to verify your results.

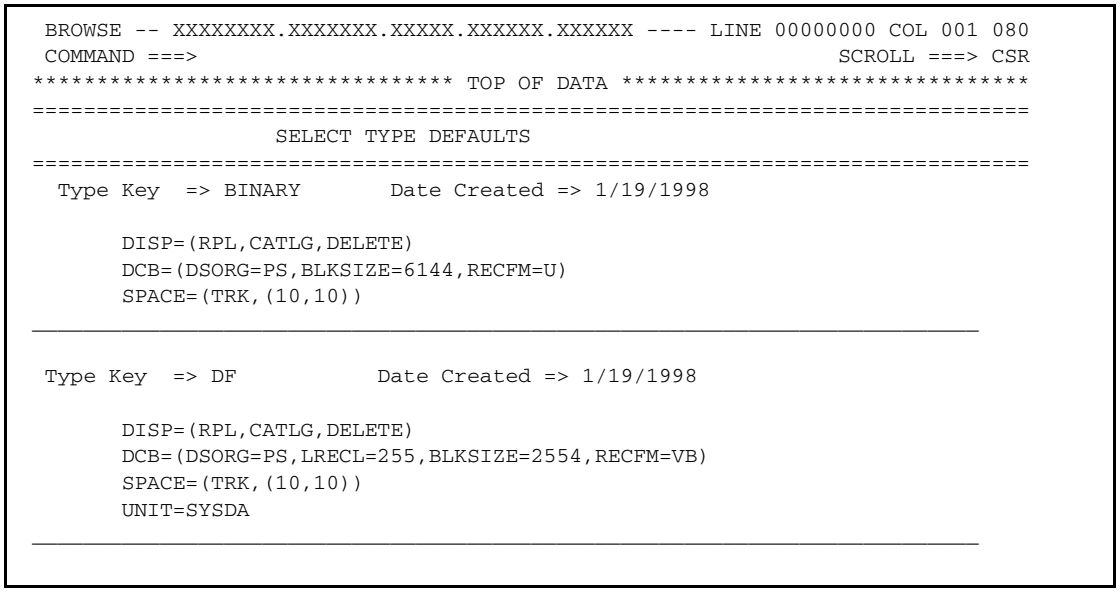
## Selecting Type Records through the IUI

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. Type the TYPEKEY for each member you want to select. Refer to the description of parameters for the SELECT TYPE command on page 124, or press **PF1** to view the Help.
3. Indicate the output destination in the Output Destination field.

4. Provide the requested information and press **ENTER**.

   An output destination of **DIS** produces a display similar to the following figure.



```
 BROWSE -- XXXXXXXX.XXXXXXX.XXXXX.XXXXXX.XXXXXX ---- LINE 00000000 COL 001 080
 COMMAND ===>                                                 SCROLL ===> CSR
******************************** TOP OF DATA ********************************
================================================================================
                    SELECT TYPE DEFAULTS
================================================================================
  Type Key  => BINARY       Date Created => 1/19/1998

       DISP=(RPL,CATLG,DELETE)
       DCB=(DSORG=PS,BLKSIZE=6144,RECFM=U)
       SPACE=(TRK,(10,10))
_______________________________________________________________________________

 Type Key  => DF           Date Created => 1/19/1998

       DISP=(RPL,CATLG,DELETE)
       DCB=(DSORG=PS,LRECL=255,BLKSIZE=2554,RECFM=VB)
       SPACE=(TRK,(10,10))
       UNIT=SYSDA
_______________________________________________________________________________
```

# Chapter 6

# Maintaining the Network Map

This chapter contains the following topics:

- Network Map Contents
- API Signons
- Examples of Local and Adjacent Node Records
- Updating the Network Map
- Viewing the Network Map

**Note:** You can also use the Connect:Direct Browser to perform some of the procedures in this chapter. To learn more about the Connect:Direct Browser, see the *Connect:Direct Browser User Interface User's Guide* on the Connect:Direct Browser CD-ROM or online in the Sterling Commerce Documentation Library.

## Network Map Contents

The network map (also known as the NETMAP) identifies the local Connect:Direct node and the nodes it communicates with. It consists of a local node entry and one or more adjacent node entries that identify the communication name and protocol of each Connect:Direct node.

The network map source is generated during installation. The network map load utility, DMCNTMPL, uses this source to create a Connect:Direct network map (a VSAM file).

See the *Connect:Direct for z/OS Installation Guide* for rules governing the network map for cross-domain VTAM definitions.

## Local Node Entry

The local node entry specifies the logical name of the local Connect:Direct and its associated communications name. The local node entry also contains the name of the Transmission Control Queue (TCQ) and the SUPERUSR ID password, if specified. The syntax is displayed in the following figure.

```
LOCAL.NODE=(                                                    -
            (node name,communications name,,superuser password) -
            TCQ=(tcxdsn, tcqdsn)                                -
            CONTACT.NAME="name"                                 -
            CONTACT.PHONE="phone information"                   -
            DESCRIPTION="description information"               -
            )
```

**Note:** Place any comments on separate lines in the network map outside of LOCAL.NODE or ADJACENT.NODE definitions as shown in the following example. Comments inside of a NODE definition cause the map build to be done incorrectly.

```
LOCAL.NODE=((CD.DALLAS,NDMAPP1,,XYZZY) -
       TCQ=(DSC.DALLAS.TCX DSC.DALLAS.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))
```

### Local Node Parameters

The network map local node entry contains the following positional and keyword parameters.

### Positional Parameters

| Parameter | Description |
|---|---|
| node name | The first positional parameter is the 1–16 alphanumeric character node name. It specifies the logical name of the local Connect:Direct DTF.<br><br>The z/OS operating system accepts the following characters for the adjacent node:<br>A-Z, 0-9, @, #, $, !, %, ^, &, _ , +, -, [, ], {, }<br>The z/OS operating system does not accept the following characters for the adjacent node:<br>(, ) =, \, “, ‘, <, >, \|, \|\| |

| Parameter | Description |
|---|---|
| communications name (for SNA only) | The second positional parameter is the 1–8 character communications name. It specifies the VTAM APPLID that Connect:Direct uses to communicate over the network.<br>If the node uses only TCP/IP communications (SNA=NO is specified in the initialization parameters), leave this parameter blank. Refer to *Initializing Connect:Direct without SNA Support* in the *Connect:Direct for z/OS Installation Guide* for more information. |
| null | The third positional parameter is not used. |
| superuser password | The fourth positional parameter is the 1–8 character SUPERUSR ID password. The initial value for this parameter is specified during installation.<br>The SUPERUSR ID is provided to bypass your usual security system at signon. This bypass can 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. |

### Keyword Parameter

| Parameter | Description |
|---|---|
| TCQ=<br>(tcxdsn,tcqdsn) | The TCQ is a VSAM relative record data set (RRDS) that holds all Processes submitted to Connect:Direct.<br>The TCQ parameter specifies the two files that comprise the Transmission Control Queue (TCQ). This parameter is required.<br>**Note:** Use the correct order (tcxdsn,tcqdsn) when you specify the two files.<br>**tcxdsn** identifies the data set name of the TCQ index (TCX).<br>**tcqdsn** identifies the data set name of the TCQ. |
| CONTACT.NAME= ”name”<br>CONTACT.PHONE ”phone information”<br>DESCRIPTION=<br>”description information” | These are free-form text parameters, which provide additional general information about an adjacent node entry. The CONTACT.NAME and CONTACT.PHONE parameters are limited to a maximum of 40 characters. The DESCRIPTION parameter is limited to a maximum of 255 characters. |

## Defining Local Node as Adjacent Node

You must also define the local node as an adjacent node to:

- Specify the VTAM application IDs that are used for IUI and batch sessions. The APPLIDS match those defined during installation preparation as described in the *Connect:Direct for z/OS Installation Guide*.
- 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 z/OS Installation Guide*. Observe the rules as described in *PNODE=SNODE Processing* on page 142.

- ✦ Supply the communication address for a TCP API, if the address is not specified during the user signon.

**Note:** If you are using TCP/IP only and SNA is set to NO, refer to *Initializing Connect:Direct without SNA Support* in the *Connect:Direct z/OS Installation Guide* for more information.

## Adjacent Node Entry

Adjacent node entries specify network nodes that the local Connect:Direct can communicate with. Each entry specifies a locally used Connect:Direct name, its associated network communications name, and session control parameters for these nodes. The syntax in the following figure is for a typical adjacent node entry.

```
ADJACENT.NODE=(                                                    -
     (nodename,                                                    -
      communications name | channel-range-start-addr,              -
      remote library name | IP address or Alias | addr-count,      -
      session type,                                                -
      security node type,                                          -
      data direction restriction)                                  -
     PARSESS=(max,default)                                         -
     SOURCEIP=IP address                                           -
     SESS.SNODE.MAX = 255                                          -
     LDNS=hostname                                                 -
     ENVIRONMENT=operating environment                             -
     LOGMODE=logmode entry name                                    -
     APPLIDS=(vtam applid1 [,vtam applid2,...] )                   -
     NETID=networkid | CTCA server name                            -
     PNODE.LUS=(luname1 [,luname2,...] )                           -
     SNODE.LUS=(luname1 [,luname2,...] )                           -
     USE.SERVER.NODE = NO | YES                                    -
     TCPAPI= (port number, IP address)                             -
     CRC= (OFF | ON | DEFAULT)                                    -
     PLEXCLASS= (* | plexclass, * | plexclass)                      -
     BUFFER.SIZE= 3072-262144|3K-256K                              -
     ALTernate.COMMinfo = (ALTernate.RESTART=YES | NO,                  -
              ALTernate.DIRection=BALANCE | TOP,                        -
             (ALTernate.ADDRess= | ALTernate.NODEDEF=, ALTernate.PORT=, -
              SOURCEIP=IP address,                                      -
              ALTernate.TYPE=SNA | TCP | UDT33 | LU62                   -
              ALTernate.LOGmode=logmode entry name,                     -
              ALTernate.USE.OUTbound=YES | NO)                          -
              )                                                         -
     CONTACT.NAME="name"                                           -
     CONTACT.PHONE="phone information"                             -
     DESCRIPTION="description information")
```

In an environment when one Connect:Direct system (either a Connect:Direct/Plex environment or Connect:Direct/Stand-alone Server) communicates with a Connect:Direct/Plex environment, add the following statement to each Connect:Direct ADJACENT.NODE entry that defines the other Connect:Direct/Manager.

```
ENVIRONMENT=ZOS
```

**Note:** When an OS/390 or z/OS Connect:Direct system communicates with another Connect:Direct system in a Connect:Direct/Plex environment and ENVIRONMENT=ZOS|OS390 is not specified, Process redirection does not function correctly.

See *TCP/IP Considerations* on page 142 for network map entry requirements for TCP/IP nodes.

See *Channel-to-Channel Support* on page 147 for a discussion of channel-to-channel support.

See *Examples of Local and Adjacent Node Records* on page 150 for examples.

## Types of Parameters

The network map adjacent node entry contains positional and keyword parameters. Both parameter types are described in the following sections.

### Positional Parameters

The following are the positional parameters for the network map adjacent node entry:

| Parameter | Description |
|---|---|
| nodename | The first positional parameter is the 1–16 alphanumeric character node name. This name represents the partner Connect:Direct and is used in communications with the local Connect:Direct. This parameter is required. The node name is always changed to upper case in the network map, regardless of the remote platform.<br>The following characters are allowed:<br>A-Z, 0-9, !, @, #, $, %, &, {, }, +, -, ^<br>Connect:Direct for z/OS does not accept the following characters for the adjacent node:<br>(, ) =, \, “, ‘, <, >, \|, \|\| |

| Parameter | Description |
|---|---|
| communications name \| channel-range-start -addr | The second positional parameter is the 1–8 alphanumeric character communications name. It specifies the network name of the partner Connect:Direct. It can be an SNA VTAM APPLID or a TCP/IP port number. This parameter is optional.<br><br>For **SNA,** this field must contain the VTAM APPLID of the remote Connect:Direct node that the local DTF uses for DTF-to-DTF communications with the remote node.<br>This name is the same name that is defined for the communications name in the network map of the remote Connect:Direct node.<br><br>For **OpenVMS** and **HP NonStop**, use the PNODE.LUS and SNODE.LUS parameters and leave this field blank.<br><br>For **TCP/IP**, this field contains the TCP/IP port number of the remote partner Connect:Direct. You do not need to use this field if the partner Connect:Direct is initialized using the default TCP/IP port number. This port number does not change the port number for the host Connect:Direct that is defined at initialization. See page 144 for more information about the TCP/IP port number.<br><br>For **CTCA**, this field contains the CCUU of the first CTCA address used by this node. |
| remote library name \| IP address or Alias \| addr-count | The third positional parameter is for the Host ID (DNS), or IP Address, or Library name. This is for OS/400 or TCP/IP nodes only.<br>The positional For **OS/400** nodes, this parameter specifies the name of the library where the Connect:Direct for OS/400 program SMMAIN resides for the partner Connect:Direct.<br>For **TCP/IP** nodes, this parameter specifies the IPv4 address (1-15 dotted decimal characters), DNS alias name (1-16 alphanumeric characters) or the IPv6 address (1-39 colon separated hexadecimal characters) to establish the TCP/IP session.<br>For DNS alias names greater than 16 characters, leave this field blank and use the LDSN parameter.<br>For **CTCA**, this parameter specifies the number of CTCA addresses used by this node. Specify this with an even number value (a minimum of 2). |
| session type | The fourth positional parameter is 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:<br>SNA (for LU0 protocol)<br>SNUF (for LU0 protocol for the OS/400)<br>LU62 (for LU6.2 protocol)<br>TCP (for TCP/IP protocol)<br>UDT33 (for UDT protocol)<br>CTCA (for Channel to Channel connections) |

| Parameter | Description |
|---|---|
| security node type | The fifth positional parameter is the security node type. It classifies the node as an internal or external node. Specify this parameter for Trusted Node security. It is optional if you do not use Trusted Node security.<br>For further information on Trusted Node security refer to *Trusted Node Security* on page 80.<br>**EXTERNAL\|EXT** specifies an external security classification for this node.<br>**INTERNAL\|INT** specifies an internal security classification for this node. This is the default. |
| data direction restriction | The sixth positional parameter is 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 *Data Direction Restriction* on page 81. This parameter is optional. Valid data direction values are:<br>**RECEIVE\|RECV** indicates that when the adjacent node initiates a transfer, it is only allowed to receive data from this node. It is never allowed to send data to this node.<br>**Note:** For **CTCA**, **RECEIVE\|RECV** indicates that the first address in channel-range-start-addr is used for inbound traffic; the next address is used for outbound traffic. For two Connect:Direct systems to communicate through CTCA, one adjacent node must specify **SEND** and the other must specify **RECEIVE\|RECV**.<br>**SEND** indicates that when the adjacent node initiates a transfer, it is only allowed to send data to this node. It is never allowed to receive data from this node.<br>**Note:** For **CTCA**, **SEND** specifies that the first address specified in channel-range-start-addr is is used for outbound traffic; the next address is used for inbound traffic.<br>For two Connect:Direct systems to communicate through CTCA, one adjacent node must specify **SEND** and the other must specify **RECEIVE\|RECV**.<br>**BOTH i**ndicates that when the adjacent node initiates a transfer, it is allowed to both send and receive data from this node. This value is the default.<br>**NONE** indicates that when the adjacent node initiates a transfer, it is neither allowed to send or receive data from this node. |

### Keyword Parameters

The following are the keyword parameters for the network map adjacent node entry.

| Parameter | Description |
|---|---|
| PARSESS=(max,default) | This is an optional parameter that defines two session control values for parallel session-capable nodes. PARSESS controls the number of Processes initiated by one node. PARSESS does not control the total number of Processes submitted.<br>**max** specifies the maximum number of simultaneous DTF-to-DTF sessions that the local Connect:Direct node can initiate with this adjacent node. The range of this subparameter is 2–255. 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.<br>**default** specifies the class assigned to a Process if one is not specified on the Process statement or when the Process is submitted. The range of this subparameter is 1-the value coded for parallel sessions. If you do not code this parameter, the node is not parallel session-capable, and the max and default values are set to 1. This parameter is required if you do PNODE=SNODE processing.<br>**Note:** If you do not specifically code the PARSESS parameter on an adjacent node definition, a default of PARSESS=(1,0) is set by the system, which allows only a single session. The default class 0 means no CLASS which shows up as a CLASS of NONE in an executing Process. You cannot code the default PARSESS=(1,0) on an ADJACENT.NODE definition—the minimum value that can be coded is PARSESS=(2,1).<br>**Note:** The following formula is a quick way to determine the number of parallel sessions available when class is not defined for a Process (the default class is used): Default Parsess value = (max – default) +1<br>**Note:** For CTCA connections, this value is automatically set to 1/2 of the addr-count. Do not try to reset this parameter for CTCA connections.<br>**Note:** For best results when using Independent LU6.2, code PARSESS the same for both the local and remote nodes.<br>Selecting a Class<br>Connect:Direct selects a class in which a Process is to run by starting with the default class, or the coded class, and proceeds upward in class values until an available class slot is found, or until all possible class values are tested.<br>For more information on Process class, refer to the Connect:Direct Processes Web site at<br>http://www.sterlingcommerce.com/documentation/processes/processhome.html.<br>For more information about parallel sessions, refer to *Building,Modifying, and Submitting Processes Queueing* in *Connect:Direct for z/OS User's Guide*. |

| Parameter | Description |
|---|---|
| SOURCEIP = IP address | If an IPv6 or IPv4 address is required for the network map, this parameter defines the source IP address that is bound to during outbound connection requests. If the SOURCEIP parameter is not specified, the local or default address is bound. The destination address is obtained from the network map. The source port is obtained as assigned by TCP/IP, or through the TCP.SRC.PORTS or UDP.SRC.PORTS table. If the address from the SOURCEIP cannot be bound during Process execution, the Process is placed in WA or WT state.<br>**Note:** If the network map adjacent node contains a IPv6 address, but Connect:Direct is not enabled to support it, the submit of a Process for that network map entry fails. For information about enabling Connect:Direct for IPv6 support, refer to *TCP/IP Addressing* on page 142. |
| SESS.SNODE.MAX = nnn | This is an optional parameter that can be used to control the number of concurrent sessions that an adjacent node can initiate as the SNODE. The range for SESS.SNODE.MAX is 1 to 255. The default value is 255. In a Connect:Direct/Plex environment, this parameter controls the number of sessions that an adjacent node can initiate with each Connect:Direct/Plex server.<br>**Note:** For use with TCP/IP and LU6.2 only. |
| LDNS=hostname | LDNS is an optional parameter that specifies the host name in the network map adjacent node entry.<br>If you use this parameter, you can leave the third positional parameter (IP address or alias) blank. You must also specify TCP as the fourth positional parameter (session type).<br>**hostname** specifies the host name for this node. The host name length is from 1–256 characters.<br>To see an example of a long DNS record, see *The Network Map* in *Connect:Direct for z/OS User's Guide.* |
| ENVIRONMENT= operating environment | The ENVIRONMENT parameter identifies the adjacent node operating system environment. This parameter is required when the session type positional parameter specifies LU6.2 protocol for OS/400 systems and when one Connect:Direct system (either a Connect:Direct/Plex environment or Connect:Direct/Stand-alone Server) communicates with a Connect:Direct/Plex environment. Other protocols can use it for documentation purposes.<br>Valid values are: GIS, HPNONSTOP, I5OS, LINUX, OPENVMS, OS400, SELECT, VOS, STRATUS, UNIX, VM, VMS, VSE, WINDOWS, and ZOS.<br>**Note:** When an OS/390 or z/OS Connect:Direct system communicates with another Connect:Direct system in a Connect:Direct/Plex environment and ENVIRONMENT=ZOS\|OS390 is not specified, Process redirection does not function correctly. |

| Parameter | Description |
|---|---|
| LOGMODE=logmode entry name | The LOGMODE parameter identifies the VTAM logmode entry that defines the communication protocol for this node.<br>This parameter is only required for LU6.2 nodes.<br>It is optional for LU0 connections. If you specify this parameter for LU0 connections, 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 determines the RUSIZE. For a host-to-PC transfer, the LOGMODE entry in the host VTAM MODETAB is used.<br>This parameter is not valid for TCP/IP nodes or CTCA connections.<br>Refer to the *Planning the Installation* chapter and the *Sample VTAM Definitions* appendix of the *Connect:Direct for z/OS Installation Guide* for more information about VTAM definitions. |
| APPLIDS=(vtam applid1 [,vtam applid2,...] ) | The APPLIDS parameter specifies the VTAM APPLIDs that establish a session between the local Connect:Direct SNA API and the remote Connect:Direct DTF. APPLIDs are defined on the local node. This parameter is valid only for z/OS, VSE/ESA, and VM/ESA nodes. |
| NETID=networkid \| servername | The NETID parameter specifies the 1–8 character network ID for this node. When a Process starts, the network ID provided at the session start is verified against the network ID in the network map for the adjacent node. If they do not match, the Process is terminated with an error.<br>For multiple session signons, the network ID of the node signing on is verified against the network map network ID of the node being signed on to. If they do not match, the signon fails.<br>If this keyword is not coded or the Connect:Direct initialization parameter NETMAP.CHECK is set to NO, the network ID is not checked at Process start or multiple session signon.<br>For a CTCA connection in a Connect:Direct/Plex environment, this parameter specifies the 1–8 character name of the Connect:Direct/Server. |
| PNODE.LUS=(luname1 [,luname2,...] ) ) | The PNODE.LUS parameter 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.<br>This parameter applies to OpenVMS nodes. It is not valid for CTCA connections. |

| Parameter | Description |
|---|---|
| SNODE.LUS=(luname1 [,luname2,...] ) ) | For OpenVMS, the SNODE.LUS parameter specifies the logical unit names used by the local node to initiate a session with this remote node. For all other platforms, it specifies the logical units used for all communications with the remote node.<br>Communications to nodes that cannot handle parallel sessions can require 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 communicate with the local node must be defined to the local node on the corresponding adjacent node network map entry.<br>HP NonStop and OS/400 adjacent node entries use the SNODE.LUS keyword only to define the LU pool.<br>OpenVMS nodes assign the logical units in the pool as either ACTIVE, session initiating, PASSIVE, or listening for session requests. This distinction in function is defined to the z/OS node by specifying the ACTIVE logical units with the PNODE.LUS keyword and the PASSIVE logical units with the SNODE.LUS keyword.<br>This parameter is not valid for CTCA connections. |
| USE.SERVER.NODE = YES \| NO | Setting this parameter to YES tells a Connect:Direct/Server to use its CDPLEX.SERVER.NODE initialization parameter as identification when communicating with this adjacent node. If this parameter is set to NO, the Connect:Direct/Server identifies itself to this adjacent node as the same local node as all other members of the Connect:Direct/Plex environment. (See *CDPLEX.SERVER.NODE = node name* on page 440 for more information.)<br>This parameter is required if a remote Connect:Direct system, using NETMAP checking, communicates with more than one Connect:Direct/Server in a Connect:Direct/Plex environment.<br>This parameter is ignored in a Connect:Direct/Stand-alone Server. |
| TCPAPI=(port number, IP address) | This parameter defines the adjacent node communication address used by an external API that uses TCP to communicate with the node. This value must be the same as the TCP.API.PORTNUM initialization parameter of the node that you are communicating with.<br>If the adjacent node is an SNA node, include both the port number and IP address.<br>If the adjacent node is a TCP node, you must supply the port number, but do not need to supply the IP address. If you do not supply the IP address, you must define the IP address in the Adjacent node record. See *API Signons* on page 148 for more information. |
| CRC =(OFF\|ON\|DEFAULT) | Determines if you will perform CRC checking for any TCP/IP Process sending to this node. If overrides are allowed, this parameter enables you to override the CRC setting in the initialization parameters for this node.<br>**Note:** Although DEFAULT is an acceptable value, it is the equivalent of not specifying the network map parameter at all and would normally only be seen in the output of the JNETUNLD (network map file unload) when no CRC specification had been supplied previously. |

| Parameter | Description |
|---|---|
| PLEXCLASS= (* \| plexclass, * \| plexclass) | This parameter specifies a default PLEXCLASS for the PNODE (the first parameter) and SNODE (the second parameter). This Connect:Direct/Server checks this PLEXCLASS to determine if it can run the Process. (See the CDPLEX.PLEXCLASS= initialization parameter on page 438 for more information.)<br>Each PLEXCLASS name is 1–8 characters long. An asterisk (*) is also an acceptable entry, which indicates that the Connect:Direct/Server supports any Process that does not specify a PLEXCLASS, or specifies a PLEXCLASS of '*'. |
| BUFFER.SIZE=V2.buffer override | V2.buffer override defines the buffer size for the adjacent node. It is only valid for TCP and CTCA session types. Use it to dynamically override the V2.BUFSIZE value of the local node during TCP buffer size negotiation. This parameter is generally coded to reduce the V2.BUFSIZE value so that Connect:Direct provides less data on each TCP send. Valid values are 3072–262144 \| 3K–256K. |

| Parameter | Description |
|---|---|
| ALTernate.COMMinfo=<br>(ALTernate.RESTART=YES \| NO<br>ALTernate.DIRection=<br>BALANCE \| TOP,<br>(ALTernate.ADDRess= \|<br>ALTernate.NODEDEF=,<br>ALTernate.PORT=,<br>SOURCEIP=,<br>ALTernate.TYPE=SNA \|<br>TCP \| UDT33 \| LU62,<br>ALTernate.LOGmode=<br>logmode entry name,<br>ALTernate.USE.OUTbound<br>=<br>Yes \| No)) | This parameter enables you to specify multiple remote node addresses for NETMAP checking.<br>**ALTernate.RESTART** indicates if an alternate communications path is considered when a Process must restart.<br>**Note:** Do not specify ALT.NODEDEF if you use ALT.RESTART. All alternate communications paths must use ALT.ADDR and the same protocol (all Version 1 or Version 2). For example, if you use ALT.TYPE=SNA, specify all alternate addresses as ALT.TYPE=SNA (Version 1 protocol). Since LU62, TCP, and UDT33 are all Version 2 protocols, you can use LU62, TCP, or UDT33 when specifying alternate addresses. (For a more information on Version 1 and 2 flows, see Version 1/Version 2 Flows on page 462.)<br>**ALTernate.DIRection** specifies the direction of the communications paths. BALANCE (default) indicates a balanced approach; that is, all current sessions are scanned for this same adjacent node and the least used path is selected as the primary path for the Process. The list is processed forward from there. **TOP** indicates that the paths are processed from the top down.<br>**ALTernate.ADDRess** specifies either a TCP/IP address or an SNA address as appropriate.<br>If ALT.TYPE=SNA or ALT.TYPE=LU62, ALT.ADDR must be a 1–8 character APPLID. If ALT.TYPE=TCP, ALT.ADDR must be a 15-character IPv4 or 39-character IPv6 TCP/IP address or a 1–256 character LDNS name. This subparameter is required if you do not specify ALTernate.NODEDEF.<br>**ALTernate.NODEDEF** specifies an alternate node definition to use for NETMAP checking. This subparameter references another entry in the NETMAP. This subparameter is required if you do not specify ALTernate.ADDRess.<br>**ALTernate.PORT** specifies the alternate address port number if the alternate address is TCP/IP. This field is not required. This subparameter defaults to the port of the adjacent node record. If not specified in the adjacent node record, the default is 1364. You can only use this subparameter if ALT.TYPE=TCP.<br>**Note:** If ALTernate.DIRection=BALANCE is specified, the value specified for ALTernate.PORT is not used.<br>**SourceIP**, which has no short form, specifies an alternate IPv6 or IPv4 address that is bound to during outbound connection requests. The source port is obtained as assigned by TCP/IP, or through the TCP.SRC.PORTS or UDP.SRC.PORTS table. If the address from the SOURCEIP cannot be bound during Process execution, the Process is placed in WA or WT state.<br>If SourceIP is specified for this ADJACENT NODE but not in the ALT.COMM entry, the value specified for the ADJACENT NODE also applies to this ALT.COMM entry.<br>**Note:** If the network map adjacent node contains a IPv6 address, but Connect:Direct is not enabled to support it, the submit of a Process for that network map entry fails. For information about enabling Connect:Direct for IPv6 support, refer to *TCP/IP Addressing* on page 142.<br>**ALTernate.TYPE** specifies the protocol used for the alternate address. This value defaults to that of the adjacent node record. Valid values are SNA, TCP, UDT33, LU62. This subparameter is only used with ALTernate.ADDRess. This subparameter is required if you do not specify ALTernate.NODEDEF. |

| Parameter | Description |
|---|---|
| ALTernate.COMMinfo=<br>(ALTernate.RESTART=YES \| NO<br>ALTernate.DIRection=<br>BALANCE \| TOP),<br>(ALTernate.ADDRess \|<br>ALTernate.NODEDEF,<br>ALTernate.PORT,<br>SOURCEIP,<br>ALTernate.TYPE=SNA \|<br>TCP \| LU62,<br>ALTernate.LOGmode=<br>logmode entry name,<br>ALTernate.USE.OUTbound<br>=Yes \| No)<br>(Continued from previous page) | **ALTernate.LOGmode** specifies an SNA logmode used when ALTernate.TYPE=SNA or LU62. This parameter is required for LU62 if the adjacent node is not defined as LU62.<br>**ALTernate.USE.OUTbound** specifies whether the alternate communications path is used for outbound Processes, providing session failover for Processes sent to this adjacent node. Valid values are Yes (default) and No.<br>For more information on this parameter, refer to *Using Alternate Communications Paths* on page 140 and *Alternate Communications Path Examples* on page 150. |
| CONTACT.NAME=”name”<br>CONTACT.PHONE=”phone information”<br>DESCRIPTION=”description information” | These are free-form text parameters, which provide additional general information about an Adjacent Node entry. The CONTACT.NAME and CONTACT.PHONE parameters are limited to a maximum of 40 characters. The DESCRIPTION parameter is limited to a maximum of 255 characters. |

## Using Alternate Communications Paths

This section describes how to use the ALT.COMM parameter to set up alternate communications paths. Refer to *Examples of Local and Adjacent Node Records* on page 150 for examples. To see an example of using alternate communications paths in a Connect:Direct/Plex environment, see *Strategies for Communicating with Non-Plex Servers* on page 176.

Note the following:

- The protocol and number of alternate communications paths depend on the capability of the remote node.
- If you use alternate communications paths, the remote Connect:Direct node must have the capability to support the ALT.COMM parameter and have a defined method of performing network map checking for these alternate paths.
- If you are using the STS protcol with Connect:Direct Secure+ Option for z/OS, all communications paths must be LU6.2, TCP, or UDT. If you are using either the SSL or TLS protocol, all communications paths must be TCP or UDT.
- Do not define PARSESS for each alternate communications path.

### Alternate Communications Paths with Current Node as Primary Node

The adjacent node record defines a communications path. You can use the ALT.COMM parameter in the adjacent node record to define alternate communications paths.

When a Process is submitted, the primary communications path is selected based on the alternate.direction subparameter. If alternate.direction=BAL, all communications paths are scanned to find the least used path, including the path defined in the adjacent node record, and all paths defined in the alternate.address subparameters. The least used path becomes the primary path. If alternate.direction=TOP, the communications path defined in the adjacent node record is used as the primary path.

If the Process fails to establish a session using the primary communications path, the network map is processed to determine the next eligible communications path. This is repeated until the session is successfully established or the number of communications paths is reached. When this number is reached, the primary communications path is restored and the Process is placed in timer retry (TI RE) status. After all session retries (MAXRETRIES) are reached, the original communications path is restored and the Process is placed in hold (HO WC) status.

Each failed attempt to establish a session produces the following result:

- Message SVTM310I SESSION NOT ESTABLISHED WITH SNODE=xxxx
- Session Begin statistics record that identifies the communications path used at the time of the failure
- Diagnostics to RPLERRCK
- When the MAXRETRIES is reached and the Process is placed in the HOLD queue, the following message is issued in addition to the previous message(s): SVTM105I PNAME=pnam, PNUM=pnum MOVED TO Q=HOLD, QSTATUS=WC

When the Process establishes a session, the Session Begin statistics record identifies the communications path that was successfully used. That communications path is used for the life of the Process.

### Alternate communications Paths with Current Node as Secondary Node

When the current Connect:Direct node is the secondary node (SNODE), the ALT.COMM parameter is used for network map checking purposes only.

### Outbound Processes

The ALT.COMM parameter is used for outbound Processes when all of the following conditions are true:

- The current Connect:Direct is the PNODE.
- The adjacent node entry indicates that the remote node is the SNODE.
- The Process is not in restart (the default ALT.RESTART=NO is in effect).

**Note:** If you specify ALT.RESTART=YES, Connect:Direct considers alternate communications paths if the path used to establish a session fails and the Process restarts. For an example, see *Restarting Processes using an Alternate Communications Path* on page 151.

- The Process is not PNODE=SNODE.
- The Process is not SNODE=TCPNAME=.

## PNODE=SNODE Processing

Connect:Direct can initiate Processes where the local node is both the initiating and target node (PNODE and SNODE). You enable PNODE=SNODE processing by creating an adjacent node entry with the same node name as the local node entry in the network map.

If the PNODE=SNODE connection is SNA, SNA=YES is specified in the initialization parameters. If you are using TCP, refer to *Configure Connect:Direct without SNA Support* in the *Connect:Direct for z/OS Installation Guide*.

Observe the following rules when setting up the adjacent node:

- Define an LU0 communications name for the PNODE=SNODE network map entry. Do not specify an LU6.2 logmode entry name for the common name of the adjacent node entry.
- The communications name for the adjacent node must be different from the communications name of the local node. If the names are the same, PNODE=SNODE processing is disabled at initialization.
- 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 134.

## TCP/IP Considerations

A network map entry is not required for every TCP/IP node. A default entry provides 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. For more information, see *Defining a TCP/IP Default Entry* on page 147.

---

**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 and the remote library name positional parameter are not used on any TCP/IP node and cannot be coded on the network map. A warning is generated for any unnecessary 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 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.

Connect:Direct supports both IPv4 and IPv6 protocols, where IPv6 allows a much larger address range than IPv4. During product initialization, Connect:Direct determines if the TCP Stack is IPv6 or only IPv4. (IPv6 must be enabled within TCP/IP itself.) If Connect:Direct initializes on a system

where IPv6 is not enabled, any function involving an IPv6 address fails including attempts to resolve an address or name using the DNS name resolution. For information about enhancing and extending IPv6 through the TCP.LISTEN parameter, see *Multiple Port TCP/IP Listen* on page 145.

### Command Syntax

The following examples demonstrate the command syntax for IPv4 and IPv6 addressing.

**Note:** The address must be specified on a single line. You cannot break the address between lines.

The following example demonstrates how to break the command syntax over an additional line:

```
(SC.DUB.MWATL3,4399,   -
1111:2222:3333:4444:5555:6666:7777:8888,TCP)
```

The following example demonstrates the appropriate network syntax:

```
/*                */
/*  IPV6 ADDRESS  */
/*                */
   ADJACENT.NODE=(    -
     (SC.DUB.MWATL3,4399,1111:2222:3333:4444:5555:6666:7777:8888,TCP) -
     PARSESS=(00000255 00000001) -
     APPLIDS=(M1CDI7P6 M1CDI7P7 M1CDI7P8) )
/*                */
/*  IPV4 ADDRESS  */
/*                */
   ADJACENT.NODE=(    -
     (SC.DUB.MWATL3,4399,111.222.333.444,TCP) -
     PARSESS=(00000255 00000001) -
     APPLIDS=(M1CDI7P6 M1CDI7P7 M1CDI7P8) )
```

The following example demonstrates the appropriate command syntax for a multiple listen for IPv6 addresses through the initialization parameter:

```
TCP.LISTEN=((10.20.201.2,4199),  -
            (fd00:0:0:20cc::2,4299),  -
            (10.20.201.2,4399),  -
            (10.20.201.2,4499) )
```

The following two examples represent a simple specification for a single listen for either an IPv4 or IPv6 address through the initialization parameters:

```
/*                */
/*  IPV4 address  */
/*                */
    TCP.LISTEN = (111.222.333.444,01364)
```

```
/*                */
/*  IPV6 address  */
/*                */
    TCP.LISTEN = (1111:2222:3333:4444:5555:6666:7777:8888,01364)
```

The following example represents a simple specification for a single listen for either an IPv4 or IPv6 address through the Process syntax:

```
/*                 */
/*  IPV6 Address   */
/*                 */
label PROCESS   -
  SNODE=TCPNAME=1111:2222:3333:4444:5555:6666:7777:8888;nnnnn
/*                 */
/*  IPV4 Address   */
/*                 */
label PROCESS SNODE=TCPNAME=111.222.333.444;nnnnn

(where nnnnn is the 5 digit port number)
```

### Domain Name Resolution

Because IP addresses are difficult to remember and not descriptive of the host it is assigned to, TCP/IP enables meaningful names to map to the IP address. TCP/IP provides a function called Domain Name Resolution to map the name to the IP address. Refer to your TCP/IP implementation documentation for information on the setup and use of Domain Name Resolution.

### Releasing Processes Serially from Connect:Direct Windows

To support dial-up connections from a Connect:Direct Windows node and release Processes serially, define the Connect:Direct Windows node in an adjacent node entry using a null IP address of 0.0.0.0. You can let the port number default to 1364 on the node where the network map resides since it will be resolved at connection time. Processes submitted to nodes defined in this manner default to HOLD=CALL status and are not executed since the connection cannot be resolved.

To release the HOLD=CALL Processes, create a NULL or ENABLE Process from the Connect:Direct Windows SNODE. (A NULL Process is an empty one with no steps and is only valid in Connect:Direct Windows.) When the Connect:Direct Windows node establishes a connection with Connect:Direct for z/OS and sends the NULL Process, Connect:Direct for z/OS uses the same session and runs any HOLD=CALL Processes one at a time. Checkpoint restart is supported for such nodes.

## TCP/IP Port Number

The initialization parameters TCP.LISTEN and TCP.API.LISTEN provide the best method for defining a single port or multiple ports for Connect:Direct to listen to and accept incoming TCP/IP connection requests. You can define a single port for establishing a listen task, or define multiple ports by establishing a list of IP address and port number combinations. Multiple port listening allows Connect:Direct to accept incoming traffic from a variety of addresses.

The initialization parameters TCP.ADDR, TCP.NAME, TCP.PORTNUM, CDPLEX.TCPIP, and CDPLEX.TCPNAME are used to define a single port for establishing a listen task, and are mutually exclusive with TCP.LISTEN. The initialization parameter TCP.API.PORTNUM is used to define a single port, and is mutually exclusive with TCP.API.LISTEN. Combining these mutually exclusive parameters results in a warning message issued to NDMLOG. To avoid these messages, use TCP.LISTEN or TCP.API.LISTEN to specify listening tasks.

## Single Port TCP/IP Listen

Connect:Direct implements a standard TCP client/server architecture. Each Connect:Direct node 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 that is established through the TCP/IP network to a Connect:Direct server. When 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 uses a predefined TCP/IP port number for both client and server. This port is defined in TCP/IP as a TCP service and requires the same port for all Connect:Direct TCP/IP nodes in your network. Refer to your TCP/IP implementation documentation for how to define TCP servers. If you cannot use the predefined TCP/IP port, you can override it for both the client and server.

In addition, TCP.API.PORTNUM is used to establish a listen task for API connections only. This initialization parameter defaults to port number 1363.

### 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 does not change the port number used by the server functions. You must code the network map entry in your network map in order to use the override port number.

### Server Override Port Number

The server override port number is coded in 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 does not change the port number used by the client functions.

## Multiple Port TCP/IP Listen

Connect:Direct accepts incoming traffic from a variety of addresses. The initialization parameters TCP.LISTEN and TCP.API.LISTEN support both IPv4 and IPv6 protocols. They also allow for a list of IP address and port number combinations. These parameters are mutually exclusive with the TCP/IP single port parameters. If the single port parameters are specified without the multiple port parameters, a single listen is established. However, if both types of parameters are specified, initialization terminates with a warning message. To avoid these messages, migrate to use TCP.LISTEN or TCP.API.LISTEN to specify single listen and multiple listening tasks.

The first address defined in the parameter becomes the local or default address. Up to eight different addresses/ports can be defined for each server.

For information about multiple port listening in a Connect:Direct/Plex environment, refer to *Connect:Direct/Plex Overview* on page 167.

### Viewing a TCP Listen Status Report

To view the TCP listening status through the IUI interface:

1. Access the INQUIRE TCP command by selecting option **INQ** from the Connect:Direct Administrative Options Menu. The Inquire DTF Internal Status screen is displayed.
2. Type **ITCP** and press **ENTER** to display the status report. The following report is an example of the output.

```
=====================================================================
CD.ZOS.NODE       *INQUIRE TCP      * DATE: yyyy.mm.dd TIME: hh:mm:ss
SYSTEM INITIALIZED -----------------        yyyy.mm.dd       hh:mm:ss
=====================================================================

Address                   Port Num    Status    Type     Address Family
===================       =========== ========= ======= ===============
10.20.201.2 (Default)      1364        LISTEN    NODE    IPV4
10.20.129.3                4100        LISTEN    NODE    IPV4
10.20.129.3                4101        ERROR     NODE    IPV4
10.20.201.2                1363        LISTEN    API     IPV4
```

### VTAM Independence

VTAM independence enables Connect:Direct to initialize without SNA support. It also enables Connect:Direct to continue functioning if VTAM is not available, and to reattach to VTAM when it is restored.

To use VTAM independence, you must specify SNA=YES in the initialization parameters. You must also specify a valid VTAM APPLID in the network map local node record.

If SNA= YES is specified in the initialization parameters and you try to start Connect:Direct when SNA is not available, or if SNA becomes unavailable during a session, the system prompts the operator for action.

See page 412 for more information about the SNA initialization parameter. See *API Signons* on page 148 for more information about operator actions. Also see *Initializing Connect:Direct without SNA Support* in Chapter 2 of the *Connect:Direct z/OS Installation Guide*.

### TCP/IP API Communications

The TCP.API.PORTNUM or TCP.API.LISTEN initialization parameters enable an API to communicate with a Connect:Direct using TCP/IP via OE sockets. The TCP.API.PORTNUM parameter defines a listen port that is used for API communications.

The TCP.API.LISTEN parameter allows for a list of IP address and port number combinations. For more information, refer to *Multiple Port TCP/IP Listen* on page 145.

Also, you can use the TCPAPI keyword parameter in the adjacent node record to define a default protocol for API communication. See page 137 for details.

The API must also use the SIGNON command to specify a transport type and communications address. See *API Signons* on page 148 for more information. Additionally, refer to *Managing Sessions* in *Connect:Direct for z/OS User's Guide* for more information about the SIGNON command.

### Defining a TCP/IP Default Entry

The following example defines a TCP/IP default record using ADJACENT.NODE entries.

```
/*                                                                  */
/* The following entry is for the TCP/IP default entry              */
/*                                                                  */
 ADJACENT.NODE=(PARSESS=(8,1)                -
  ENVIRONMENT=ZOS                           -
  (TCP.IP.DEFAULT, 2048,, TCP))
```

**Note:** For a TCP/IP default entry, the nodename parameter must be TCP.IP.DEFAULT and the session type must be TCP. A TCP/IP default entry is required if any Process uses SNODE=TCPNAME=.

In the previous example, nodename=TCP.IP.DEFAULT, communications name=2048, and session type=TCP. For additional information about adjacent node entries, see *Adjacent Node Entry* on page 130.

When a Process is submitted, the port number value is determined in the following order:

1. If the node name within a Process is defined in the network map, the port number associated with the node name entry is used.
2. If the node name within a Process is not defined in the network map, the port number associated with the default entry is used.
3. If no port number exists 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 node name within a Process is defined in the network map, the PARSESS value associated with the nodename entry is used.
2. If the node name within a Process is not defined in the network map, the PARSESS value associated with the default entry is used.
3. If no PARSESS value is 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 are single-threaded.

## Channel-to-Channel Support

Connect:Direct for z/OS provides channel-to-channel support for direct channel links between z/OS platforms using the IBM ESCON CTCA or IBM 3088 CTCA support.

The syntax for creating a CTCA adjacent node follows.

```
ADJACENT.NODE=(                                                    -
     (nodename,                                                    -
      channel-range-start-addr, addr-count, CTCA,,                 -
      SEND| RECV))                                                 -
     NETID=server name                                             -
     PARSESS=(max default)                                         -
     ENVIRONMENT=operating environment                             -
                    )
```

# API Signons

The communication address used to establish a connection to Connect:Direct is determined by the TRANSPORT parameter defined in the SIGNON command. The default for the TRANSPORT parameter is NET (NETMAP), which means that the protocol defined in the NETMAP adjacent node entry is used. See *Adjacent Node Definition Examples* on page 149.

| Parameter Value | Description |
|---|---|
| TRANSPORT = NET | Default. When TRANSPORT is defined as NET, the signon process retrieves the adjacent node entry to determine if the TCPAPI parameter has been defined. If TCPAPI exists, then a TCP connection is attempted using the communications address defined. The communications port number is obtained from the TCPAPI parameter, and if the IP address exists in the TCPAPI parameter, it is also used. If the IP address does not exist in the TCPAPI parameter, it must be obtained from either the adjacent node or the LDNS parameter.<br>If the TCPAPI does not exist, the APPLID parameter is retrieved, and SNA is used as the protocol. |
| TRANSPORT = SNA | When TRANSPORT is defined as SNA, the signon process retrieves the adjacent node entry to determine the APPLID parameter, and SNA is used as the protocol. If the APPLID parameter does not exist, then an ESF SIGNON is performed. |
| TRANSPORT = TCP | When the TRANSPORT is defined as TCP, the communications address must be specified on the SIGNON command.<br>To initialize Connect:Direct without SNA support, refer to *Initializing Connect:Direct without SNA Support* in Chapter 2 of the *Connect:Direct z/OS Installation Guide*. |

## Adjacent Node Definition Examples

To only allow SNA API signons:

```
/*  PNODE=SNODE WITH SNA API SIGNON ONLY         */
    ADJACENT.NODE=(( CD.ZOS.NODE,M1DEV93C) PARSESS=(53 2)  -
    APPLIDS=(M1CDI701 M1CDI702 M1CDI703) -
     )
```

To only allow TCP API signons using the IP address:

```
/*  PNODE=SNODE WITH TCP API SIGNON ONLY USING IP ADDRESS    */
    ADJACENT.NODE=(( CD.ZOS.NODE,M1DEV93C) PARSESS=(53 2)  -
    TCPAPI=(4198,111.222.333.444)   -
     )
```

To only allow TCP API signons using the LDNS parameter:

```
/*  PNODE=SNODE WITH TCP API SIGONON ONLY USING LDNS       */
    ADJACENT.NODE=(( CD.ZOS.NODE,M1DEV93C) PARSESS=(53 2)  -
    TCPAPI=(4198,)   -
    LDNS=long.domain.name -
     )
```

To allow both SNA and TCP API signons:

```
/*  PNODE=SNODE WITH BOTH SNA AND TCP API SIGNON          */
    ADJACENT.NODE=(( CD.ZOS.NODE,M1DEV93C) PARSESS=(53 2)  -
    TCPAPI=(4198,111.222.333.444)   -
    APPLIDS=(M1CDI701 M1CDI702 M1CDI703) -
     )

or

/*  PNODE=SNODE WITH BOTH SNA AND TCP API SIGNON          */
    ADJACENT.NODE=(( CD.ZOS.NODE,M1DEV93C) PARSESS=(53 2)  -
    TCPAPI=(4198,)   -
    LDNS=long.domain.name -
    APPLIDS=(M1CDI701 M1CDI702 M1CDI703) -
     )
```

# Examples of Local and Adjacent Node Records

This section contains examples of local and adjacent node records for various platforms.

## Local Node and Corresponding Adjacent Node Record

The following is an example of a local node entry and its corresponding adjacent node entry. Notice that 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.DALLAS.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))
```

The following is an example of a local node entry and its corresponding adjacent node entry where TCPAPI is defined.

```
LOCAL.NODE=((CD.DALLAS,NDMAPP1,,XYZZY) -
       TCQ=(DSC.DALLAS.TCX DSC.DALLAS.TCQ))
/*                                                                  */
/*  THE FOLLOWING ENTRY IS FOR THE LOCAL NODE WHERE TCPAPI IS DEFINED */
/*  FOR USE BY Connect:Direct APIs TO SIGNON USING TCP/IP           */
/*                                                                  */
ADJACENT.NODE=(PARSESS=(12,2) (CD.DALLAS,NDMAPP2) -
       TCPAPI=(1363,111.222.333.444) -
       APPLIDS=(NAI01 NAI02 NAI03 CDDD12 -
                  CDDD17 CDDD18 CDDD32 CDDD41 CDDD42))
```

## Alternate Communications Path Examples

The following examples show two different ways of using alternate communications paths.

### Specifying ALT.ADDR and ALT.NODEDEF

The following is an example of an adjacent node entry that has four alternate communications paths that can be used if the main communication path specified for the adjacent node (TCP/IP address of 199.1.4.1) cannot be used. Three of the alternate communications paths are defined using the ALT.ADDR parameter while one uses the ALT.NODEDEF parameter.

```
ADJACENT.NODE=((CD.NODE,1364,199.1.4.1,TCP)    -
 PARSESS=(6 2) -
ALT.COMM=((ALT.ADDR=1.1.1.2,ALT.TYPE=TCP), -
          (ALT.ADDR=VTAMAPL1,ALT.TYPE=SNA),  -
          (ALT.NODEDEF=TEST.NODE), -
          (ALT.ADDR=1.1.1.3,ALT.PORT=4374,ALT.TYPE=TCP)) -
)
```

### Restarting Processes using an Alternate Communications Path

In the following example, the adjacent node in the previous example has been redefined to allow an alternate communications path which only supports Version 2 flows. In this system, Connect:Direct is running on a machine with two Network Interface Cards (NICs), which means that there are two IP addresses for this machine. On the remote Connect:Direct system, both of these IP addresses are included in the network map adjacent node entry for the original Connect:Direct system. (For a more information on Version 1 and 2 flows, see *Version 1/Version 2 Flows* on page 462.)

In this example, assume a Process is started by the remote Connect:Direct to this adjacent node with two NICs. The first path (TCP/IP address of 1364,199.1.4.1) is selected when the Process establishes its session. For some reason, the NIC fails on the machine, and the Process fails and goes into retry.

If ALT.RESTART=NO (the default), this Process can never restart successfully because it will always use the path used during initial session establishment (in this case, the failed NIC). When ALT.RESTART=YES is specified, the Process will try all ALT.COMM paths and will restart successfully because an additional path is available (ALT.ADDR=1.1.1.3,ALT.PORT=4374). ALT.NODEDEF points to another network entry and only uses the primary address for that node. Connection failure using the primary address does not result in running that node's ALT.COMM parameters (assuming it has any).

```
ADJACENT.NODE=((CD.NODE,1364,199.1.4.1,TCP)    -
 PARSESS=(6 2) -
ALT.COMM=(ALT.RESTART=YES,ALT.DIR=TOP,-
         (ALT.ADDR=1.1.1.3,ALT.PORT=4374,ALT.TYPE=TCP))    -
        )
```

## z/OS Adjacent Node Examples

The following examples show how to define adjacent z/OS nodes.

### SNA LU0

The following example shows an adjacent z/OS node named CD.NYCZOS, with the communications name (VTAM APPLID) of CDDD10. The APPLIDS parameter indicates nine API sessions are possible.

```
ADJACENT.NODE=(PARSESS=(4,2)  (CD.NYCZOS,CDDD10) -
  APPLIDS=(CDAPI01  CDAPI02   CDAPI03  CDAPI04 -
  CDAPI05  CDAPI06   CDAPI07  CDAPI08   CDAPI09))
```

### CTCA

The following example shows an adjacent z/OS node named CD.DALLAS.ZOS1 which uses channel-to-channel adapter addresses E001-E008. The first address of each pair is used for outbound traffic; the second address is used for inbound traffic.

```
ADJACENT.NODE=(PARSESS=(4,1)                                    -
    ENVIRONMENT=ZOS                                            -
    (CD.DALLAS.ZOS1,E001,8,CTCA,,SEND))
```

In the network map entry of the adjacent node, an adjacent node entry for the other side of the CTCA connection is in the following example. The z/OS node named CD.DALLAS.ZOS2 uses channel-to-channel adapter addresses F001–F008. The first address of each pair is used for inbound traffic. The second address of each pair is used for outbound traffic.

```
ADJACENT.NODE=(PARSESS=(4,1)                                          -
    ENVIRONMENT=ZOS -
    NETID=SERVER1                                                      -
    (CD.DALLAS.ZOS2,F001,8,CTCA,,RECV))
```

### TCP/IP and UDT

The following example shows three adjacent node entries with session protocol types of TCP/IP and UDT:

- The first, with a TCP/IP net name of ZOS.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 ZOS.CD.DALLAS, specifies a UDT port number of 4444. The IP address will default to the node name, to be resolved by domain name resolution.
- The third, with a TCP/IP net name of ZOS.CD.AUSTIN, specifies a TCP/IP port number of 4443 and an IP address of 199.8.8.8.
- The fourth defines a TCPAPI to use port number 4442 and obtain the IP address from the adjacent node record.
- The fifth defines the LDNS parameter.

```
ADJACENT.NODE=(PARSESS=(4,2) -
  (ZOS.CD.CHICAGO,,199.1.4.51,TCP) -
  ENVIRONMENT=ZOS)

ADJACENT.NODE=(PARSESS=(4,2) -
  (ZOS.CD.DALLAS,4444,,UDT33) -
  ENVIRONMENT=ZOS)

ADJACENT.NODE=(PARSESS=(4,2) -
  (ZOS.CD.AUSTIN,4443,199.8.8.8,TCP) -
  ENVIRONMENT=ZOS)

ADJACENT.NODE=(PARSESS=(4,2) -
  (ZOS.CD.AUSTIN,4443,199.8.8.8,TCP) -
  TCPAPI=(4442,) -
  ENVIRONMENT=ZOS)

ADJACENT.NODE=(PARSESS=(4,2) -
  (ZOS.CD.AUSTIN,4443, , TCP)  -
  LDNS=(TCP.AUSTIN.DOMAIN)   -
  TCPAPI=(4442, )  -
  ENVIRONMENT=ZOS)
```

### SNA LU6.2

The following example shows an adjacent node entry for a node named CD.LAZOS with a communications name (APPLID) of APPLLAI and a session protocol type of LU6.2. The operating

environment of this adjacent node is z/OS, and the VTAM logmode entry which defines the session protocol used when communicating with this node is LU62MOD4. The LOGMODE parameter is required for LU6.2.

```
ADJACENT.NODE=(PARSESS=(4,2) -
  (CD.LAZOS,APPLLAI,,LU62) -
  ENVIRONMENT=ZOS LOGMODE=LU62MOD4 -
  APPLIDS=(CDDD2,CDDD3,CDDD4))
```

### Trusted Node

The following example shows an adjacent node entry for a node named SC.NODE.A with a security type of external (EXT) and data direction restriction of SEND.

```
ADJACENT.NODE=(PARSESS=(4,2)  -
           (SC.NODE.A,NZOSD20,,,EXT,SEND) -
           APPLIDS=(NZOSA36,NZOSA37,NZOSA38))
```

## VM/ESA 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/ESA 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))
```

## Windows Adjacent Node Examples

The following example show how to define adjacent Windows nodes.

```
ADJACENT.NODE=(                                              -
                (WIN.TCPIP.NODE,1364,123.4.5.67,TCP)       -
                PARSESS=(20,1)                               -
                ENVIRONMENT=WINDOWS -
               )
```

### UNIX Adjacent Node Examples

The following examples show how to define adjacent UNIX nodes.

#### TCP/IP

The following example shows two adjacent node entries with session protocol types of TCP/IP:

- The first entry specifies the default TCP/IP port number by leaving the communications name positional parameter null. The IP address will default to the node name to be resolved by domain name resolution.
- The second entry 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 no APPLID or LOGMODE keywords are 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)
```

## Stratus VOS Adjacent Node Examples

The following examples show how to define adjacent Stratus VOS nodes.

### TCP/IP

The following example shows an adjacent node entry with a session protocol type of TCP/IP, a TCP net name of CD.STRAT, a TCP port number of 3333, and an IP address of 199.1.1.11.

```
ADJACENT.NODE=(PARSESS=(12,1) -
     (CD.STRAT,3333,199.1.1.11,TCP) -
     ENVIRONMENT=STRATUS)
```

### LU0

The following example shows an adjacent Stratus VOS node with a communications name (APPLID) of M1T20404 and a session protocol type of LU0. The logmode entry name is CDPCLU0. The LOGMODE parameter is optional for LU0.

```
ADJACENT.NODE=(( CD.STRAT,M1T20404,,SNA) -
      LOGMODE=CDPCLU0)
```

## 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, a 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 the 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=OS400 parameter is required for OS/400 nodes using the LU6.2 protocol. The LOGMODE parameter is required for the LU6.2 protocol.

```
ADJACENT.NODE=(PARSESS=(4,2) -
  (AS400.CD.NY,,CDLIB1,LU62) -
  ENVIRONMENT=OS400 LOGMODE=LU62MOD3 -
  SNODE.LUS=(NYLU01,NYLU02,NYLU03,NYLU04))
```

### TCP/IP

The following example shows an adjacent node entry with a node name of OS400.TCP.NODE, session protocol type of TCP, a TCP port number of 1364, and an IP address of 199.1.1.11.

```
ADJACENT.NODE=( -
  (OS400.TCP.NODE,1364,199.1.1.11,TCP,INT,BOTH) -
  CRC=DEFAULT -
  PARSESS=(20,2))
```

# Updating the Network Map

The network map is created during installation, when the network map source is input to the network map load program.

The network map source contains one local node entry and multiple adjacent node entries. It can contain $$ACTION verbs added during previous maintenance.

You can update the network map source while Connect:Direct is not executing or dynamically while Connect:Direct is executing.

**Note:** To test connectivity to an ADJACENT.NODE, use the HARTBEAT Process (see $CD.SAMPLIB). This Process makes a connection to the ADJACENT.NODE and executes a RUN TASK at the local node. Be aware that the connection causes both Connect:Direct nodes to search their TCQ files for Processes destined for the other node.

## Updating the Network Map while Connect:Direct is Not Executing

You can update the network map using the network map source and the JNETLOAD JCL which loaded the source at initialization.

You can only update local nodes by performing the following steps:

1. Change the network map source. The network map source is loaded at installation into $CD.CNTL(NETMAP01)

2. Stop Connect:Direct.
3. Delete and redefine the network map. Refer to the JCL in $CD.JCL(JNETDEF).
4. Reload the network map. Refer to the JCL in $CD.JCL(JNETLOAD).
5. Restart Connect:Direct.

## Updating the Network Map while Connect:Direct is Running

You can update the network map without deleting and redefining it. In addition, you can update the network map source without stopping Connect:Direct, by using the UPDATE NETMAP command.

After updating the network map with UPDATE NETMAP, you can refresh the network map for any Processes in the Wait queue with the Change Process command. See the *Connect:Direct for z/OS User's Guide* for a description of the Change Process command.

**Note:** Any changes to the CTCA definition do not take effect until Connect:Direct is reinitialized.

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 nodes.

The format of the UPDATE NETMAP command follows.

| Label | Command | Parameters |
|---|---|---|
| (optional) | **UPDATE NETMAP** | **WHERE (** |
| | | **NETINput** = filename(member name) |
| | | NETLOG =[ddname \| NONE] |
| | | **)** |
| | | DIS \| PRT |

**Note:** You must reinitialize Connect:Direct before CTCA definition changes or additions are effective.

### Required Parameter

WHERE is the only required parameter for the UPDATE NETMAP command.

| Parameter | Description |
|---|---|
| WHERE<br>(NETINput = filename (member name)<br>NETLOG = [ddname \| NONE] ) | This parameter specifies the network map source file and where the update activity is reported.<br>**NETINput = filename (member name)** specifies the name of the network map source file. This file can be sequential or a PDS member. The network map source can contain multiple basic action verbs, multiple special purpose action verbs, or a combination of both. The source that updates the network map can be the entire network map source or a subset of it.<br>NETLOG = [ddname \| NONE] specifies where the update activity is reported.<br>ddname specifies the data definition name allocated to the Connect:Direct DTF where the update activity is reported. The first two characters of the ddname must be CD.<br>NONE specifies that no update activity is reported.<br>If the parameter is left blank the update activity is reported to the CDLOG data set. Regardless of which option is selected the activity is recorded in the Statistics file as WTO records. |

### Optional Parameter

| Parameter | Description |
|---|---|
| DIS \| PRT | Use the DIS or the PRT optional parameter to specify the output destination.<br>**DIS** indicates that the activity is reported in display format, either to the screen for IUI requests or to the DDNAME for batch requests.<br>**PRT** indicates that the output is 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 as described in the description of the NETINPUT parameter of the UPDATE NETMAP command beginning on page 157. Each verb defines the action to take for the node entry immediately following the action verb. Three basic action verbs and three special purpose action verb pairs exist. The following table describes the action verbs:

| Action Verb | Description |
|---|---|
| $$INSERT | Inserts the following node entry into the network map. |

| Action Verb | Description |
|---|---|
| $$UPDATE | Updates the following existing network map node entry. Node entry updates are performed as a replacement at the keyword level. Therefore, updates of list-type keywords, like APPLIDS, require that you specify the entire list.<br>Do not use $$UPDATE to update network map node entries which already contain the LDNS parameter. Use $$DELETE and $$ INSERT to change nodes which contain an LDNS parameter.<br>**Note:** You can use $$UPDATE to add the LDNS parameter to a node which did not previously contain one. |
| $$DELETE | Deletes the following existing network map node entry. |
| $$SYNTAX and $$ENDSYNTAX | Performs a syntax check of the network map control statements following this verb. |
| $$VERIFY and $$ENDVERIFY | Verifies that the node definitions 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. |

## Updating the Netmap through the Batch Interface

To issue the UPDATE NETMAP command through the Connect:Direct batch utility:

1. Change the network map source using the $$ACTION verbs.
2. Place the UPDATE NETMAP commands in the DMBATCH job stream.
3. Ensure that Connect:Direct is running.
4. Submit the job.
5. Correct any errors identified on the activity report and resubmit if necessary.
6. Verify the results.

## Updating the Netmap through the IUI Interface

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, consider batch processing.

1. To change or create a new member with your updates, change the network map source to use 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:
 ==> ________________________________________________

 ENTER MEMBER NAME:     (OPTIONAL)
 ==> _________

 ENTER DDNAME FOR LOG FILE OR "NONE":     (DEFAULTS TO "CDLOG",
 ==> _________                             FIRST TWO CHARACTERS MUST BE "CD")


              OUTPUT DESTINATION ==> DIS   (DIS-DISPLAY,PRT-PRINT)
```

4. Type the network map source file name and the appropriate optional parameters and press **ENTER**. Unless you select the PRINT option on the Update NETMAP screen, the report routes to your terminal.
5. Verify the results on the activity report.
6. Correct any errors identified on the activity report and re-type if necessary.

## $$ACTION Verb Examples

The following are examples of updating the network map through the use of 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.

```
= = > * * *   START NETMAP UPDATE         * * *
= = >    DATE:  02/27/2003 TIME=14:59:26
= = >    SMUPNLGI NETLOG=NONE REQUIRED, LOGGING INACTIVE
============================================================
*INSERT THE FOLLOWING NODE DEFINITION
$$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
============================================================
```

The first message, SMUPNLGI, 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.

```
= = > * * *  START NETMAP UPDATE      * * *
= = >   DATE:  02/27/2003 TIME=14:59:26
= = >   SMUPNLGI LOGGING ACTIVE - LOG DDNAME=CDLOG
============================================================

*UPDATE THE FOLLOWING NODE DEFINITION ADDING RAPPL2
*CHANGING MAXIMUM PARALLEL SESSIONS TO FOUR (4).
$$UPDATE
  ADJACENT.NODE=(( CD.NODE2  APPLID2 )   -
           PARSESS=(4,2)                 -
           APPLIDS=(RAPPL1 RAPPL2))

= = >    SMUP032I APPLIDS RECORD UPDATED
= = >    SMUP034I ADJACENT.NODE RECORD UPDATED
= = >    SMUP008I REQUEST SUCCESSFUL FOR NODE=CD.NODE2
============================================================
```

The first message, SMUPNLGI, shows that logging is requested and that a record of the transaction is recorded in CDLOG.  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.

```
= = > * * *   START NETMAP UPDATE      * * *
= = >     DATE:  02/27/2003 TIME=15:09:36
= = >     SMUPNLGI NETLOG=NONE REQUIRED, LOGGING INACTIVE
================================================================
$$DELETE
     ADJACENT.NODE=((CD.NODE2 APPLID2) -
               PARSESS=(4,2)           -
              APPLIDS=(RAPPL1 RAPPL2 ))
= = >    SMUP032I APPLIDS RECORD DELETED
= = >    SMUP034I ADJACENT.NODE RECORD DELETED
= = >    SMUP008I REQUEST SUCCESSFUL FOR NODE=CD.NODE1
================================================================
```

The first message, SMUPNLGI, indicates that logging is not requested, so Connect:Direct does not keep a record of the transaction. The last messages indicate that the APPLIDs and adjacent node records are successfully deleted.

### $$SYNTAX Example

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. The messages are numbered in the example for clarification; they are not numbered on the actual output.

```
           = = > * * *         START NETMAP UPDATE            * * *
           = = > DATE:  02/27/2003 TIME=13:49:16
(1)        = = > SMUPNLGI NETLOG=NONE REQUIRED, LOGGING INACTIVE
=============================================================================
           $$SYNTAX
(2)        = = > SMUP011I 'SYNTAX   ' ACTION STARTED
=============================================================================
           LOCAL.NODE=(( CD.NODE1  APPLID1  ,,  SUPERUSR) -
                       TCQ=( TCQ TCX  ))
(3)        = = > SMUP005I  LOCAL.NODE RECORD PROCESSING NOT ALLOWED
                    BYPASSED
=============================================================================
           ADJACENT.NODE=(( CD.NODE1  APPLID1  ) -
              PARSESS=(5,2)  -
            APPLIDS=(LAPPL1  LAPPL2  LAPPL3))
(4)        = = > SMUP008I REQUEST SUCCESSFUL FOR NODE=CD.NODE1
=============================================================================
          ADJACENT.NODE=(( CD.NODE2  APPLID2 ) -
             PARSESS=(5,2)  -
          APPLIDS=(RAPPL1))
(4)       = = > SMUP008I REQUEST SUCCESSFUL FOR NODE=CD.NODE2
=============================================================================
          $$ENDSYNTAX
(5)       = = > SMUP012I 'SYNTAX   ' ACTION STOPPED
=============================================================================
```

The numbers in parentheses indicate:

1. Logging is not requested; therefore, no 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 output follows. The messages are numbered in the example for clarification; they are not numbered on the actual output.

```
           = = > * * *      START NETMAP UPDATE     * * *
           = = >  DATE:  02/27/2003 TIME=15:35:16
(1)        = = >  SMUPNLGI NETLOG=NONE REQUIRED, LOGGING INACTIVE
=======================================================================
           $$VERIFY

(2)        = = >  SMUP011I 'VERIFY   ' ACTION STARTED
=======================================================================

           ADJACENT.NODE=(( CD.NODE2  APPLID2 ) -
              PARSESS=(5,2) -
             APPLIDS=(RAPPL1 ))
(3)        = = > SMUP092I  APPLIDS RECORD DID NOT MATCH
           = = > SMUP094I ADJACENT.NODE RECORD DID NOT MATCH
           = = > SMUP096I RECORDS DO NOT MATCH - VERIFICATION FAILED
                       FOR NODE
                 =CD.NODE2
=======================================================================

           $$ENDVERIFY

(4)        = = > SMUP012I 'VERIFY   ' ACTION STOPPED
=======================================================================
```

The number in parentheses indicate:

1. Logging is not requested; therefore, no record is 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. Verification is complete.

# Viewing the Network Map

You can view the network map online through the IUI interface or view the contents of the network map by unloading the network map VSAM file source.

## Viewing the Netmap through the IUI Interface

To view the network map using the Connect:Direct IUI:

1. Select option **NM** from the Primary Options Menu to display the SELECT NETPMAP OR TCP INFORMATION screen. For more information, see *The Network Map* in *Connect:Direct for z/OS User's Guide.*

## Unloading the Network Map to the Source Format

Connect:Direct provides a utility to unload the network map to its source format. You can then view the source format to see network map settings. This utility is provided in the JNETUNLD member in the sample library.

To unload the network map, submit the JNETUNLD member. You can unload the network map while Connect:Direct is running.

An example of the JCL follows.

```
//STEP      EXEC  PGM=DMCNTMPL,PARM='UNLOAD'
//NETMAP DD       DSN=NETMAP.DATASET,DISP=SHR
//UNLOAD DD       DSN=NETMAP.UNLOAD,DISP=(NEW,CATLG),
//              DCB=(DSORG=PS,RECFM=FB,BLKSIZE=0,LRECL=80),
//              UNIT=SYSDA,SPACE=(TRK,(4,2,))
```

The network map source is unloaded to the data set specified in the JCL.

# Chapter 7

# Configuring a Connect:Direct/Plex Environment

This chapter describes how to configure a Connect:Direct/Plex environment. It contains the following sections:

- Connect:Direct/Plex Overview
- Setting Up a New Connect:Direct/Plex Environment
- Advanced Configuration Considerations
- Converting an Existing Connect:Direct Stand-Alone Server to a Connect:Direct/Plex Environment
- Converting Two Existing Connect:Direct Stand-Alone Server Systems to a Connect:Direct/Plex Environment
- Additional Configuration Examples

## Connect:Direct/Plex Overview

As explained in Chapter 1, *About Connect:Direct for z/OS*, Connect:Direct runs in two configurations:

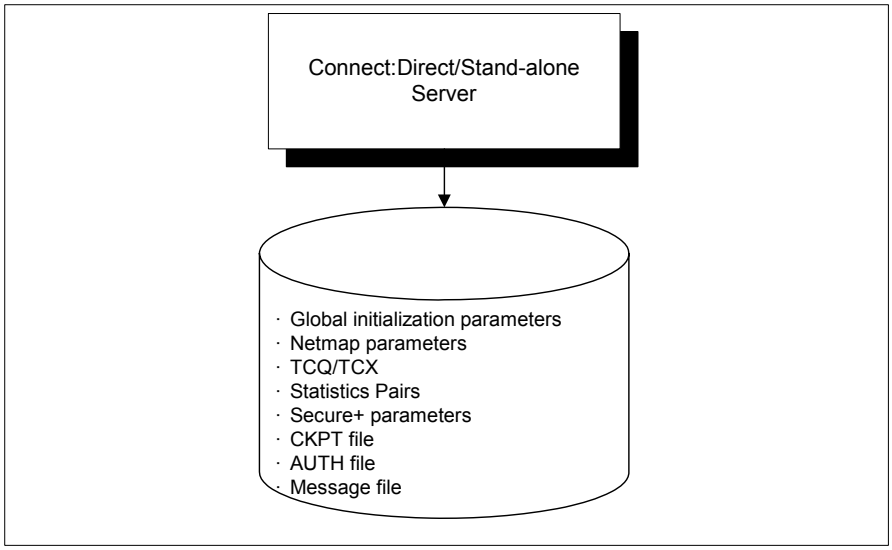
- Connect:Direct Stand-alone Server, which is a single Connect:Direct system operating within an IBM z/OS environment.
- Connect:Direct/Plex, which is a Connect:Direct system operating in an IBM z/OS sysplex or parallel sysplex environment consisting of a Connect:Direct Manager and one or more Connect:Direct Servers.

A Connect:Direct Stand-alone Server and a Connect:Direct/Plex environment have the following configuration differences:

- Initialization parameters

  The two sets of initialization parameters in Connect:Direct are global and local.

A Connect:Direct Stand-alone Server uses only global initialization parameters to set system-wide values, as shown in the following illustration.




A Connect:Direct/Plex environment uses both global and local initialization parameters. Global initialization parameters apply to each member of the Connect:Direct/Plex environment. Local initialization parameters apply to specific members of the Connect:Direct/Plex environment and can override some global initialization parameters affecting that member.

Global initialization parameters are stored in a file shared by all Connect:Direct/Plex members. In the EXEC statement (Connect:Direct Stand-alone Server and Connect:Direct/Plex), the PARM= keyword specifies the name and location of the global initialization parameters file.

The local initialization parameters of each Connect:Direct/Plex member are stored in a unique PDS member. The location of the local initialization parameters file is specified by the //CDPLEX DD in the startup JCL of each member.

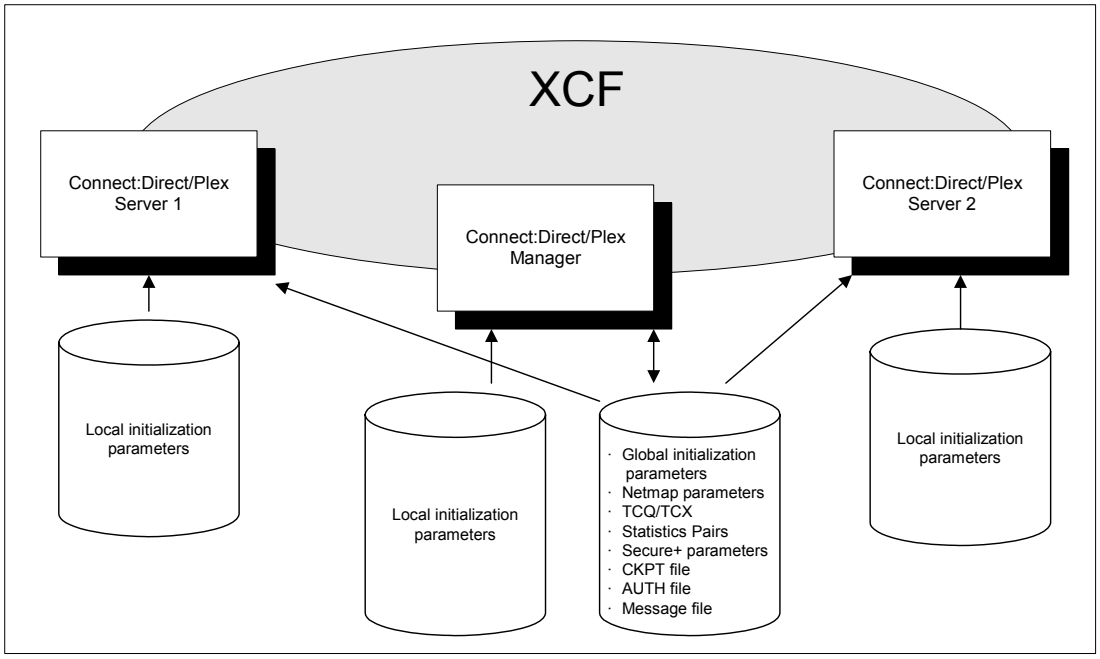
---

**Note:** In a Connect:Direct/Plex environment, you can override only the initialization parameters allowed in the *local* initialization parameters file by using the PARM= keyword in the EXEC statement at system startup.

In a Connect:Direct Stand-alone Server environment, however, you can override *global* initialization parameters with the PARM= keyword in the EXEC statement.

---

The following illustration shows how global and local initialization parameters are used in a Connect:Direct/Plex environment.




- VTAM APPLIDs

  A Connect:Direct Stand-alone Server obtains its VTAM APPLIDs from the network map.

  In a Connect:Direct/Plex environment, the Connect:Direct/Manager obtains its VTAM APPLIDs from the network map, but each Connect:Direct Server obtains its VTAM APPLIDs from local initialization parameters file.

- TCP/IP addresses and ports

  A Connect:Direct Stand-alone Server obtains its TCP/IP listen ports from the global initialization parameters file.

  In a Connect:Direct/Plex environment, the Connect:Direct Manager and Connect:Direct Servers obtain their TCP/IP addresses and listen port numbers from their local initialization parameters files. However, if the TCP port number is not specified in the local initialization parameters file of the Connect:Direct Manager, the Connect:Direct Manager obtains its listen port number from the global initialization parameters file.

  Each server overrides the global initialization parameters by specifying those parameters in that server's local initialization parameters. The first address defined in the parameter becomes the local or default address. For more information about defining TCP/IP listening tasks, see *TCP/IP Port Number* on page 144.

  The CDPLEX.REDIRECT local initialization parameter is used by the Connect:Direct/Plex Manager in the Connect:Direct/Plex environment to determine the redirection address that is presented to the remote node. This parameter allows you to specify redirection addresses based on the security node type (internal or external) and session type (TCP/IP or UDT) of the adjacent node in the network map.

When an address is specified, an internal address and external address is defined and each can have a specified redirection port defined. When the adjacent node entry is defined with the INT flag, the appropriate internal address is returned. Conversely, when the EXT flag is defined, the appropriate external address is returned.

Up to eight different addresses or ports can be defined for each server. However, in the Connect:Direct/Plex server that defines CDPLEX.REDIRECT only two are effectively used when Process redirection occurs. To use the additional port in the Connect:Direct/Plex servers, those servers must be contacted directly by the remote node.

**Note:** A special consideration exists if the Connect:Direct/Plex Manager is initialized on a system that is not IPv6 enabled, and one or more of the servers supports IPv6. When the Connect:Direct/Plex is the SNODE, the Connect:Direct/Plex Manager accepts connection requests for IPv4 only. However, if the Connect:Direct/Plex is the PNODE, the Connect:Direct/Plex Manager can assign outbound processes to a Connect:Direct Server that supports IPv6.

- System files

  In a Connect:Direct Stand-alone Server, the system files (network map, Statistics Pairs, CKPT, AUTH, Message, TCQ and TCX files) are stored in one location and apply to the entire DTF. If two Connect:Direct Stand-alone Server systems operate in a sysplex environment, each system must have its own system files.

  In a Connect:Direct/Plex environment, the system files are also stored in one location and are shared by all Connect:Direct/Plex members, as in the illustration on page 169. Only one set of system files is needed for all Connect:Direct/Plex members.

**Note:** In a Connect:Direct/Plex environment, the LU6.2 connection protocol does not enable the Connect:Direct Manager to redirect work to one of its servers. The remote node must address the server on which you want to run an LU6.2 Process. To do this, specify the node name and VTAM address of the Connect:Direct Server on which the Process is to run in the remote server's network map. Use the same CDPLEX.SERVER.NODE and CDPLEX.VTAM specified for the local initialization parameters for the Connect:Direct Server you are trying to address.

# Setting Up a New Connect:Direct/Plex Environment

This section describes a Connect:Direct/Plex setup for a new installation.

## Assumptions

The setup example in this section assumes the following:

- You have installed and brought up a single operational Connect:Direct DTF.

  To accomplish this task, refer to the *Connect:Direct for z/OS Installation Guide* and Chapter 6, *Maintaining the Network Map* of this guide.

- ✦ You are changing a Connect:Direct Stand-alone Server into a Connect:Direct/Plex environment with three members: the Connect:Direct Manager and two Connect:Direct Servers. The Connect:Direct Servers are named SERVER1 and SERVER2. SERVER1 has tape drive access for copy Processes requiring tapes.
- ✦ The global and local initialization parameters files are located in $CD.PLEX.INITPARM. The JCL to bring up the Connect:Direct/Plex environment is located in $CD.PLEX.JCL. You can either allocate these data sets or use existing data sets in their place.

## Connect:Direct/Plex Setup

To set up a Connect:Direct/Plex environment:

1. Copy your current initialization parameters file into $CD.PLEX.INITPARM as member CDPLX.
2. Copy your current Connect:Direct Stand-alone Server startup JCL into $CD.PLEX.JCL as member CDMGR.
3. Add the following initialization parameters to the CDPLX member in $CD.PLEX.INITPARM. This member becomes the Connect:Direct/Plex global initialization parameters file.

```
CDPLEX=YES
XCF.NAME=8-character-name
```

   The CDPLEX=YES parameter indicates a Connect:Direct/Plex environment. It also directs the DTF to read its local initialization parameters from the file specified in the //CDPLEX DD statement in the startup JCL.

   The XCF.NAME parameter specifies a unique name used by the z/OS Cross Systems Communication Facility (XCF) to assist communications among Connect:Direct/Plex members. This name indicates that the Connect:Direct Manager and Connect:Direct Servers are part of the same XCF group.

4. You can add the following optional parameters to the CDPLX member in $CD.PLEX.INITPARM. For more information on these parameters, see Appendix B, *Local Initialization Parameters*.
   - ◆ CDPLEX.TIMER specifies the time-out value for XCF communications in minutes.
   - ◆ CDPLEX.WLM.GOAL specifies whether IBM Workload Manager (WLM) Goal Mode queries are made. This parameter is optional.

5. Create the local initialization parameters files for each Connect:Direct/Plex member:
   a. Copy the MANAGER, SERVER1, and SERVER2 sample local initialization parameters members from the Connect:Direct installation PARMLIB into $CD.PLEX.INITPARM.
   b. Change the CDPLEX.TCPIP parameter of the MANAGER member to specify the TCP/IP stack address used by the Connect:Direct Manager.

      You do not need to change any other parameters in the MANAGER member.

```
CDPLEX.MANAGER=YES
CDPLEX.TCPIP=nnn.nnn.nnn.nnn
CDPLEX.SERVER.JOBDSN=$CD.PLEX.JCL
CDPLEX.SERVER.JOBMEM=((CDSRV1,SERVER1), -
                      (CDSRV2,SERVER2))
```

   c. Change the CDPLEX.VTAM parameter in the SERVER1 member as follows:
      - Replace the applid11 value with the VTAM APPLID used by this Connect:Direct Server for SNA copy Processes.
      - Replace the applid12 value with the PNODE-SNODE APPLID.
      - Replace the applid13 value with the APPLID for the Programmed Operator Application.

      These APPLIDs must be unique across the Connect:Direct/Plex environment and cannot be the same as those specified in the network map.

```
CDPLEX.MANAGER=NO
CDPLEX.SERVER=SERVER1
CDPLEX.VTAM=(applid11,applid12,applid13)
CDPLEX.PLEXCLASSES=(TAPE,*)
TCP.LISTEN=nnn.nnn.nnn.nnn,port
UDT33.LISTEN=nnn.nnn.nnn.nnn,port
```

   d. The TCP.LISTEN and UDT33.LISTEN initialization parameters specify the TCP and UDT listen address and port number combinations. Use a different listen port number than the one used in the existing initialization parameters file.

> **Note:** The CDPLEX.PLEXCLASSES parameter in SERVER1 specifies a ‘TAPE’ PLEXCLASS. For Processes that require tape drives, specify the ‘TAPE’ PLEXCLASS in their Process definitions. These Processes run on SERVER1. (See the chapters about building Processes and controlling Processes in the TCQ in *Connect:Direct for z/OS User’s Guide* for more information on using PLEXCLASS in a Process.)

   e. Change the CDPLEX.VTAM parameter in the SERVER2 member as follows:
      - Replace the applid21 value with the VTAM APPLID used by this Connect:Direct Server for SNA copy Processes.

- Replace the applid22 value with the PNODE-SNODE APPLID.
- Replace the applid23 value with the APPLID for the Programmed Operator Application.

These APPLIDs must be unique across the Connect:Direct/Plex environment and cannot be the same as those specified in the network map.

```
CDPLEX.MANAGER=NO
CDPLEX.SERVER=SERVER2
CDPLEX.VTAM=(applid21,applid22,applid23)
CDPLEX.PLEXCLASSES=(*)
TCP.LISTEN=nnn.nnn.nnn.nnn,port
UDT33.LISTEN=nnn.nnn.nnn.nnn,port
```

f. The TCP.LISTEN and UDT33.LISTEN initialization parameters specify the TCP and UDT listen address and port number combinations. Use a different listen port number than the one used in the existing initialization parameters file.

6. Add the CDPLEX DD statement in the following example to the CDMGR member in $CD.PLEX.JCL. (This JCL is the startup JCL copied in step ).

   This statement directs the startup JCL to the global initialization parameters file.

```
//DTF EXEC DMINIT,
// PARM='$CD.PLEX.INITPARM(CDPLX)'
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(MANAGER)
```

7. Copy the CDMGR member to CDSRV1.
8. Make the following changes to the CDSRV1 member:
   a. Change the job name so that this job can run simultaneously with the CDMGR JCL.
   b. Change the member name in the CDPLEX DD statement to SERVER1, as follows. This change directs the CDSRV1 JCL to its local initialization parameters file.

```
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(SERVER1)
```

9. Copy the modified CDSRV1 member to CDSRV2.
10. Make the following changes to the CDSRV2 member:
    a. Change the job name so that this job can run simultaneously with the CDMGR JCL and CDSRV1 JCL.
    b. Change the member name in the CDPLEX DD statement to SERVER2, as follows. This change directs the CDSRV2 JCL to its local initialization parameters file.

```
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(SERVER2)
```

**Note:** You can route jobs to a different z/OS image by specifying the local node name of the other system in an XEQ statement in the Connect:Direct Manager or Connect:Direct Server startup JCL, as follows:

```
/*XEQ njenode
```

This example routes the job to the z/OS image identified by the local node name NJENODE.

11. Submit the CDMGR JCL to bring up the Connect:Direct/Plex server.

    After the Connect:Direct Manager initializes, it submits the CDSRV1 JCL and CDSRV2 JCL to bring up the two Connect:Direct Servers.

12. After the Connect:Direct Manager initializes, use the IUI to signon to the Connect:Direct Manager.

    You can then submit Processes and perform other functions through the IUI.

# Advanced Configuration Considerations

The previous section describes a basic Connect:Direct/Plex setup. However, some installations require more complex configurations. Examples of complex configurations include:

- Converting an existing Connect:Direct system to a Connect:Direct/Plex environment (on page 178)
- Merging multiple existing Connect:Direct systems into a Connect:Direct/Plex environment (on page 182)

Before attempting a complex configuration, be aware of the following issues.

## Connect:Direct/Plex System File Considerations

All Connect:Direct:/Plex members share a single set of Connect:Direct system files. If you combine multiple existing Connect:Direct systems into one Connect:Direct/Plex environment, you may need to merge some Connect:Direct system files from the individual systems.

Do not merge system files that are listed in the following table.

| File | Comment |
|---|---|
| CKPT file | You cannot merge the CKPT files from multiple Connect:Direct images. You must either:<br>◆ Define a new CKPT file using the PLEXCKPD JCL in the SAMPLIB, or<br>◆ Use the CKPT file from one of the existing Connect:Direct systems. However, the existing CKPT file size may not be sufficient for a Connect:Direct/Plex environment. |
| Message file | You can use any existing Connect:Direct Message file. You do not need to combine Message files from the individual Connect:Direct systems. |

| File | Comment |
|---|---|
| Statistics files | You cannot merge statistics files from multiple Connect:Direct images. You can reference them using a Connect:Direct/Plex environment as archived statistic files. You can create new statistics file pairs by using the PLEXSTAD JCL in the SAMPLIB. |
| TCQ and TCX files | You cannot merge TCQ and TCX files from multiple Connect:Direct images. You must either:<br>◆ Define new TCQ and TCX files using the PLEXTCQD JCL in the SAMPLIB, or<br>◆ Use the TCQ and TCX files from one of the existing Connect:Direct systems. However, the existing TCQ and TCX file sizes may not be sufficient for a Connect:Direct/Plex environment. |

The following table lists system files that you need to merge.

| File | Comment |
|---|---|
| AUTH file | If unique entries exist in the existing Connect:Direct systems' AUTH files:<br>1 Define a new AUTH file using the PLEXAUTD JCL found in the SAMPLIB.<br>2 Copy the existing AUTH files into the new AUTH file using the PLEXAUTC JCL found in the SAMPLIB.<br>PLEXAUTC is an IDCAMS REPRO that specifies NOREPLACE. If any duplicate records exist, only the first one is saved.<br>If unique entries do not exist, use one of the existing AUTH files. |
| NETMAP file | You need to create a new network map source file.<br>The new network map source file uses information from the existing systems' network maps. (If the existing Connect:Direct systems' network map source is not available, create the source files by performing network map unloads for the existing systems' network map files. See *Unloading the Network Map to the Source Format* on page 165 for more information.)<br>To create a new network map source file:<br>1 Copy the network map source from an existing Connect:Direct system as NETMAPLX.<br>2 Copy the remote definitions from all other existing network map source files into NETMAPLX.<br>3 Remove all duplicate entries.<br>4 Define the new network map file using the PLEXNETD JCL found in the SAMPLIB.<br>5 Load the new network map file using the PLEXNETL JCL found in the SAMPLIB.<br>6 Check the output from network map load and correct any errors.<br>7 Rerun the network map load if necessary. |
| TYPE file | If unique entries exist in the existing Connect:Direct systems' TYPE files:<br>1 Define a new TYPE file using the PLEXTYPD JCL found in the SAMPLIB.<br>2 Copy the existing TYPE files into the new TYPE file using the PLEXTYPC JCL found in the SAMPLIB.<br>The PLEXTYPC is an IDCAMS REPRO that specifies NOREPLACE. If any duplicate records exist, only the first one is saved.<br>If unique entries do not exist, use one of the existing TYPE files. |

## Local Node Naming Considerations

The network map contains the local node name for the Connect:Direct/Plex environment. The node name used in the network map varies according to the type of configuration:

- Installing a new Connect:Direct/Plex environment

  If you are installing a new Connect:Direct/Plex environment, you must create a new local node name. You must provide the new node name, along with the APPLID and/or TCP/IP address and port number to all Connect:Direct partner nodes. The partner Nodes must provide you the same information for use in your local network map.

- Replacing an existing Connect:Direct system with a Connect:Direct/Plex environment

  If you are replacing an existing Connect:Direct system with a Connect:Direct/Plex environment, use the existing system node name as the Connect:Direct/Plex local node name, you must provide new APPLIDs for the Connect:Direct Manager. One Connect:Direct Server uses the existing APPLIDs in its local initialization parameters. Any additional servers require new APPLIDs.

  Refer to the setup procedure on page 178, for more details.

- Replacing multiple existing Connect:Direct systems with a Connect:Direct/Plex environment

  If you are replacing multiple Connect:Direct systems with a Connect:Direct/Plex environment, create a new local node name for the Connect:Direct/Plex. Use the existing node names in the CDPLEX.SERVER.NODE initialization parameter of the Connect:Direct Server. All adjacent node entries in the network map must include USE.SERVER.NODE=YES. You must provide new APPLIDs for the Connect:Direct Manager. Each Connect:Direct Server uses the existing APPLID from its corresponding Connect:Direct Stand-alone Server image in its local initialization parameters.

  Refer to the setup procedure on page 182, for more details.

## Strategies for Communicating with Non-Plex Servers

A Connect:Direct/Plex environment can perform workload balancing among the Connect:Direct Servers. However, if the Connect:Direct/Plex environment communicates with an external non-Connect:Direct/Plex system, the other Connect:Direct system may have problems with Processes from the same Connect:Direct adjacent node, but with a different VTAM APPLID or TCP/IP address than specified in their network map.

Connect:Direct/Plex offers three ways of avoiding this problem.

### Use Alternate Communication Paths to Define the Connect:Direct/Plex

To define a Connect:Direct/Plex that has servers running on several hosts, you can use the ALT.COMM parameter in the network map of each non-Plex Server that will communicate with the Connect:Direct/Plex. To define the Connect:Direct/Plex:

1. Specify USE.SERVER.NODE=NO in the Connect:Direct/Plex network map entry of each non-Plex Server so that all servers in the Connect:Direct/Plex environment appear as one node.

2. Define the Connect:Direct/Plex node as an adjacent node with all possible IP addresses of the hosts that the Manager can run on specified using the ALT.COMM definition.

   The following is an example of the ALT.COMM parameter:

```
ADJACENT.NODE=(  -
    (CDMGR,1366,10.1.1.1,TCP,EXT,BOTH) -
    ENVIRONMENT=ZOS -
    PARSESS=(00000010 00000002) -
    ALT.COMM=(ALT.DIR=TOP               -
              (ALT.ADDR=10.1.1.2,ALT.PORT=1366,-
               ALT.TYPE=TCP ,  ALT.USE.OUT=NO )
              (ALT.ADDR=10.1.1.3,ALT.PORT=1366,-
               ALT.TYPE=TCP ,  ALT.USE.OUT=NO )
              (ALT.ADDR=10.1.1.4,ALT.PORT=1366,-
               ALT.TYPE=TCP ,  ALT.USE.OUT=NO )) -
```

3. Copy this ALT.COMM definition and put it in the network map of each non-Plex Server that will communicate with the Connect:Direct/Plex.

The advantages to this approach are:

- ✦ You only have to define one entity, the Connect:Direct/Plex, and then copy that same definition to the network maps of the non-Plex Servers.
- ✦ You can still use the NETMAP-checking feature.

### Force All Processes to One Connect:Direct Server

To direct all Processes between the Connect:Direct/Plex environment and the external Connect:Direct to one Connect:Direct Server:

1. Specify a default PLEXCLASS parameter in the Connect:Direct/Plex network map adjacent node entry for the external Connect:Direct system.
2. Specify that PLEXCLASS parameter in only one local initialization parameter of the Connect:Direct Server.
3. Specify the VTAM APPLID or TCP/IP address of the specific Connect:Direct Server all Processes are being forced to in the network map of the external non-Plex Connect:Direct system.

The disadvantage of this approach is that it does not take advantage of Connect:Direct/Plex workload balancing.

### Define a Unique Node Name for Each Connect:Direct Server

By defining a node name for each Connect:Direct Server, the Connect:Direct/Plex environment can initiate Processes to the external Connect:Direct system through any available Connect:Direct/Plex server. For each Connect:Direct server, you must define a USE.SERVER.NODE network map parameter and a CDPLEX.SERVER.NODE initialization parameter.

To avoid making your Connect:Direct/Plex configuration more complex than necessary, use the USE.SERVER.NODE and CDPLEX.SERVER.NODE parameters only if your system meets all of the following conditions:

- The external Connect:Direct system must connect to two or more Connect:Direct Servers in the Connect:Direct/Plex environment.
- The external Connect:Direct system uses network map checking.
- The external Connect:Direct system has non-Plex servers, which cannot communicate directly with the Connect:Direct Manager.

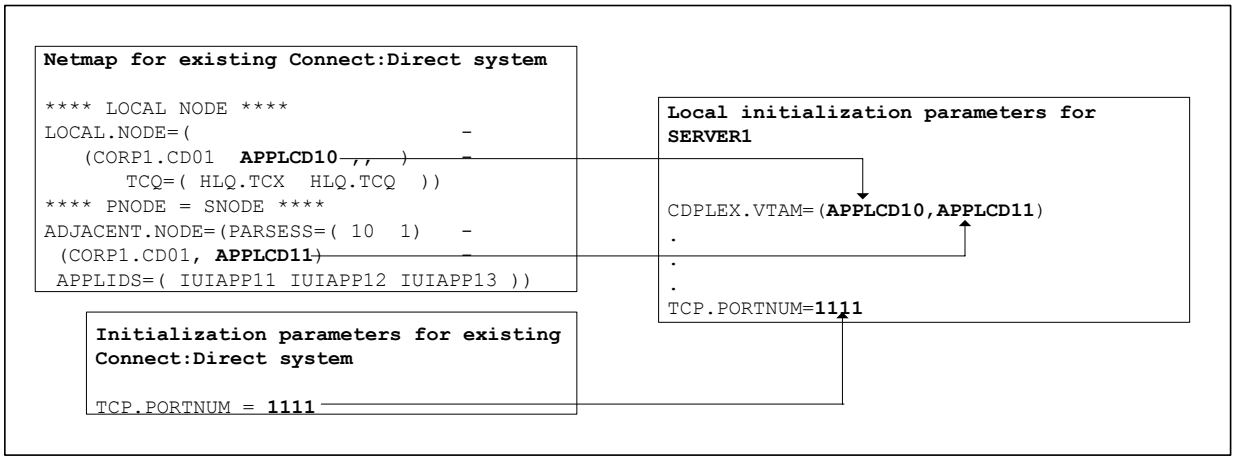
The disadvantages of this approach are:

- You must manually direct Processes initiated by the external Connect:Direct system to each Connect:Direct Server.
- You may need to create additional network map entries for remote systems.

# Converting an Existing Connect:Direct Stand-Alone Server to a Connect:Direct/Plex Environment

This section describes how to convert an existing production Connect:Direct Stand-alone Server into a Connect:Direct/Plex environment with two servers. This configuration takes advantage of the Connect:Direct/Plex single image and workload balancing capability for Processes initiated by this Connect:Direct/Plex environment. This configuration also supports external Connect:Direct systems without requiring any changes to the external systems.

The following illustration shows how the network map and initialization parameter values from the existing stand-alone Connect:Direct system map to the new Connect:Direct/Plex environment for this configuration.




## Assumptions

The setup example in this section assumes the following:

- You are currently running a production Connect:Direct Stand-alone Server.

- The global and local initialization parameter files for the Connect:Direct/Plex environment are located in $CD.PLEX.INITPARM. The JCL to bring up the Connect:Direct/Plex environment is located in $CD.PLEX.JCL.

  You can either allocate these data sets or use existing data sets in their place.
- The Connect:Direct/Plex environment identifies itself to external systems with the same node name as the production Connect:Direct Stand-alone Server.

## Connect:Direct/Plex Setup

To convert an existing production Connect:Direct Stand-alone Server into a Connect:Direct/Plex environment:

1. Copy the existing initialization parameters file into $CD.PLEX.INITPARM as member CDPLX.
2. Copy the existing Connect:Direct Stand-alone Server startup JCL into $CD.PLEX.JCL as member CDMGR.
3. Change the network map source to specify new APPLIDs for the LOCAL.NODE and the PNODE/SNODE ADJACENT.NODE.

   The existing APPLIDs are used for the Connect:Direct SERVER1, which means that you do not need to change the connections to external Connect:Direct systems.
4. Add the following initialization parameters to the CDPLX member in $CD.PLEX.INITPARM. This member becomes the Connect:Direct/Plex global initialization parameters file.

   ```
   CDPLEX=YES
   XCF.NAME=8-char-name
   ```

   The CDPLEX=YES parameter indicates a Connect:Direct/Plex environment. It also directs the DTF to read its local initialization parameters from the file specified in the //CDPLEX DD statement in the startup JCL.

   The XCF.NAME parameter specifies a unique name used by the z/OS Cross Systems Communication Facility (XCF) to assist communications among Connect:Direct/Plex members. This name indicates that the Connect:Direct Manager and Connect:Direct Servers are part of the same XCF group.
5. You can add the following optional parameters to the CDPLX member in $CD.PLEX.INITPARM. For more information on these parameters, see Appendix B, *Local Initialization Parameters*.

   - CDPLEX.TIMER specifies the time-out value for XCF communications in minutes.
   - CDPLEX.WLM.GOAL specifies whether IBM Workload Manager (WLM) Goal Mode queries are made. This parameter is optional.

6. Create the local initialization parameter files for each Connect:Direct/Plex member:
   a. Copy the MANAGER, SERVER1, and SERVER2 sample local initialization parameters members from the Connect:Direct $cd.PARMLIB library into $CD.PLEX.PARMLIB.
   b. Change the TCP.LISTEN parameter (following in bold) to specify the TCP/IP stack address that is used by the Connect:Direct Manager.

      You do not need to change any other parameters in the MANAGER member.

```
CDPLEX.MANAGER=YES
TCP.LISTEN=nnn.nnn.nnn.nnn,port
UDT33.LISTEN=nnn.nnn.nnn.nnn,port
CDPLEX.SERVER.JOBDSN=$CD.PLEX.JCL
CDPLEX.SERVER.JOBMEM=((CDSRV1,SERVER1), -
                      (CDSRV2,SERVER2))
```

   c. Change the CDPLEX.VTAM parameter in the SERVER1 member as follows:
      - Replace the applid11 value with the VTAM APPLID that is defined in the existing network map for SNA copy Processes.
      - Replace the applid12 value with the VTAM APPLID that is defined in the existing network map for the PNODE-SNODE APPLID.

      These APPLIDs must be unique across the Connect:Direct/Plex environment and cannot be the same as specified in the new network map.

```
CDPLEX.MANAGER=NO
CDPLEX.SERVER=SERVER1
CDPLEX.VTAM=(applid11,applid12)
CDPLEX.PLEXCLASSES=(TAPE,*)
TCP.LISTEN=nnn.nnn.nnn.nnn,port
UDT33.LISTEN=nnn.nnn.nnn.nnn,port
```

   d. The TCP.LISTEN and UDT33.LISTEN initialization parameters specify the TCP and UDT listen address and port number combinations. Use a different listen port number than the one used in the existing initialization parameters file.

> **Note:** The CDPLEX.PLEXCLASSES parameter in SERVER1 specifies a 'TAPE' PLEXCLASS. For Processes that require tape drives, specify the 'TAPE' PLEXCLASS in their Process definitions. These Processes run on SERVER1. (See the *Building, Modifying, and Submitting Processes* chapter of the *Connect:Direct for z/OS User's Guide* for more information on using PLEXCLASS in a Process.)

   e. Change the CDPLEX.VTAM parameter in the SERVER2 member as follows:
      - Replace the applid21 value with a new VTAM APPLID you have defined for SNA copy Processes.

- Replace the applid22 value with a new VTAM APPLID you have defined for the PNODE-SNODE APPLID.

These APPLIDs must be unique across the Connect:Direct/Plex environment and cannot be the same as specified in the new network map or used for SERVER1.

```
CDPLEX.MANAGER=NO
CDPLEX.SERVER=SERVER1
CDPLEX.VTAM=(applid21,applid22)
CDPLEX.PLEXCLASSES=(*)
TCP.LISTEN=nnn.nnn.nnn.nnn,port
UDT33.LISTEN=nnn.nnn.nnn.nnn,port
```

f. The TCP.LISTEN and UDT33.LISTEN initialization parameters specify the TCP and UDT listen address and port number combinations. Use a different listen port number than the one used in the existing initialization parameters file.

7. Add the CDPLEX DD statement in the following example to the CDMGR member in $CD.PLEX.JCL (this JCL is the startup JCL copied in step 2).

```
//DTF EXEC DMINIT,
// PARM='$CD.PLEX.INITPARM(CDPLX)'
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(MANAGER)
```

8. Copy the CDMGR member to CDSRV1.
9. Make the following changes to the CDSRV1 member:
   a. Change the job name so that this job can run simultaneously with the CDMGR JCL.
   b. Change the member name in the CDPLEX DD statement to SERVER1, as follows. This change directs the CDSRV1 JCL to its local initialization parameters file.

```
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(SERVER1)
```

10. Copy the modified CDSRV1 member to CDSRV2.
11. Make the following changes to the CDSRV2 member:
    a. Change the job name so that this job can run simultaneously with the CDMGR JCL and CDSRV1 JCL.
    b. Change the member name in the CDPLEX DD statement to SERVER2, as follows. This change directs the CDSRV2 JCL to its local initialization parameters file.

```
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(SERVER2)
```

**Note:** You can route jobs to a different z/OS image by specifying the local node name of the other system name in an XEQ statement in the Connect:Direct Manager or Connect:Direct Server JCL as follows:

```
/*XEQ njenode
```

This example routes the job to the z/OS image identified by the local node name NJENODE.

12. Submit the CDMGR JCL to bring up Connect:Direct/Plex.

    After the Connect:Direct Manager initializes, it submits the CDSRV1 JCL and CDSRV2 JCL to bring up the two Connect:Direct Servers.

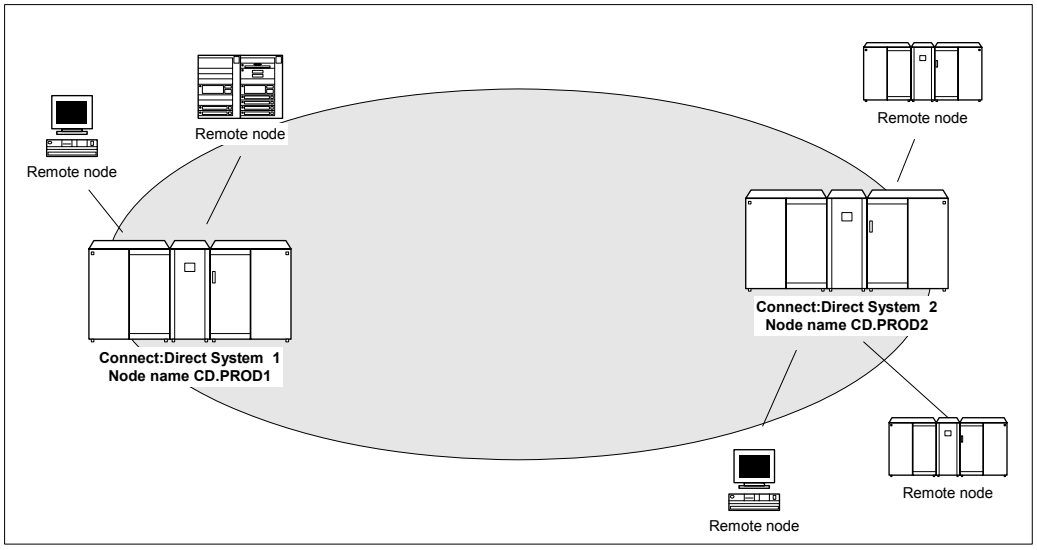
13. After the Connect:Direct Manager initializes, use the IUI to signon to the Connect:Direct Manager.

    You can then submit Processes and perform other functions through the IUI.

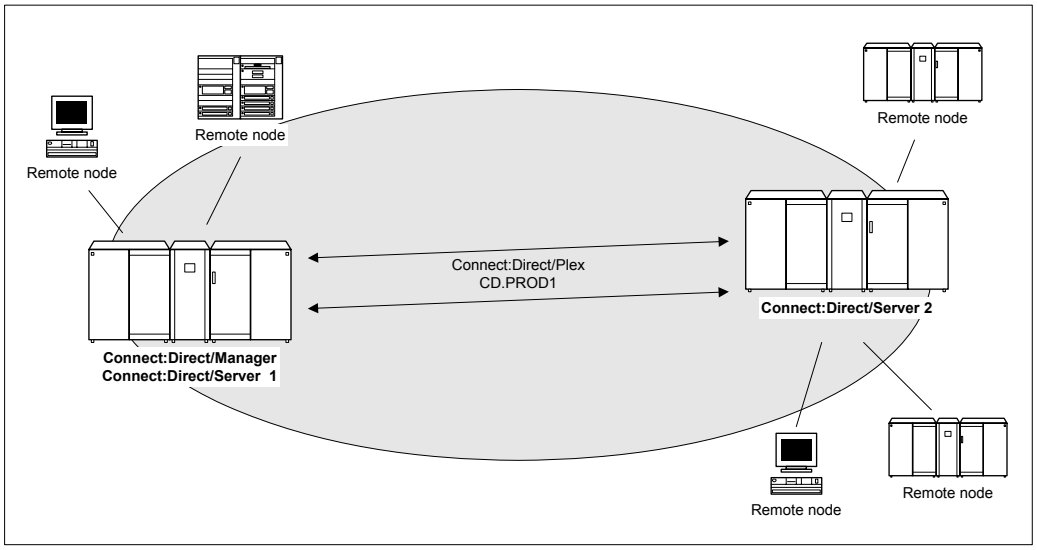
# Converting Two Existing Connect:Direct Stand-Alone Server Systems to a Connect:Direct/Plex Environment

This section describes how to convert two existing production Connect:Direct Stand-alone Server systems into a Connect:Direct/Plex environment with two servers. This configuration takes advantage of the Connect:Direct/Plex single image and workload balancing capability for Processes initiated by this Connect:Direct/Plex environment. This configuration also supports external Connect:Direct systems without requiring any changes to the external systems.

In the following illustration, two separate Connect:Direct Stand-alone Server systems run in a z/OS sysplex environment.



The procedure in this section combines the separate systems into the single Connect:Direct/Plex environment as follows.

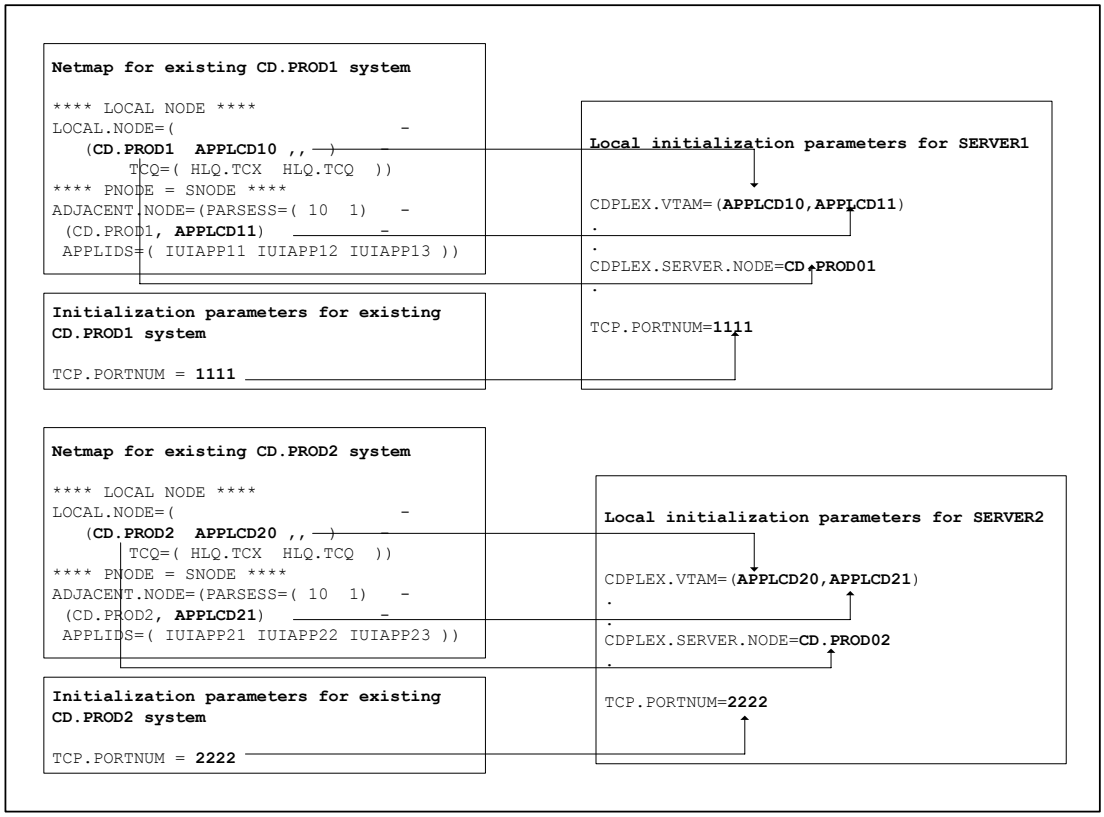



In this configuration, the original CD.PROD1 system becomes the Connect:Direct Manager and Connect:Direct Server1. The CD.PROD2 system becomes the Connect:Direct Server2. The Connect:Direct/Plex environment is given the node name CD.PROD1. No changes are made to the remote nodes' network maps. The remote nodes communicate with the Connect:Direct/Plex environment as if they communicated with a single Connect:Direct image.

To create this configuration you must:

- Define new APPLIDs for the Connect:Direct/Plex environment
- Use the APPLIDs from the existing CD.PROD1 system in the Server1 local initialization parameters
- Use the APPLIDs from the existing CD.PROD2 system in the Server2 local initialization parameters

The following illustration shows how the network map and initialization parameter values from the existing Connect:Direct systems map to the Connect:Direct/Plex environment.




## Assumptions

The example in this section assumes the following:

- You are running two production Connect:Direct Stand-alone Server systems: CD.PROD1 and CD.PROD2.
- The global and local initialization parameter files are located in $CD.PLEX.INITPARM. The JCL that brings up the Connect:Direct/Plex environment is located in $CD.PLEX.JCL. You can either allocate these data sets or use existing data sets in their place.
- The Connect:Direct/Plex environment identifies itself to external systems with a new node name. Each Connect:Direct Server identifies itself to external systems with the same node name it used as a Connect:Direct Stand-alone Server.

## Connect:Direct/Plex Setup

To convert two existing Connect:Direct Stand-alone Server systems into one Connect:Direct/Plex environment:

1. Copy the initialization parameters file from CD.PROD1 into $CD.PLEX.INITPARM as member CDPLX.

2. Resolve any differences (other than data set names) between the new initialization parameters file and the CD.PROD2 initialization parameters file.
3. Copy the CD.PROD1 Connect:Direct/Plex startup JCL into $CD.PLEX.JCL as member CDMGR.
4. Resolve any differences, such as trace DDs, with the CD.PROD2 startup JCL.
5. Merge the existing AUTH and TYPE files from both systems as described in *Connect:Direct/Plex System File Considerations* on page 174, using the same files names used for CD.PROD1.
6. Create new TCQ/TCX, CKPT, and statistics file pairs as discussed in *CKPT file* on page 174, using the same file names used for CD.PROD1.
7. Merge the source from the individual network map files as described in *NETMAP file* on page 175.
8. Change the new network map source (created from the merged network map files) as follows:
   a. Specify new APPLIDs and a new LOCAL.NODE name for the LOCAL.NODE and the PNODE/SNODE ADJACENT.NODE.

      Because the existing APPLIDs are used for SERVER1, you do not need to change the external Connect:Direct connections.

   b. Specify USE.SERVER.NODE=YES on all ADJACENT.NODE records.
   c. Use the same local node name that is used for CD.PROD1.
   d. Load the network map.
9. Add the following initialization parameters to the CDPLX member in $CD.PLEX.INITPARM. This member becomes the Connect:Direct/Plex global initialization parameters file.

   ```
   CDPLEX=YES
   XCF.NAME=8-char-name
   ```

   The CDPLEX=YES parameter indicates a Connect:Direct/Plex environment. It also directs the DTF to read its local initialization parameters from the file specified in the //CDPLEX DD statement in the startup JCL.

   The XCF.NAME parameter specifies a unique name used by the z/OS XCF to assist communications among Connect:Direct/Plex members. This name indicates that the Connect:Direct Manager and Connect:Direct Servers are part of the same XCF group.
10. You can add the following optional parameters to the CDPLX member in $CD.PLEX.INITPARM:
    - CDPLEX.TIMER specifies the time-out value for XCF communications in minutes. Refer to Appendix A, *Global Initialization Parameters*, for more information.
    - CDPLEX.WLM.GOAL specifies whether IBM Workload Manager (WLM) Goal Mode queries are made. This parameter is optional. Refer to Appendix A, *Global Initialization Parameters*, for more information.

11. Create the local initialization parameters files for each Connect:Direct/Plex member:
    a. Copy the MANAGER, SERVER1, and SERVER2 sample local initialization parameters members from the Connect:Direct $cd.PARMLIB library into $CD.PLEX.PARMLIB.
    b. Change the CDPLEX.TCPIP parameter of the MANAGER member to specify the TCP/IP stack address used by the Connect:Direct/Plex Manager.

       You need not change any other parameters need in the MANAGER member.

```
CDPLEX.MANAGER=YES
CDPLEX.TCPIP=nnn.nnn.nnn.nnn
CDPLEX.SERVER.JOBDSN=$CD.PLEX.JCL
CDPLEX.SERVER.JOBMEM=((CDSRV1,SERVER1), -
                      (CDSRV2,SERVER2))
```

    c. Change the CDPLEX.VTAM parameter in the SERVER1 member as follows:
       - Replace the applid11 value with the VTAM APPLID from CD.PROD1 for SNA copy Processes.
       - Replace the applid12 value with the VTAM APPLID from CD.PROD1 for the PNODE-SNODE APPLID.

       These APPLIDs must be unique across the Connect:Direct/Plex and cannot be the same as those specified in the new network map.

```
CDPLEX.MANAGER=NO
CDPLEX.SERVER=SERVER1
CDPLEX.VTAM=(applid11,applid12,applid13)
CDPLEX.PLEXCLASSES=(TAPE,*)
CDPLEX.TCPIP=nnn.nnn.nnn.nnn
TCP.LISTEN=nnnn
```

    d. Change the CDPLEX.TCPIP parameter in the SERVER1 member to specify the TCP/IP stack address used by this Connect:Direct/Plex server.
    e. Change the TCP.LISTEN parameter in the SERVER1 member to the TCP.LISTEN value from CD.PROD1.

> **Note:** The CDPLEX.PLEXCLASSES parameter in SERVER1 specifies a 'TAPE' PLEXCLASS. For Processes that require tape drives, specify the 'TAPE' PLEXCLASS in their Process definitions. These Processes run on SERVER1. (See the *Building, Modifying, and Submitting Processes* chapter of the *Connect:Direct for z/OS User's Guide* for more information on using PLEXCLASS in a Process.)

    f. Add the following statement to the SERVER1 member.

```
CDPLEX.SERVER.NODE=CD.PROD1
```

    g. Change the CDPLEX.VTAM parameter in the SERVER2 member as follows:
       - Replace the applid21 value with the VTAM APPLID from CD.PROD2 for SNA copy

Processes.

- Replace the applid22 value with the VTAM APPLID from CD.PROD2 for the PNODE-SNODE APPLID

These APPLIDs must be unique across the Connect:Direct/Plex and cannot be the same as those specified in the new network map or used for SERVER1.

```
CDPLEX.MANAGER=NO
CDPLEX.SERVER=SERVER2
CDPLEX.VTAM=(applid21,applid22)
CDPLEX.PLEXCLASSES=(*)
CDPLEX.TCPIP=nnn.nnn.nnn.nnn
TCP.LISTEN=nnnn
```

h. Change the CDPLEX.TCPIP parameter in the SERVER2 member to specify the TCP/IP stack address used by this Connect:Direct/Plex server.

i. Change the TCP.LISTEN parameter in the SERVER2 member to the TCP.LISTEN value from CD.PROD2.

j. Add the following statement to the SERVER2 member.

```
CDPLEX.SERVER.NODE=CD.PROD2
```

12. Add the CDPLEX DD statement in the following example to the CDMGR member in $CD.PLEX.JCL. (This JCL is the startup JCL copied in step 3).

```
//DTF EXEC DMINIT,
// PARM='$CD.PLEX.INITPARM(CDPLX)'
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(MANAGER)
```

13. Copy the CDMGR member to CDSRV1.
14. Make the following changes to the CDSRV1 member:
    a. Change the job name so that this job can run simultaneously with the CDMGR JCL.
    b. Change the member name in the CDPLEX DD statement to SERVER1 as follows. This directs the CDSRV1 JCL to its local initialization parameters file.

```
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(SERVER1)
```

15. Copy the modified CDSRV1 member to CDSRV2.
16. Make the following changes to the CDSRV2 member:
    a. Change the job name so that this job can run simultaneously with the CDMGR JCL and CDSRV1 JCL.
    b. Change the member name in the CDPLEX DD statement to SERVER2 as follows. This directs the CDSRV2 JCL to its local initialization parameters file.

```
//CDPLEX DD DISP=SHR,DSN=$CD.PLEX.INITPARM(SERVER2)
```

**Note:** You can route jobs to a different z/OS image by specifying the local node name of the other system name in an XEQ statement in the Connect:Direct Manager or Connect:Direct Server JCL as follows:

```
/*XEQ njenode
```

This example routes the job to the z/OS image identified by the local node name NJENODE.

17. Submit the CDMGR JCL to bring up the Connect:Direct/Plex.

    After the Connect:Direct Manager initializes, it submits the CDSRV1 JCL and CDSRV2 JCL to bring up the two Connect:Direct Servers.

18. After the Connect:Direct Manager initializes, use the IUI to signon to the Connect:Direct Manager.

    You can then submit Processes and perform other functions through the IUI.

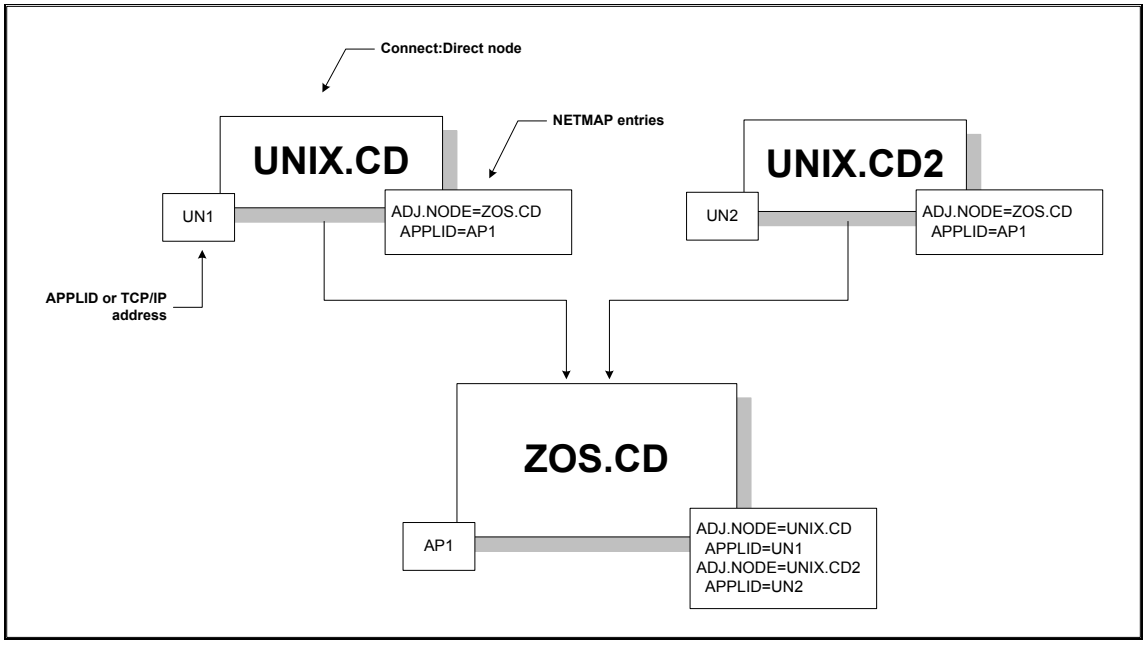
# Additional Configuration Examples

This section shows additional Connect:Direct/Plex configuration examples. While your site configuration can vary due to the number of external nodes, use this section as a guide in determining the best way to configure a Connect:Direct/Plex.

**Note:** The examples in this section are high-level descriptions for use as a configuration model. They do not describe all configuration changes to set up a Connect:Direct/Plex. See *Setting Up a New Connect:Direct/Plex Environment* on page 170 for detailed information about how to set up a Connect:Direct/Plex environment.

## Configuration Examples Using One Connect:Direct for z/OS System

This section assumes the following Connect:Direct environment exists.



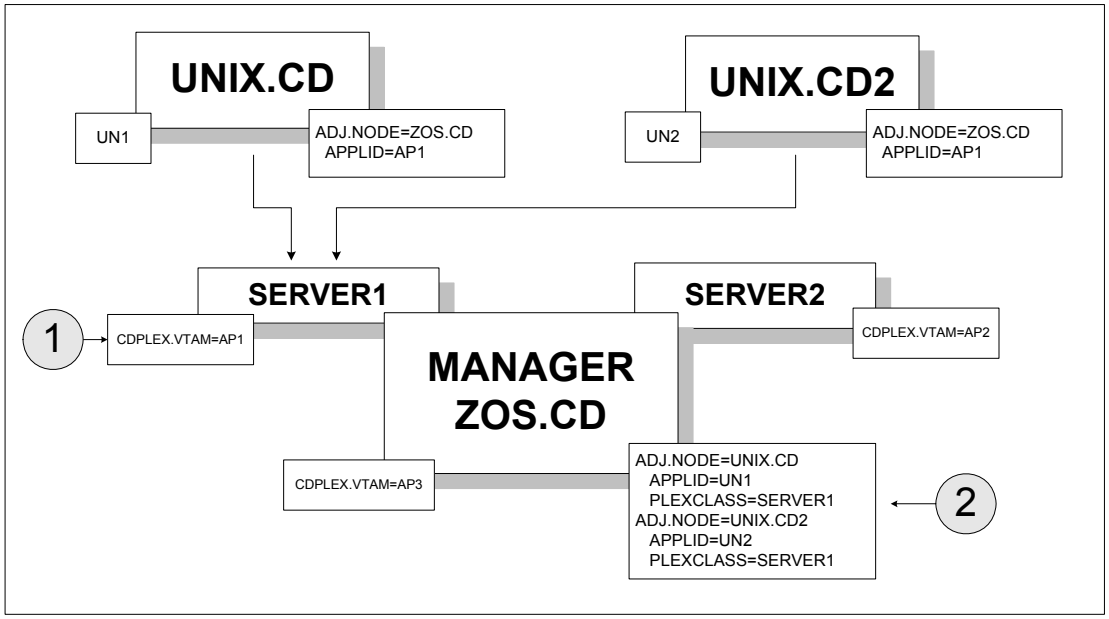

In this environment, two Connect:Direct for UNIX nodes (UNIX.CD and UNIX.CD2) communicate with a Connect:Direct for z/OS system (ZOS.CD). The ZOS.CD system uses the APPLID “AP1.” The UNIX.CD system uses the APPLID “UN1”, while the UNIX.CD2 system uses the APPLID “UN2.” The network map entries define the adjacent nodes.

Although this example uses Connect:Direct for UNIX as the external nodes, the relationship is the same when any Connect:Direct platform is an external node.

### Scenario 1 – External Nodes Communicate with One C:D Server

This section describes the simplest Connect:Direct/Plex configuration – both external nodes communicate with the same Connect:Direct Server.

In the following illustration, the ZOS.CD system is configured as a Connect:Direct/Plex consisting of a Connect:Direct Manager and two Connect:Direct Servers. Both external Connect:Direct for UNIX systems communicate only with SERVER1.




To accomplish this setup, assign the APPLID from the original Connect:Direct system (AP1) to SERVER1 through the local initialization parameters of SERVER1 (callout 1 in the preceding illustration). Note that you must create new APPLIDs for the SERVER2 (AP2) and the Connect:Direct Manager (AP3).

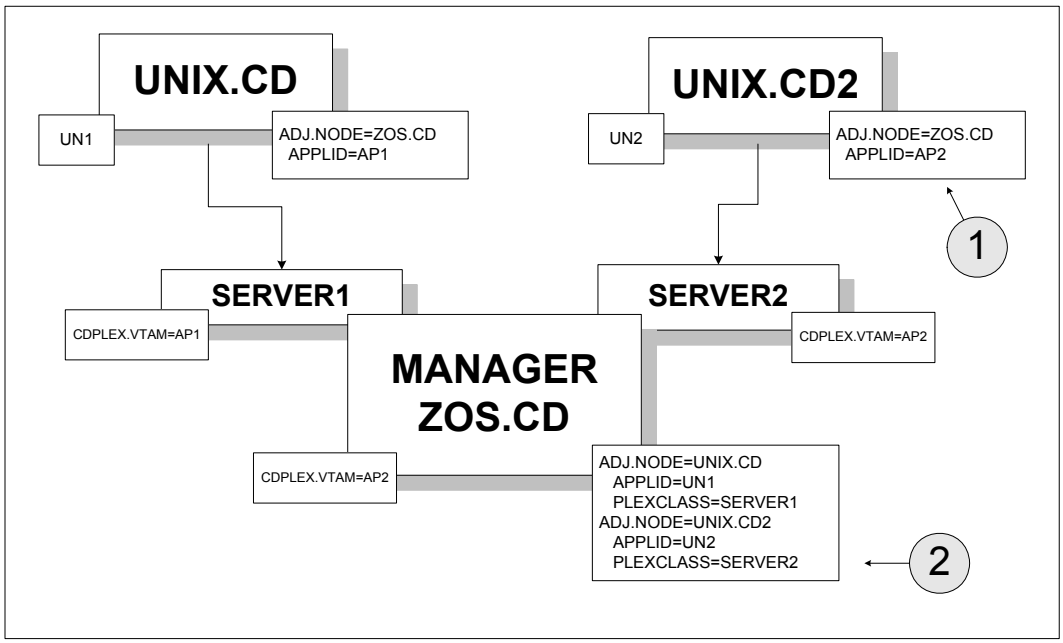
Then, specify SERVER1 as the default PLEXCLASS in the Connect:Direct/Plex network map (callout 2). This routes all work and communication among the nodes through SERVER1.

The advantage of this approach is that the Connect:Direct for UNIX nodes do not need to change any initialization parameter or network map definitions. They do not have any knowledge of the change to the z/OS node.

The disadvantage of this approach is that you cannot use Connect:Direct/Plex workload balancing to its full potential. You cannot perform workload balancing on Processes received from or sent to the external nodes. However, work originating and done entirely on the Connect:Direct for z/OS system can use Connect:Direct/Plex workload balancing.

### Scenario 2 – External Nodes Communicate with Individual C:D Servers

In this configuration, each external Connect:Direct node communicates with a specific Connect:Direct Server. This configuration makes better use of the Connect:Direct/Plex environment by spreading the work from the external nodes between the Connect:Direct Servers.




To accomplish this configuration, change the UNIX.CD2 network map to point to the APPLID for SERVER2 (callout 1).

Then, make the following changes to the Connect:Direct/Plex network map (callout 2):

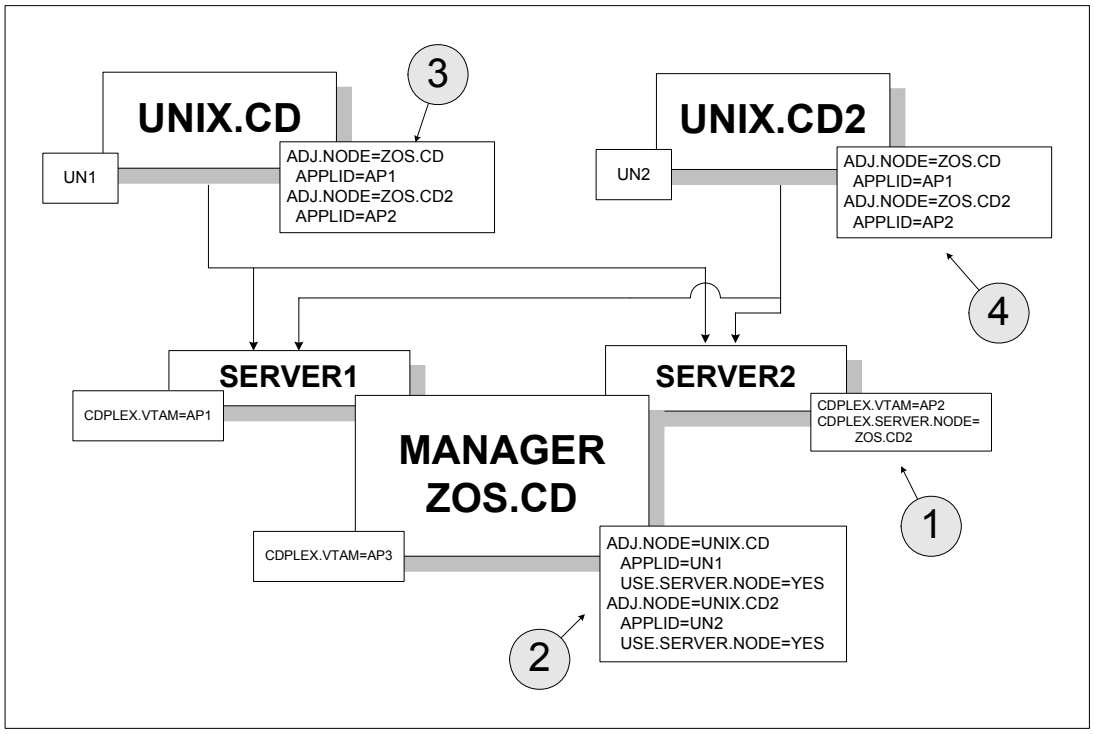
- Route all Processes from UNIX.CD to SERVER1 by defining SERVER1 as the default PLEXCLASS in the adjacent node definition for UNIX.CD.
- Route all Processes from UNIX.CD2 to SERVER2 by defining SERVER2 as the default PLEXCLASS in the adjacent node definition for UNIX.CD2.

The advantage of this approach is that work from each UNIX node runs on a different Connect:Direct Server, so work from one node does not interfere with work from the other. You do not need to change Processes submitted from either node to run on the specified servers (unless the Process itself specifies a TCP/IP address).

The disadvantage of this approach is that you still cannot use the Connect:Direct/Plex workload balancing to its full potential. You cannot perform workload balancing on Processes received from or sent to the external nodes. Work originating and done entirely on the Connect:Direct for z/OS system can use Connect:Direct/Plex workload balancing.

### Scenario 3 – External Nodes Communicate with Both C:D Servers

This configuration uses the Connect:Direct/Plex workload balancing capability. In this environment, both external nodes can communicate with either Connect:Direct Server.




In this configuration, the network map of each Connect:Direct for UNIX node is changed to point to both Connect:Direct Servers. However, because the Connect:Direct/Plex normally is displayed as a single node to external systems, you must first create a unique node name for SERVER2. To create a unique node name, specify:

- CDPLEX.SERVER.NODE = ZOS.CD2 in the SERVER2 local initialization parameters (callout 1)

  You do not need to specify the CDPLEX.SERVER.NODE initialization parameter for SERVER1 because it uses the local node (ZOS.CD).

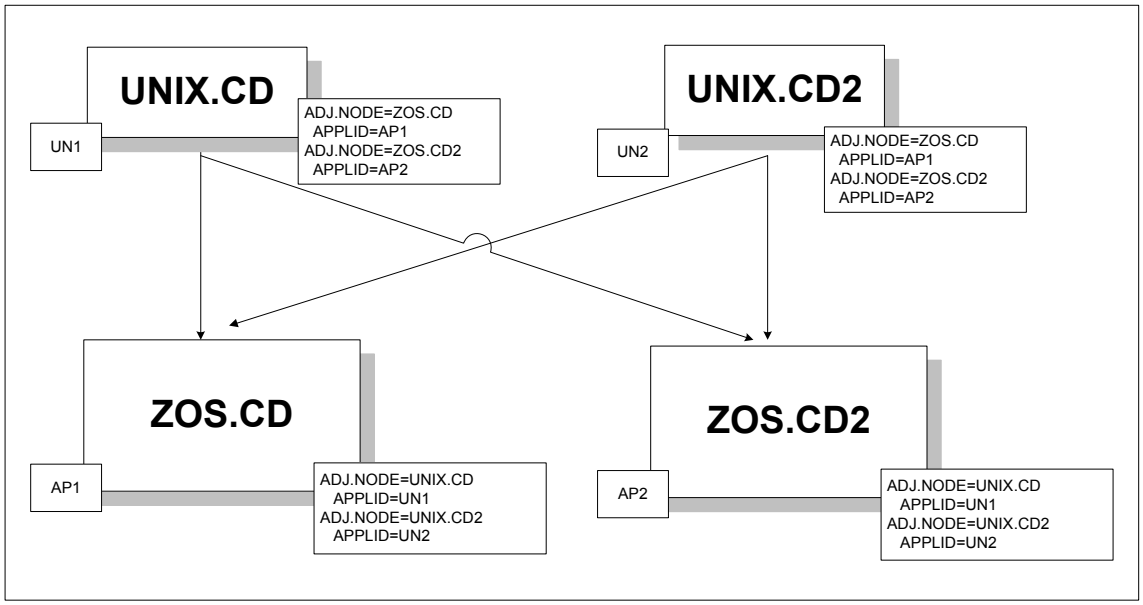
- USE.SERVER.NODE=YES in the Connect:Direct/Plex network map adjacent node definitions (callout 2)

The ZOS.CD2 node name is then added to the external nodes' network maps (callouts 3 and 4).

The advantage of this configuration is that you can perform workload balancing on outgoing Processes from the Connect:Direct/Plex. However, you cannot perform automatic workload balancing on Processes received from the external nodes; you must manually balance them by changing the SNODE.

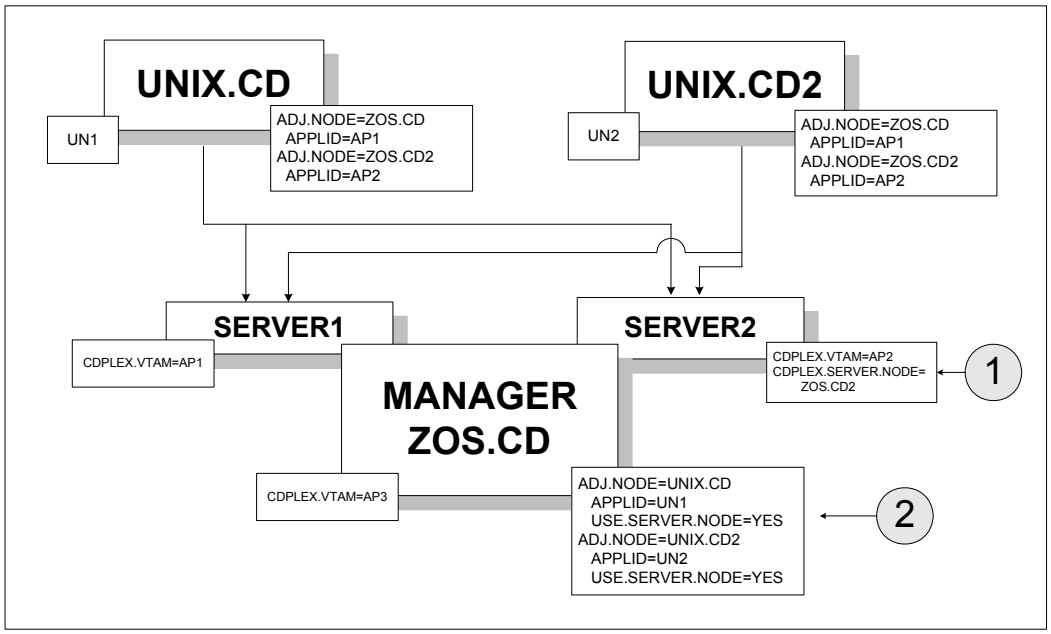
## Configuration Example Using Two Connect:Direct for z/OS Systems

This section describes a more complex Connect:Direct/Plex configuration. It assumes that the following Connect:Direct environment exists.




In this environment, two Connect:Direct for UNIX nodes (UNIX.CD and UNIX.CD2) communicate with two different Connect:Direct for z/OS systems (ZOS.CD and ZOS.CD2).

To change this setup to a Connect:Direct/Plex, the ZOS.CD and ZOS.CD2 systems are merged into a single Connect:Direct/Plex. ZOS.CD is designated as the Connect:Direct Manager and SERVER1, while ZOS.CD2 is designated as the Connect:Direct Server SERVER2. (See *Converting Two Existing Connect:Direct Stand-Alone Server Systems to a Connect:Direct/Plex Environment* on page 182 for more information.)




The initialization parameter CDPLEX.SERVER.NODE=ZOS.CD2 is added to the SERVER2 local initialization parameter (callout 1). USE.SERVER.NODE=YES is added to the Connect:Direct/Plex network map adjacent node definitions (callout 2).

The advantages of this approach are:

- Changes to the z/OS nodes have no affect on the Connect:Direct for UNIX nodes. Therefore, no changes are required to the Connect:Direct for UNIX nodes.
- You can perform workload balancing on Processes sent from ZOS.CD to the Connect:Direct for UNIX nodes.
- The Connect:Direct/Plex provides a single administrative and operating environment.

The disadvantage of this approach is that you must manually balance Processes coming from the Connect:Direct for UNIX nodes.

# Chapter 8

# Configuring Extended Recovery

This chapter describes how to set up extended recovery. It contains the following sections:

- Extended Recovery Overview
- Setting Up Extended Recovery for a Connect:Direct/Stand-Alone Server
- Setting Up Extended Recovery for a Connect:Direct/Plex Environment

## Extended Recovery Overview

Connect:Direct uses the IBM Extended Recovery Facility (XRF) to quickly recover and resume processing after an abnormal termination. This recovery is accomplished by using a standby Connect:Direct system that waits to resume processing if the active Connect:Direct system fails.

Both the Connect:Direct/Stand-alone Server and Connect:Direct/Plex configurations support extended recovery. This chapter describes how to set up extended recovery in either environment.

## Setting Up Extended Recovery for a Connect:Direct/Stand-Alone Server

This section describes how to configure a Connect:Direct/Stand-alone Server to use extended recovery.

1. Specify the XCF.NAME in the initialization parameters file. The XCF.NAME is a unique 8-character string that identifies a Connect:Direct/Stand-alone Server using extended recovery.

   The following example shows a Connect:Direct/Stand-alone Server assigned the XCF.NAME of MNPLS.

```
XCF.NAME=MNPLS
```

**Note:** XCF group names cannot begin with the letters A through J or with the letters SYS because these are reserved by IBM.

2. Add the following parameter to the initialization parameters file.

```
EXTENDED.RECOVERY=YES
```

3. Add the XRFJOB DD statement to the Connect:Direct startup JCL as follows.

```
//XRFJOB DD DISP=SHR,DSN=$CD.CNTL(CDJOBX)
```

$CD.CNTL is the PDS where the active (current) Connect:Direct startup JCL is located. The CDJOBX member is the standby Connect:Direct startup JCL or startup command to run as a started task.

4. To run as a started task, use one of the following, or skip to step 5 if you are not running as a started task.

   - To run the Connect:Direct standby image as a started task on the same z/OS image as the Connect:Direct active image, use the following as the first statement in the JCL.

     ```
     START=membername,parms
     ```

     In this example, START= indicates to issue a START command. When the START command is issued, the equal sign (=) is replaced with a blank, and the entire statement is passed to z/OS as a command.

     For example, if START=HOSTJCL,X is the first statement, then START HOSTJCL,X is issued to z/OS.

   - To run the Connect:Direct standby image as a started task on an z/OS image in the sysplex that is not the Connect:Direct active image, use the following as the first statement in the JCL.

     ```
     /*$VS,'command'
     ```

     Where command is the command you want the Job Entry Subsystem (in this case, a JES2 environment) to send to z/OS. Because the statement does not begin with START=, Connect:Direct submits the statement to JES. JES identifies the /*$VS and issues the command to z/OS rather than placing it in the job queue.

     For example, if /*$VS,'RO CSGB,S CDICOMB' is submitted in the JCL, the RO CSGB,S CDICOMB command is issued rather than placed in the job queue.

5. Copy the Connect:Direct startup JCL to the CDJOBX member if you are submitting the JCL.
6. Change the job name in CDJOBX so that it runs simultaneously with the active Connect:Direct image. (Both the active Connect:Direct image and the standby Connect:Direct image run at the same time.)

7. Change the XRFJOB DD statement in the standby Connect:Direct startup JCL to reference the Connect:Direct startup JCL or command to run as a started task, as in the following example.

```
//XRFJOB DD DISP=SHR,DSN=$CD.CNTL(CDJOB)
```

8. Submit the CDJOB JCL to bring up Connect:Direct using extended recovery.

   After Connect:Direct initializes, the JCL specified in the startup JCL initializes the standby Connect:Direct system. The standby Connect:Direct image partially initializes, then begins monitoring the active Connect:Direct image.

   If the active Connect:Direct image terminates abnormally, the standby Connect:Direct image resumes initialization, becomes the active Connect:Direct image, and submits the JCL in its XRFJOB DD statement. This JCL initializes the original active Connect:Direct image, which now becomes the standby system.

   If the active Connect:Direct image shuts down normally, the standby Connect:Direct image also terminates normally.

# Setting Up Extended Recovery for a Connect:Direct/Plex Environment

This section describes how to configure the Connect:Direct/Plex environment to use extended recovery.

1. Add the following parameter to the initialization parameters file.

```
EXTENDED.RECOVERY=YES
```

2. Add the XRFJOB DD statement to the Connect:Direct startup JCL as follows.

```
//XRFJOB DD DISP=SHR,DSN=$CD.PLEX.JCL(CDMGRX)
```

   $CD.PLEX.JCL is the PDS where the active (current) Connect:Direct startup JCL is located. The CDMGRX member is the standby Connect:Direct startup JCL or command to run as a started task.

3. To run as a started task, use one of the following, or skip to step 4 if you are not running as a started task.

   - To run the Connect:Direct standby image as a started task on the same z/OS image as the Connect:Direct active image, use the following as the first statement in the JCL.

```
START=membername,parms
```

In this example, START=indicates to issue a START command. When the START command is issued, the equal sign (=) is replaced with a blank, and the entire statement is passed to z/OS as a command.

For example, if START=HOSTJCL,X is the first statement, then START HOSTJCL,X is issued to Connect:Direct for z/OS.

- To run the Connect:Direct standby image as a started task on a different z/OS image in the sysplex as the Connect:Direct active image, use the following as the first statement in the JCL.

```
/*$VS,'command'
```

Where command is the command you want JES to send to z/OS (in this case, a JES2 environment). Because the statement does not begin with "START=", Connect:Direct submits the statement to JES. JES identifies the /*$VS and issues the command to z/OS rather then placing it in the job queue.

For example, if /*$VS,'RO CSGB,S CDICOMB' is submitted as JCL, the RO CSGB,S CDICOMB command is issued rather than placed in the job queue.

4. Copy the changed CDMGR JCL member to CDMGRX.
5. Change the job name in CDMGRX so that it runs simultaneously with other Connect:Direct/Plex members.
6. Change the XRFJOB DD statement in CDMGRX to point to the CDMGR member, as in the following example.

```
//XRFJOB DD DISP=SHR,DSN=$CD.PLEX.JCL(CDMGR)
```

7. Add the following DD statement to the CDSRV1 member (the SERVER1 startup JCL) in $CD.PLEX.JCL.

```
//XRFJOB DD DISP=SHR,DSN=$CD.PLEX.JCL(CDSRV1X)
```

8. Copy the changed CDSRV1 JCL member to CDSRV1X.
9. Change the job name in CDSRV1X so that it runs simultaneously with other Connect:Direct/Plex members.
10. Change the XRFJOB DD statement in CDSRV1X to point to CDSRV1, as in the following example.

```
//XRFJOB DD DISP=SHR,DSN=$CD.PLEX.JCL(CDSRV1)
```

11. Add the following DD statement to the CDSRV2 member (the SERVER2 startup JCL) in $CD.PLEX.JCL.

```
//XRFJOB DD DISP=SHR,DSN=$CD.PLEX.JCL(CDSRV2X)
```

12. Copy the changed CDSRV2 member to CDSRV2X.
13. Change the job name in CDSRV2X so it can run simultaneously with other Connect:Direct/Plex members.
14. Change the XRFJOB DD statement in CDSRV2X to point to CDSRV2, as in the following example.

```
//XRFJOB DD DISP=SHR,DSN=$CD.PLEX.JCL(CDSRV2)
```

15. Submit the CDMGR JCL to bring up the Connect:Direct image using extended recovery.

    Each Connect:Direct member initializes using extended recovery and submits the JCL specified in its XRFJOB DD statement. This JCL starts the standby Connect:Direct members. Each standby Connect:Direct member partially initializes, then begins monitoring its active Connect:Direct member.

    If the active member terminates abnormally, or is shut down with a STOP CD CDPLEX RECOVER command, the standby member resumes initialization, becomes the active member, and submits the JCL in its XRFJOB DD statement. This JCL initializes the original active Connect:Direct/Plex member, which now becomes the standby member.

**Note:** If you want the standby member to run on a different z/OS image in the sysplex, you must define the VTAM APPLID as dynamic in both z/OS images, and you must define TCP/IP addresses as dynamic VIPA addresses.

# Chapter 9

# Configuring SNMP Support

This chapter describes how to set up SNMP support. It contains the following sections:

- SNMP Support Overview
- Identifying the Trap Variables
- Setting Up SNMP

## SNMP Support Overview

The Simple Network Management Protocol (SNMP) defines a set of protocols that describe management data and the protocols for exchanging that data between systems. This management data is a set of defined variables called the Management Information Base (MIB).

Three primary functional entities are defined for SNMP: managers, agents, and subagents. A manager is a network management application, like Netview or HP OpenView. The agent is a server that responds to request for management data from a network manager. Subagents provide support for particular MIBS to the agent.

The primary function of an SNMP environment and the communication between these functional entities is to enable the network manager to poll a device or application to inquire about specific management data that the device or application is monitoring. The device or application can alert the network manager of certain conditions by sending traps to reflect the status of that condition. Traps are asynchronous, unsolicited messages sent to the network manager, when the agent and/or subagent detect certain conditions.

Connect:Direct provides support for an SNMP agent to send SNMP traps to alert a network manager of certain events. An event is any Connect:Direct message that is written to the console using Connect:Direct members. Each event is triggered by the Connect:Direct message ID and the trap text (short message text of that Connect:Direct message). The Connect:Direct events generated are defined by category and type.

The Connect:Direct Trap Table can hold up to 127 entries. It is built using the input from the data set specified by the SNMP.DSN initialization parameter. One entry is generated for each message that can trigger a SNMP trap. The predefined Connect:Direct traps are triggered by 61 messages. A maximum of 66 additional user messages can be used to trigger SNMP traps. To see information

about each SNMP trap defined in the table along with its status, see *Displaying the SNMP Trap Table* on page 210.

Define message traps using the SNMP.DSN initialization parameter and a data set that contains the variables associated with traps. The traps and associated variables are listed in the following section.

# Identifying the Trap Variables

Traps are defined as alarm or status alerts which enable the network manager to display the trap in the appropriate color on the network manager console. Alarm trap variables signal events that are critical to the operation of Connect:Direct. Status trap variables signal events that are not critical to the operation of Connect:Direct, but show valuable information. The tables in the following sections describe the predefined traps, the message that triggers the trap, and a description of the trap and associated text.

Following are the six categories for trap variables:

- Type events
- Initialization events
- Shutdown events
- API events
- Execution events
- STATS events
- Miscellaneous events

Valid values for all events is YES to enable and NO to disable.

## Type Events

Use the events in the following table to enable or disable all alarm events or all status events.

| Trap Event | Description | Event |
|---|---|---|
| sendAlarmTraps | NO disables all Alarm Trap Variables regardless of individual settings. YES enables all Alarm Trap Variables, allowing you to disable individual Alarm Trap Variables | Alarm |
| sendStatusTraps | NO disables all Status Trap Variables regardless of individual settings. YES enables all Status Trap Variables, allowing you to disable individual Status Trap Variables | Status |

## Initialization Events

The following table details alarm and status events that occur at initialization.

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| apInitFailure | An initialization error occurred during open processing of the AP Key file. | APSM012E | Unable to initialize the AP system. | Alarm |
| apKeyValidFailure | The AP Key file is invalid. | APSM008E | AP Key is not valid. | Alarm |
| emergencyKey | The EMERGENCY-KEY is in use. | APSM998I | Connect:Direct Initialization using Emergency Key. | Alarm |
| productHasExpired | The Connect:Direct product license expired. | APSM009E | AP Key has expired. | Alarm |
| | | APSM010E | Warning, AP Key has expired. | |
| productWillExpire | The Connect:Direct product expires in x number of days. | APSM011E | AP Key will expire in &VALUE days, &OPTION. | Alarm |
| generalAPFailure | An unexpected failure occurred while processing the AP Key file. | APSM000E | AP detected an unauthorized program modification. | Alarm |
| | | APSM001E | AP Key is not for this system, &VALUE. | |
| | | APSM002E | AP Key will not be valid for &VALUE days. | |
| | | APSM004E | Warning, AP Key option has expired, &option. | |
| | | APSM005E | AP option, &option, will expire in &value days. | |
| | | APSM011E | AP Key will expire in &VALUE days, &OPTION. | |
| | | APSM015E | AP Key not valid. | |
| Initializationcomplete | The initialization of this Connect:Direct node has completed successfully. In a Connect:Direct Plex environment, the member of the Connect:Direct Plex is named in the message text. | SITA036I | Connect:Direct rel-level for z/OS Initialization Complete. | Status |

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| snaNotAvailable | The SNA support is temporarily unavailable, either because the SNA=NO initialization parameter is specified, the VTAM ACB is inactive and could not be opened, the VTAM ACB is disabled during Connect:Direct processing or the Connect:Direct VTAM APPLID is already in use by another Connect:Direct. | SVTJ018I | SNA Support is Not Available. | Status |
| snaNowAvailable | The VTAM ACB is successfully opened and Connect:Direct now supports all SNA functions. | SVTJ019I | SNA Support is Now Available. | Status |
| tcpNotAvailable | The TCP support is temporarily unavailable either because the TCP=NO initialization parameter is specified, the TCP/IP connection cannot be established, or the connection to TCP/IP is terminated. | STCP103I | TCP Support is Not Available. | Status |
| tcpNowAvailable | The connection to TCP/IP is successful and all TCP functions are now supported. | STCP104I | TCP Support is Now Available. | Status |

## Shutdown Events

The following table details alarm and status events that occur at shutdown.

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| abnormalShutdown | An abnormal termination of Connect:Direct occurred. | SSHA021I | Abnormal termination of Connect:Direct. | Alarm |
| shutdownRequest | A Connect:Direct Stop command issued. | SSHA002I | Connect:Direct QUIESCE shutdown begun. | Status |
| | | SSHA003I | Connect:Direct Run Task IMMEDIATE shutdown begun. | |

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| | | SSHA004I | Connect:Direct IMMEDIATE shutdown begun. | |
| | | SSHA019I | Connect:Direct STEP shutdown begun. | |
| normalShutdownComplete | Connect:Direct normal shutdown completed successfully. | SITB001I | Connect:Direct Termination Complete. | Status |

## API Events

The following table details alarm and status events that occur from the API.

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| maxBatchReached | MAX.BATCH is reached. | STAA009I | Task not created, MAX BATCH task count reached | Status |
| maxUserReached | MAX.USER is reached. | STAA004I | Task not created, MAX IUI/API task count reached. | Status |

## Execution Events

The following table details alarm and status events that occur when a Process executes.

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| processFailure | A Connect:Direct Process failed with a return code greater than 0, due to abnormal session termination, NETMAP check failure, or FM72 Security failure. | SVTM024I | &var1 EXECPROC: FMH-72 ¬RECEIVED; | Alarm |
| | | SVTM026I | SESSION (&class) NOT ESTABLISHED WITH &node=&snode | |
| | | SVTM030I | &var1 FMH-74 ¬RECEIVED AFTER STEP ERROR: | |
| | | SVTM050I | &var1 PROCESS INTERRUPTED: RECOVERY INITIATED | |

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| | | SVTM052I | &stpnm &func &pname(&pnum) &node=&snode &var1 | |
| | | SVTM054I | &var1 SNODE REQUESTING SESSION SHUTDOWN F/END_OF_STEP | |
| | | SVTM063I | PASSWORD NOT MATCHED IN C:D AUTH FILE--MSG=SAFB005I | |
| | | SVTM102I | MSGID=&mgid &msgtext,NODE=&snode SENSE=&sense LUNAME=&slu | |
| sessionRetryExceeded | The Connect:Direct Process exceeded the session retry threshold and is placed in the Hold queue. | SVTM505I | Session Retry exceeded for &pname &pnum | Alarm |
| processRetryExceeded | The Connect:Direct Process exceeded the process retry threshold and is placed in the Hold queue. | SVTM506I | Process Retry exceeded for &pname &pnum | Alarm |
| maxProcess | MAXPROCESS value is reached. | STAA010I | Task not created. Max Process count reached. | Alarm |
| maxPnode | The maximum number of PNODE Processes is reached. | STAA006I | Task not created. Max primary task count reached. | Alarm |
| ProcessNotStarted | Process was not started because Connect:Direct quiesced and a task could not be created. | STAA011I | SNODE task not created, Session Quiesce in progress | Alarm |
| | | STAA012I | PNODE task not created, Session Quiesce in progress | |
| | | STAA008I | Task not created. No free TCA available. | |
| | | STAA003I | Task not created. Connect:Direct is quiescing. | |
| maxSnode | The maximum number of SNODE Processes is reached. | STAA005I | Task not created. Max secondary task count reached. | Alarm |

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| tcpCloseFailure | A TCP Close failed leaving the TCP/IP socket in use and unavailable. | STCP105I | TCP Close Socket Failure | Alarm |
| userMessageAlarm | A user-defined Connect:Direct message is issued. | User-defined | message text | Alarm |
| tcqMovement | A Connect:Direct Process is moved to the Hold queue due to errors during Process execution. | SVTM105I | PNAME=&pname, PNUM=&pnum MOVED TO Q=&q, QSTATUS=&qstat | Status |
| processFlushed | The Connect:Direct Process is flushed. | SOPD049I | Connect:Direct Process,&pnam &pnum, flushed by &userid. | Status |
| userMessageStatus | A user-defined Connect:Direct message is issued. | User-defined | message text | Status |

## STATS Events

The following table details alarm and status events that occur due to the STATS queue.

| Trap Event | Description | Trap Trigger | Short Message Text | Event |
|---|---|---|---|---|
| statsDisabled | An error occurred that caused the STATS logging to be disabled. | SSTL001I | Statistics logging function is disabled. | Alarm |
| statsStress | Connect:Direct STATS queue is under stress. | SSTL041I | Statistics facility under stress, waiting on queue elements. | Alarm |
| statsStressResolved | The STATS Queue stress is resolved. | SSTL042I | Statistics facility stress resolved. | Alarm |
| statsSwitchOccurred | A Connect:Direct STATS file switch has occurred. | SSTL013I | Statistics file pair switch from &a to &b | Status |

### Miscellaneous Events

The following table details other alarm and status events.

| Trap Event | Description | Trap Trigger | Short Message Text | Type |
|---|---|---|---|---|
| tracesEnabled | A Connect:Direct MODIFY DEBUG command is issued. | STRA028I | Connect:Direct Traces enabled. | Status |
| netmapUpdate | Dynamic update of the Connect:Direct NETMAP occurred. | SMUP191I | Connect:Direct NETMAP file updated. | Status |
| authUpdate | Dynamic update of the Connect:Direct AUTH file occurred | SAFC005I | Connect:Direct AUTH file updated. | Status |
| typeUpdate | Dynamic update of the Connect:Direct TYPE file occurred | SAFI013I | Connect:Direct TYPE file updated. | Status |
| initparmRefresh | Dynamic update of the Connect:Direct INITPARM file occurred | SITA992I | INITPARM Refresh by &userid completed | Status |
| changeProcess | A CHANGE PROCESS command occurred. | SOPB017I | Change Process command by &userid completed. | Status |
| deleteprocess | A DELETE PROCESS command occurred. | SOPC011I | Delete Process command by &userid completed | Status |
| tcqFull | TCQ file is full. | SPQL001I | TCQ File Full | Alarm |
| tcqThreshold | TCQ file becoming full. | SPQL002I | TCQ Full &VAR1% Full. Max.# CI: &VAR2 # Used CI: &VAR3 | Alarm |
| tcqThresholdResolved | TCQ is now below the defined threshold. | SPQL003I | TCQ File is now below the user defined Threshold of%%. | Status |

## Setting Up SNMP

Use the following procedure to set up SNMP Support.

**Note:** Before performing this procedure, you must migrate the CDMIB and Connect:Direct Trap Configuration files to HP OpenView. Refer to the *Customizing SNMP* section in the *Installing Connect:Direct* chapter of the *Connect:Direct for z/OS Installation Guide* for more information.

1. Specify SNMP=YES in the initialization parameters file.

2. If you want to exclude or disable any trap event or define additional trap triggers described in *Identifying the Trap Variables* on page 202, create a data set containing all the trap events that you want to disable. Following is an example.

   **Note:** All traps are enabled by default.

   ```
   sendStatusTraps = N
   statsDisabled = N
   statsStress = N
   statsStressResolved = N
   userMessageAlarm = ( SVTM100I ,  SVTM101I  )
   ```

   **Note:** A sample SNMP.DSN file is in the $CD.SAMPLIB data set, member CDSNMP.

   If you do not want to exclude any trap events, go to step 4.

3. Set the SNMP.DSN initialization parameter to the data set name created in step 2.

   ```
   SNMP.DSN=data set name
   ```

4. Set the SNMP.MANAGER.ADDR initialization parameter. This parameter is the TCP/IP address or hostname of the host where the SNMP Network Manager is initialized. By default, this address is the same as the TCP/IP address that Connect:Direct is using or the local hostname. In a Connect:Direct/Plex environment, the default is the TCP/IP address of the Connect:Direct Manager. This parameter is required if the network manager resides on a different host or is required to use a different TCP/IP address. Following is an example.

   ```
   SNMP.MANAGER.ADDR=123.4.5.6
   ```

5. Set the SNMP.MANAGER.PORTNUM initialization parameter. This port is the TCP/IP port that is defined for UDP traffic to the network manager. The default is 162. This parameter is required if the defined UDP port number is something other than 162. Following is an example.

   ```
   SNMP.MANAGER.PORTNUM=163
   ```

At Connect:Direct installation, the SNMP trap table is initialized and whenever any event occurs after the SITA628I message is issued, Connect:Direct determines if the event is a trap trigger and issues the appropriate trap to a network manager. The following messages are common at initialization:

- SITA001I Connect:Direct initialization begun.
- SITA002I Connect:Direct parameter file allocated and open.
- SITA022I Loading Connect:Direct modules.
- SITA601I The TCP server modules are loaded.

- ✦ SITA067I MESSAGE file is open.
- ✦ SITA628I SNMP Trap Agent Initialization Complete.

If any error occurs during initialization of SNMP, the appropriate message is issued to indicate that the SNMP Trap Agent is disabled or that the initialization will terminate.

You can refresh and modify the SNMP initialization parameters after initialization completes by using the MODIFY INITPARM command.

## Displaying the SNMP Trap Table

The INQUIRE SNMP command displays the contents of the SNMP trap table.

### Command Format

The INQUIRE SNMP command has the following format.

| Label | Command | Parameter |
|---|---|---|
| (optional) | INQuire SNMP | |

### Issuing the INQUIRE SNMP Command through the IUI

To display the contents of the SNMP trap table from the IUI:

1. Select option INQ from the Connect:Direct Administrative Options Menu.

   The Inquire DTF Internal Status screen is displayed.
2. Select the ITRP option.
3. Press **ENTER**.

   The contents of the SNMP trap table are displayed, as in the following sample.

```
******************************** Top of Data ***********************
================================================================================
node.name         *SNMP TRAP TABLE*   DATE: mm.dd.yyyy TIME: hh:mm:ss
================================================================================
TRAP TRIGGER   TRAP NAME                  TRAP STATUS TRAP TYPE
================================================================================
APSM012E       APINITFAILURE              ENABLED     ALARM
APSM008E       APKEYVALIDFAILURE          ENABLED     ALARM
APSM998I       EMERGENCYKEY               ENABLED     ALARM
APSM010E       PRODUCTHASEXPIRED          ENABLED     ALARM
APSM009E       PRODUCTHASEXPIRED          ENABLED     ALARM
APSM011E       PRODUCTWILLEXPIRE          ENABLED     ALARM
APSM002E       GENERALAPFAILURE           ENABLED     ALARM
APSM005E       GENERALAPFAILURE           ENABLED     ALARM
APSM001E       GENERALAPFAILURE           ENABLED     ALARM
APSM000E       GENERALAPFAILURE           ENABLED     ALARM
APSM015E       GENERALAPFAILURE           ENABLED     ALARM
APSM004E       GENERALAPFAILURE           ENABLED     ALARM
APSM003E       GENERALAPFAILURE           ENABLED     ALARM
```

### Using the INQUIRE SNMP Command from the Batch Interface

To use the INQUIRE SNMP command from the Batch interface:

1. Place your commands in a batch job stream as demonstrated in the *Connect:Direct for z/OS User's Guide*.
2. Submit the job while Connect:Direct is running. The settings are displayed.

**Note:** 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.

# Chapter 10

# Using Connect:Direct Exits

**Note:** APARs and PTFs from Sterling Commerce will have "HOLD for ACTION" directives that will tell you if reassembly of exits is required. Please reassemble exits accordingly. If you are upgrading from a prior release, reassemble your exits to pick up any macro or control block changes.

This chapter includes information about the following Connect:Direct exits:

- Statistics Exit
- Submit Exit
- Allocation Exit
- I/O Exits
- Data Exit
- WLM Exit
- Tapemount Exit

Connect:Direct provides several sample exits some of which can be used to customize the online execution of Connect:Direct. One of these exits, the Stage 2 Security exit, can be used to test new applications and customer connections. For more information, see *Process Exit for Testing (NDMPRCXT)* on page 249.

Before coding or using an exit, read *Special Considerations* on page 260 to ensure that the exit executes properly.

**Note:** All sample exits provided in Connect:Direct define the proper AMODE and RMODE settings within the source member themselves. All user exits should be link-edited with AMODE=ANY and capable of executing in 31-bit mode. Each user exit should preserve the mode in which it was invoked and return to the caller in the proper mode. Modules written to execute in 31-bit mode can be link-edited with RMODE=ANY or RMODE=24. Check the source for the sample exits to see how Connect:Direct defines the proper AMODE and RMODE settings.

**Note:** To notate modifications by user in the the module maintenance history section of NDMLOG, include the local user ID by specifying the &UID SETC 'xxxx' local identifier in Connect:Direct exits as part of the SCENTER macro. This value can contain up to 8 characters.

**Note:** Sample JCL for assembling user exits is provide in SAMPLIB members ASMSTG1 and ASMSTG2.

# Statistics Exit

Connect:Direct generates and logs statistics to an online journal, then writes the information to the Connect:Direct statistics log as individual records.

Each record contains information about a single event, and is identified by a 2-character record type. For example, type CT is a copy termination record and FP is a flush Process record.

Connect:Direct provides a statistics exit that gives a user-written program access to the statistics records as they are generated. This exit can:

- Output the records or data generated from the records to a user-defined journal, including an SMF log
- Include or exclude the logging of any record to Connect:Direct by return codes

***Caution:*** Statistics records are often essential in debugging Connect:Direct problems. Excluding records from the statistics log makes problem determination more difficult or even impossible.

The statistics exit runs as a subtask in the Connect:Direct DTF address space. Connect:Direct uses the STATISTICS.EXIT initialization parameter to specify the exit module name. You define this name, but it cannot conflict with the name of any Connect:Direct module. If a user-defined journal is required, you must add the necessary data definition (DD) statements to the Connect:Direct startup job stream.

**Note:** In a Connect:Direct/Plex environment, the statistics exit, by default, only runs on the Connect:Direct/Manager. See the discussion on the STATISTICS.EXIT initialization parameter on page 419 for more information on how to cause the statistics exit to run on the Connect:Direct/Plex server.

The statistics exit indicates if a record is logged by setting a return code set in the SQUSER field of the SQCB. The return code is initialized to zero before the exit is invoked.

You do not have to rewrite existing exits if you do not want to exclude records. Existing exits do not alter the SQUSER field and operate the same as before.

## Sample Statistics Exits

Connect:Direct provides the following sample statistics exits in SAMPLIB.

| Exits | Description |
|---|---|
| DMGSMF | This sample exit logs Connect:Direct statistics records to the SMF log. It logs records to the SMF log by prefixing each statistics record with an SMF record header, and then uses the SMFWTM macro to write to the SMF log. The SMF record is type 132. |
| ESSEVX01 | This sample exit provides a means for an application to access event data.<br>This exit writes each event record to a predefined data set. You must modify ESSEVX01 to specify the name of your event exit data set, and define the data set to accommodate records up to 2048 bytes in length.<br>For more information on how to use the Event Services Support (ESS) facility with this exit, see the *Connect:Direct for z/OS Facilities Guide*. |
| STATACCT | This exit documents the path to log user accounting data from the copy termination (CT) records.<br>This exit is invoked for every Connect:Direct statistics record written to the statistics file, but only processes CT records. |
| STATEXIT | This sample exit simply checks for copy termination records. When Connect:Direct encounters a copy termination, the system issues a WTO. |
| STATEXMC | This sample exit precludes the logging of PDS member copy (MC) records that have good return codes. It enables the logging of records of this type only when they have non-zero return codes. It also enables the logging of all other record types regardless of their return codes. |

## Statistics Exit Calling Conventions

Connect:Direct calls the statistics exit once for each statistics record generated. Standard linkage conventions apply.

### Information Passed

The exit is given control with register 1 pointing to a list of two parameters. They are:

- The first parameter is a pointer to the statistics record.
- The second parameter is a pointer to an SQCB that you need for setting a return code if record exclusion is appropriate.

The first 2 bytes of the record contain the record length in binary format. The third and fourth bytes of the record contain the 2-character record identifier. The table on page 216 contains a list of the record type identifiers.

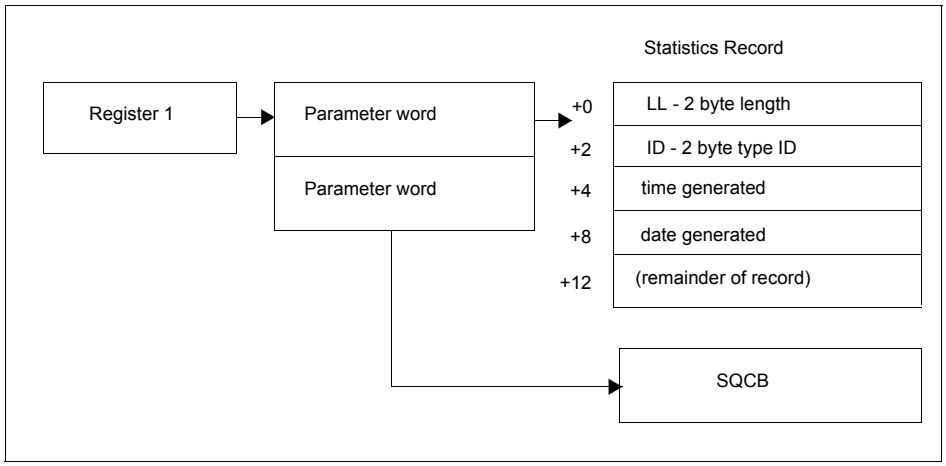
The second word of the record contains the time of day that the record is generated. The third word contains the date the record is generated.

Additional information in the records depends on the record type.

Assembler macros are provided in SAMPLIB to generate dummy sections (DSECTS) to map all the record types. The exit program includes the DSECTS that map the record types to the exit processes.

A return code of **0** indicates that the record is logged. A return code of **4** indicates that the record is not logged.

The following figure depicts the information passed to the exit.




## Statistics Records

Connect:Direct calls the statistics exit once for each statistics record generated in the DTF. The input to the exit is a pointer to the statistics record that is ready to be logged and a pointer to an SQCB. The record can be any record type. The exit must examine the record type identifier at a displacement of X'02' bytes from the beginning of the record to determine the record type and the DSECT that describes its contents.

The following table lists the statistics record types, their corresponding record type identifiers, and the name of the assembler macro in SAMPLIB that generates the DSECT describing the record contents.

For information about selecting, displaying, and printing statistics information for Connect:Direct activities, see the *Connect:Direct for z/OS User's Guide.*

The statistics records in this section also apply to Event Services Support.

| Record ID | Description | Macro |
|---|---|---|
| CE | Copy I/O Start | DMSTEPR |
| CH | Change Process | DMCPTR |
| CI | Copy Step Start | DMSTEPR |
| CS | Statistics Command | DMFSCMD |

| Record ID | Description | Macro |
|---|---|---|
| CT | Copy Termination | DMCTR |
| DC | Directory Commands | DMDTR |
| DP | Delete Process | DMDPTR |
| DT | Select Task | DMDTR |
| DU | Delete User | DMAER |
| EI | Event Services Start Command | DMEVR |
| ET | Event Services Stop Command | DMEVR |
| EV | Event Services Command | DMEVR |
| FA | IGWFAMS Message | DMFAMSCR |
| FI | Long File Name Record | DMFIR |
| FP | Flush Process | DMFPTR |
| FS | Suspend Process | DMFPTR |
| FT | Flush Task | DMDTR |
| GO | Process Modal - GOTO, ELSE, or EXIT Statement | DMMODAL |
| HW | High Concurrent Session Count | DMHWR |
| IA | Inquire Statistics | DMDTR |
| IB | Inquire Debug | DMDTR |
| ID | Inquire STATDIR | DMDTR |
| IF | Process Modal - IF Statement | DMMODAL |
| IK | Inquire APKey File | DMDTR |
| IP | Inquire Initialization parameters | DMDTR |
| IT | Inquire SNMP Trap Table | DMDTR |
| IU | Insert User | DMAER |
| IX | Inquire Connect:Direct/Plex | DMDTR |
| JI | Run Job Start | DMSTEPR |
| LF | ICO Log File Record | DMFREPRT |
| M2 | Multiple Copy Record | DMDSSIOX |
| MC | PDS Member Copy | DMFMCR |
| NL | Process modal - EIF or PEND statement | DMMODAL |
| NM | NETMAP Updated | DMNMR |

| Record ID | Description | Macro |
|---|---|---|
| PE | Connect:Direct/Plex Error Record | DMPER |
| PI | Process Start | DMPIR |
| PR | Performance Measurement Record | DMPRRB |
| PS | Process Submit | DMPSSR |
| PT | Process Termination | DMPTR |
| PX | Connect:Direct/Plex Activity (Leave or Join Connect:Direct/Plex) | DMPXR |
| QE | Queue Change to EXEC Queue | DMQCR |
| QH | Queue Change to HOLD Queue | DMQCR |
| QT | Queue Change to TIMER Queue | DMQCR |
| QW | Queue Change to WAIT Queue | DMQCR |
| RE | ICO Report Record | DMFREPRT |
| RJ | Run Job | DMRJTR |
| RO | ICO Event Record | DMFROEVT |
| RT | Run Task | DMRTTR |
| S2 | Statistics Logging Statistics | DMFS2R |
| SB | Session Begin | DMSSCR |
| SC | Statistics Control Record | DMFSCR |
| SD | Start Connect:Direct | DMSDCR |
| SF | Statistics Format | DMFSFREC |
| SI | Signon | DMSFR |
| SN | Select Netmap | DMDTR |
| SO | Signoff | DMSFR |
| SP | Select Process | DMDTR |
| SS | Select Statistics | DMDTR |
| ST | Stop Connect:Direct | DMSTDCR |
| SU | Select User | DMAER |
| SW | Submit within a Process | DMPSSR |
| SY | SYSOPTS | DMSYR |
| TF | TCQ Threshold Full | DMTXR |
| TI | Run Task Start | DMSTEPR |

| Record ID | Description | Macro |
|---|---|---|
| TL | TCQ Threshold Low | DMTXR |
| TP | Throughput Record Statistics | DMTPR |
| TR | Trap Event Record | DMTRP |
| TS | Suspend Task | DMFPTR |
| TW | TCQ Threshold Warning | DMTXR |
| UM | Update Network map | DMAER |
| UU | Update User | DMAER |
| VP | View Process | DMDTR |
| WO | WTO | DMFWTOST |
| WS | Select Stat Command | DMDTR |
| XO | Trace On/Off | DMXOR |
| ZI | SNODE Process Start | DMPIR |
| ZT | SNODE Process Terminated | DMPTR |

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 |
| DMDTR | Display Termination Record |
| DMEVR | Event Services Command Statistics Record |
| DMFAMSCR | IGWFAMS (File and Attribute Management Services) Macro Statistics Record |
| DMFIR | Long File Name Statistics Record |
| DMFMCR | PDS Member Copy Record |
| DMFPTR | Flush and Suspend Process Statistics 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) |

| Macro Name | Description |
|---|---|
| DMFRJXCB | Run Job exit |
| DMFROEVT | InterConnect Report Event Record containing one record per report written (Record Type is RO) |
| DMFS2R | Statistics Logging Record |
| DMFSCMD | Statistics Command Record |
| DMFSCR | Statistics ESDS Control Record |
| DMFSFREC | Statistics Format Record |
| DMFWTOST | WTO Statistics Record |
| DMHWR | High Concurrent Session Count Statistics Record |
| DMLSR | Log Swap Statistics Record |
| DMNMR | NETMAP Updated |
| DMPER | XCF Error Message Statistics Record (from Sysplex) |
| DMPIR | Process Initiation Statistics Record |
| DMPRRB | Performance Measurement Statistics Record |
| DMPSSR | Submit Process Statistics Record |
| DMPTR | Process Termination Record |
| DMPXR | Sysplex (Connect:Direct/Plex) Statistics Record |
| DMQCR | Process Queue Change Statistics Record |
| DMRJTR | Run Job Termination Record |
| DMRTTR | Run Task Termination Record |
| DMSDCR | Start Connect:Direct Statistics Record |
| DMSFR | Signon/Signoff Statistics Record |
| DMSIGNDB | Secure+ Statistics Record |
| DMSSCR | Session Begin Record |
| DMSTDCR | Stop Connect:Direct Statistics Record |
| DMSTEPR | Step Start/Copy Start Statistics Record |
| DMSYR | SYSOPTS Record |
| DMTCQGT | GOTO Statement |
| DMTCQIF | If Statement |
| DMTCQNUL | NULL Statement |
| DMTPR | Throughput Record Statistics |

| Macro Name | Description |
|---|---|
| DMTRP | Trap Event Record |
| DMTXR | TCQ Threshold Warning<br>TCQ Threshold Full<br>TCQ Threshold Low |
| DMWRPST | Statistics File Wrap Record |
| DMXOR | TRACE On/Off Statistics Record |

# Submit Exit

The Submit exit provides an interface to a user-written program when you submit a Connect:Direct Process. With this interface, the user program can change Process information, such as Process name, priority, class, and secondary node, and copy step information such as data set name.

## Sample Submit Exits

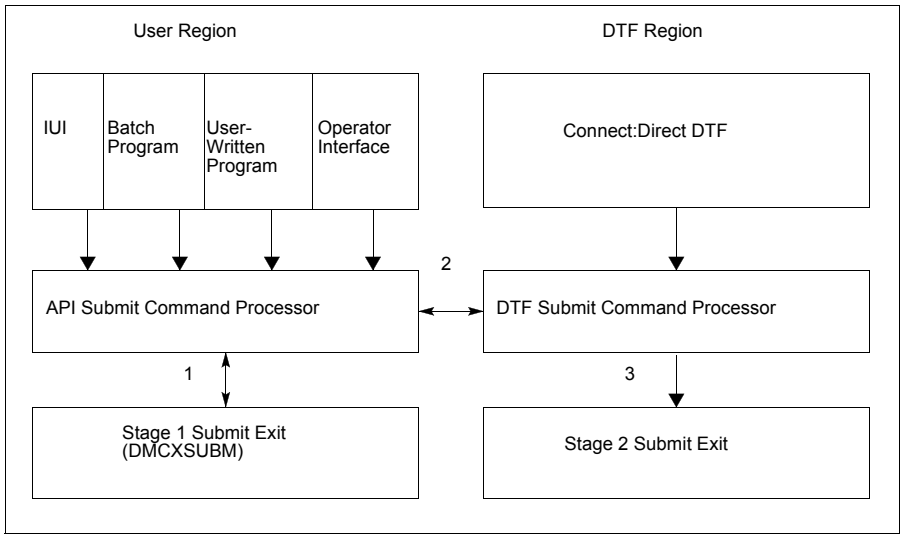
Connect:Direct provides the following sample submit exits in SAMPLIB.

| Exits | Description |
|---|---|
| SUBCLASS | This exit sets a 0 return code to pass back to the calling program. It also locates and increments the Process class by 1. It runs either as a stage 1 or stage 2 exit. |
| SUBDSN | This exit documents the control block path to locate the source and destination data set names specified in the Process. It sets a 0 return code to pass back to the calling program. It also locates and increments the Process class by 1. It runs either as a stage 1 or stage 2 exit. |
| SUBMEXIT | This exit sets a return code of 0 to pass back to the calling program. It runs either as a stage 1 or stage 2 exit. |
| SUBPNDID | This exit determines if the submitted Process contains a RUN JOB statement. If the Process contains a RUN JOB statement and the exit encounters a stage 1 password, the exit returns an error. You must then provide either a PNODEID or an SNODEID, depending on which node the Process is executing. Use the PNODEID or SNODEID to validate the RUN JOB submitted job stream through the **user=uid, password=pwd** parameter built by the RUN JOB exit. |
| SUBSORR | This exit documents the control block path to determine whether the Process is performing a send or receive. It sets a return code of 0 to pass back to the calling program. It runs either as a stage 1 or stage 2 exit. |

| Exits | Description |
|---|---|
| SUBXACCT | This exit is a Stage 1 SUBMIT exit example that shows how to update the Primary and Secondary Node accounting (PACCT and SACCT) information in the submitted Process.<br>**Note:** To update accounting information, you must also update your SUBMIT exit. |

## Submit Exit Processing Flow

The following figure illustrates the execution order of the Connect:Direct SUBMIT command.




The Submit exit processing flow is:

1. After you issue a Connect:Direct SUBMIT command or SUBMIT statement, 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 SAMPLIB, called SUBMEXIT, which you can use as a model for either the stage 1 (DMCXSUBM) or stage 2 (SUBMIT.EXIT = modname) exit. In most cases, you only need to run the stage 1 Submit exit.

## Stage 1 Submit Exit

The stage 1 Submit exit 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. Observe the following restrictions and requirements:

- The Connect:Direct stage 1 Submit exit is implemented as an executable load module.
- You must name the load module DMCXSUBM.

- You must link-edit the module as NORENT and NOREUS.
- You must link-edit the module with an authorization code of **1**.
- The module must come from an authorized library.
- For the TSO IUI, the module must come from a library in the LNKLST or ISPLLIB. Do not put the module in a STEPLIB. The only time a STEPLIB works under ISPF is when ISPLLIB is not allocated.
- For DMBATCH and DMCHLAPI, retrieve the module from a library in the LNKLST or STEPLIB.
- Because information passed to the exit by Connect:Direct is located above the 16 megabyte line, you must link-edit the module with AMODE ANY to make it capable of executing in 31-bit mode.

## Stage 2 Submit Exit

The stage 2 Submit exit control point executes in the DTF address space when a SUBMIT command or a SUBMIT statement is encountered. Observe the following restrictions and requirements:

- The stage 2 Submit exit is implemented as an executable load module.
- The name of the load module is user-defined, but cannot conflict with any Connect:Direct load module names.
- Activate the stage 2 Submit exit by specifying SUBMIT.EXIT=(modname) in the Connect:Direct initialization parameters.
- You must link-edit the module as re-entrant and place it in a load library that the Connect:Direct DTF can access.
- The module must come from an authorized library.
- Because information passed to the exit by Connect:Direct is located above the 16 megabyte line, you must link-edit the module with AMODE ANY to make it capable of executing in 31-bit mode.

## Control Block Format

Because Submit exits are invoked before a Process is actually submitted, some control block fields may not be filled in yet. This section presents the control blocks used with all Process statements and NETMAP entries.

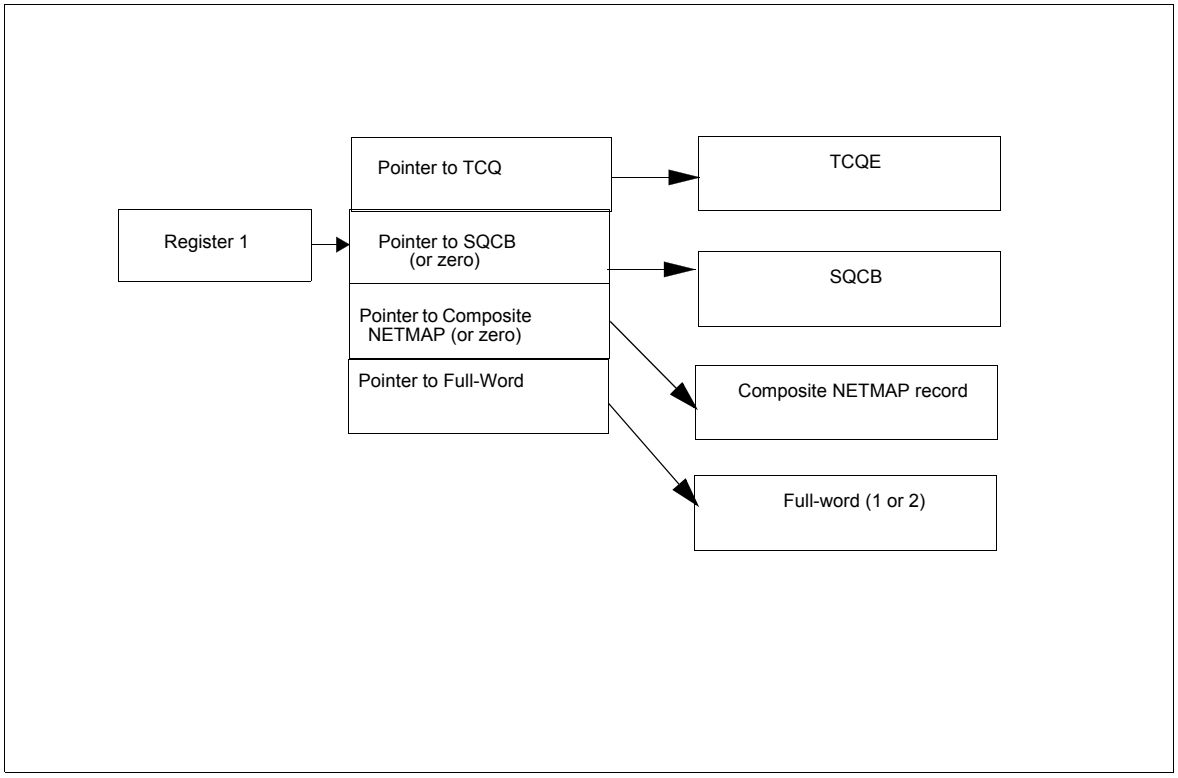
Upon entry to the Submit exit, Register 1 points to a parameter list (PLIST) as shown in the following figure. This list contains the addresses of the following components:

- TCQE
- SQCB
- Composite NETMAP record
- A full-word that contains 1 or 2 indicating whether this is a Stage 1 or Stage 2 exit

---

**Note:** The SQCB and composite NETMAP record addresses may be zero.

---




If the Process submission is rejected, the exit must set a positive non-zero value in Register 15 and the return code field (TQRTNCD) of the TCQE. A message ID in the TQMSGID field in the TCQE should also be set.

The following figure illustrates the layout of the TCQE. In the Process contained in the Statement Control Block, DMxxxxxx represents the macro name for the statement (COPY, RUN JOB, RUN TASK, SUBMIT, etc.).

| | |
|---|---|
| TCQ Header | macro= DMTCQE<br>DSECT= TCQE |
| Command Statement Header Section | macro= DMTCQSH<br>DSECT= TCQSH |
| Statement Control Block | macro= DMxxxxxx |

Displacement values found in the TCQE and pointers in the TCQSH to the next or previous TCQSH are from the top of the TCQE and may need to be multiplied by 16 (if the Process is larger than 64K). Displacement values found in the statement control blocks are from the top of the TCQSH associated with that statement control block.

The following figure illustrates the layout of the composite NETMAP record.

| | |
|---|---|
| COMPOSITE NETMAP RECORD<br><br>$$NN - displacement to ADJACENT NODE Record<br>$$AA - displacement to ALT.COMM Record<br>$$BA - displacement to BATCH.APPLI Record<br>$$CA - displacement to CICS.APPLID<br>$$DN - displacement to LDNS Record<br>$$NA - displacement to APPLIDS Record<br>$$ND - displacement to LUPOOL Record<br>$$NT - displacement to TCP.API Record<br>$$TA - displacement to TSO.APPLIDS Record<br>$$N6 - displacement to IPv6 Record<br>$$NU - displacement to UDT Record<br>$$CM - displacement to COMMENT 13 Record | macro= DMNETFLE<br>DSECT= $$REC |
| ADJACENT NODE Record | macro= DMNETNDE<br>DSECT= NNODEREC |
| ALT.COMM Record | macro= DMNETALT<br>DSECT= ALTADDRH |
| LDNS Record | macro= DMNETDNS<br>DSECT= DNREC |
| APPLIDS Record | macro= DNNETAPL<br>DSECT= NAAPLREC |
| LUPOOL Record | macro= DMNETDDM<br>DSECT= NDLUPREC |
| TCP.API Record | macro= DMNETTCP<br>DSECT= NTAPIREC |
| IPv6 Record | macro= DMNETIP6<br>DSECT= N6TCPREC |

Displacement values found in the composite network map record are from the top of the composite network map.

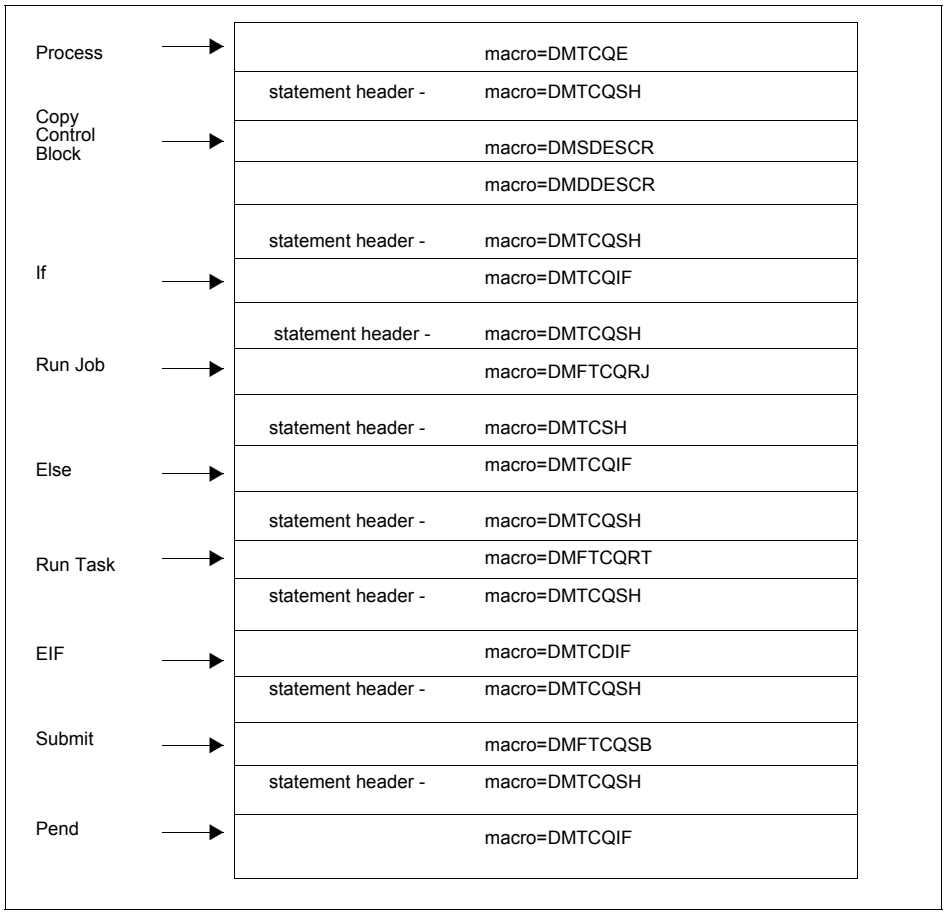
**Note:** Modifying fields in the composite network map record is prohibited.

## Example of Created Control Block

This sample section shows how a Process is submitted and the control block that is created when the Submit exit is invoked. The following figure shows the submitted Process.

```
TEST01 PROCESS   SNODE=THERE
STEP01 COPY FROM (DSN=THIS.DATA.SET) -
    TO   (DSN=THAT.DATA.SET DISP=OLD)
IF01 IF  (STEP01=0) THEN
STEP02 RUN JOB    (DSN=Z99.CONTROL(RUNJ))PNODE
  ELSE
STEP03 RUN TASK -
  (PGM=RTEXAMPL,PARM=(CL44'THIS.DATA.SET')) -
  PNODE
  EIF
STEP04 SUBMIT DSN=Z99.PROCLIB(TEST02) HOLD=Y
```

The following figure illustrates the resulting layout of the Process control block after submitting the Process named TEST01.



## Modifiable TCQE Fields

The following table describes TCQE fields that you can examine or modify using the Submit exit.

| TCQE Field | Content |
|---|---|
| TQCBHLNG | contains the length of the entire TCQE header. 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 user ID for the submitter of this Process. |
| TQUPAS | contains the password for the submitter of this Process. |
| TQPUID | contains the security user ID at the primary node. |
| TQOPPAS | contains the old security password at the primary node. |
| TQNPPAS | contains the new security password at the primary node. |
| TQSUID | contains the security user ID 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 changes this field when an error is encountered in the exit or if the Process is no longer submitted upon return from the exit. |
| TQMSGID | contains the Process message ID. The user exit includes 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.<br>**Note:** If TQGT64K in TQFLAGA is 1, this displacement must be multiplied by 16. |
| TQPARSES | contains the value of the maximum number of parallel sessions allowed for the SNODE when submitting a Process. |
| 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 submitted Process. |
| TQSCHDTE | contains the Julian date the Process is scheduled to submit. |
| TQSCHTME | contains the time of day the Process is scheduled to submit. |
| TQSCHDAY | contains the day of the week that the Process is scheduled to submit. |
| TQPRTY | contains the priority for Process selection. |
| TQRETAIN | contains the retain status for the Process. |

| TCQE Field | Content |
|---|---|
| TQTODFLG | contains the following interval control flags:<br>- If TQTODTD is on, a Process has a scheduled time and date it is submitted.<br>- If TQTOTME is on, a Process has a scheduled time it is submitted.<br>- If TQTODDAY is on, a Process has a scheduled day of the week it is submitted.<br>- 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. |

An exception to the table entry TQPARSES occurs in the stage 1 Submit exit. The stage 1 exit runs in the user address space (API) and the network map associated with that address space is where this information is retrieved. The network map used by the API may not be the same network map used by the DTF. The stage 2 Submit exit runs in the DTF address space and is invoked for every submit that takes place; therefore, the stage 2 Submit exit is more reliable.

## Conversion of Parallel Session Values

The session class value is stored in two bytes (TQPRSBYT and TQPRSBIT) in the TCQE. The specified class can be derived from these values. The following table shows a sample of the two bytes for the first 16 classes (maximum is 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 |

| TQPRSBYT | TQPRSBIT | CLASS |
|---|---|---|
| 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 invokes the exit prior to any allocation activity by the receiving Connect:Direct.

Through the exit, you can change information that Connect:Direct uses 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.

## Sample Allocation Exits

Connect:Direct provides the following sample allocation exits in the SAMPLIB.

| Exits | Description |
|---|---|
| ALLOCDSN | This exit documents the path of the data set name and unit that receives data during a COPY. It runs just prior to data set allocation. It is invoked for every COPY step on the receiving end of a transfer. |
| AXUNIQ | This exit creates a unique z/OS PDS member name, if the data set name specified in the COPY TO statement is found in the PDS directory.<br>Each request can only specify one member name. The COPY TO statement must specify the member name. The COPY TO statement must also specify **SYSOPTS=”UNIQUE=YES”**.<br>The exit only supports copying sequential files to a PDS member. |
| DMALOGDX | This exit enables a Data exit to be invoked for any or every copy performed by Connect:Direct. |
| DMGA3390 | This exit converts UNIT=3390 to UNIT=SYSDA. |
| DMGALOEX | This exit shows how to access the VSAMPL and the TCQSH, and both the source and destination description in the TCQSH. 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. |

| Exits | Description |
|---|---|
| DMGALRCL | In a COPY Process, this exit checks if a data set is archived or migrated. If the data set is archived, the exit requests retrieval and tells the COPY Process to go to the Timer Retry queue (TI RE), from which it is retried based on the ALLOC.WAIT and ALLOC.RETRIES initialization parameters. To ensure that the serially reusable operating system SYSZTIOT resource is freed up for subsequent users and does not cause the Connect:Direct region to hang in a wait condition, use the DMGALRCL exit and specify ARCH as the value for the ALLOC.CODES initialization parameter. Change the setting of the INVOKE.ALLOC.EXIT global initialization parameter to BOTH if using this exit. If running with the CA-DMS product, you must modify the sample exit code as described in the comments at the beginning of the sample code. If you use this exit, add the following DD Statement to the DTF JCL:<br>`//READER DD SYSOUT=(A,INTRDR)`<br>**Note:** This exit is not necessary to process migrated or archived data sets. Connect:Direct processes recalled data sets synchronously and the COPY Process remains in the Execution queue instead of being diverted to the Timer Retry queue and potentially to the Hold queue should retry limits be exceeded.<br>In releases prior to Version 4.7, when the DMGALRCL allocation exit attempted to recall a migrated or archived data set offline, the Process went into fail state and was taken out of the Execution queue and put into the Timer Retry (TI RE) queue. As Connect:Direct waited for the allocation and recall to be performed asynchronously, it would retry the Process based on the ALLOC.WAIT and ALLOC.RETRIES initialization parameters. If the Process exceeded the maximum time limit specified for retrying it (number of retries as specified by the ALLOC.RETRIES initialization parameter multiplied by the amount of time that Connect:Direct waits between retries as specified by the ALLOC.WAIT initialization parameter), the Process was put into the Hold queue requiring manual intervention.<br>In Version 4.7 and later, when Connect:Direct executes a COPY Process without the DMGALRCL exit being present, it will use the ARCHRCAL macro synchronously, which means that the Process stays in the Execution queue not having to loop between the Execution, Timer, and Hold queues. The Process does not terminate while the recall operation is being performed but if the recall is unsuccessful, the COPY step produces a return code indicating the unsuccessful data set recall and instructs the user to correct the error and resubmit the Process.<br>Although it is not needed for data recall, the DMGALRCL exit will continue to be supported and invoked prior to any allocation activity if the ALLOCATION.EXIT=DMGALRCL initialization parameter is specified. However, this exit is no longer required to process migrated or archived data sets.<br>***Caution***: If the sample exit is not being used and a migration/recall product is not active, installed, or is temporarily down, the following messages are displayed:<br>`ARC0050A DFSMSHSM IS NOT ACTIVE - START DFSMSHSM`<br>`ARC0051A JOB xxxxxxxx WAITING FOR DFSMSHSM TO RECALL DSN=dsname`<br>`*73 ARC0055A REPLY 'GO' OR 'CANCEL'`<br>To proceed with the allocation, you must reply. Please refer to IBM documentation regarding the ARCxxxxA messages. If you reply CANCEL to the ARC0055A message, the Process completes with MSGID=SDE021CI. If 21C is in the ALLOC.CODES= list, the Process retries; otherwise, the Process terminates. |
| DMPALLOC | This exit provides %PNUM substitution in a Process. |

## Restrictions and Requirements

Observe the following restrictions and requirements:

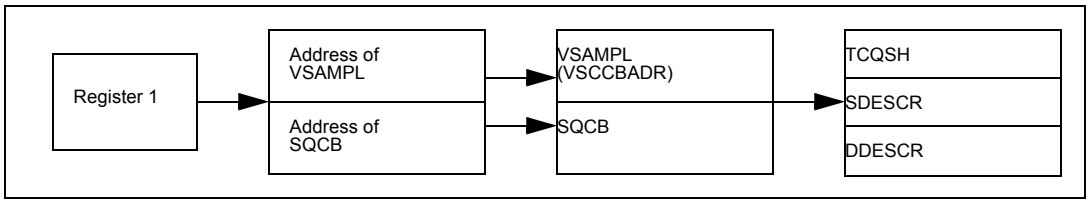
- The name of the allocation exit load module is user-defined, but it must not conflict with any other Connect:Direct load module names.
- Because the control blocks provided by Connect:Direct that the exit must access are located in storage requiring 31-bit addressability, you must link-edit the module with AMODE ANY to make it capable of executing in 31-bit mode.
- To activate the exit, specify ALLOCATION.EXIT=modname in the Connect:Direct initialization parameters file. You must link-edit the allocation exit as re-entrant and place it in a load library that the Connect:Direct DTF can access.
- If an exit is not working, check the setting of the INVOKE.ALLOC.EXIT global initialization parameter. The default setting is RECV, which invokes the allocation exit upon receiving a file. If you are implementing an exit that should be invoked when a file is sent, change the setting to SEND, and to invoke the exit when both sending and receiving a file, change the setting to BOTH.

## 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 initiated the copy is not in restart mode.

The following figure illustrates the structure of the parameter list for the allocation exit.




The following table is a list of the allocation exit parameters.

| Parameter | Explanation |
|---|---|
| R1 | Register 1 that contains the address of a standard parameter list upon entry into the user-written allocation exit |
| PLIST | Stands for standard parameter list |
| SQCB | Security control block |
| VSAMPL | Stands for VSAM parameter list, whose address is the first full word in the PLIST |
| VSCCBADR | Address of the Process step header and is contained in VSAMPL |
| TCQSH | Process step header portion of the Copy control block (each step of a Process generates a TCQSH) |

| Parameter | Explanation |
|---|---|
| SDESCR | Source data set descriptor portion of the Copy control block |
| DDESCR | Destination data set descriptor portion of the Copy Control Block (a sample of the DDESCR Control Block is included in this section) |

The following macros map the control block structures.

| Macro | Explanation |
|---|---|
| DMVSAMPL | Macro that defines the VSAMPL control block |
| DMTCQSH | Macro that defines the TCQSH portion of the Copy control block |
| DMSDESCR | Macro that defines the SDESCR portion of the Copy control block |
| DMDDESCR | 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 entry in the PLIST contains the address of the VSAMPL. The second entry in the PLIST contains the address of the SQCB, if available. The VSCCBADR field in the VSAMPL contains the address of the Process step header, TCQSH. The SDESCR and DDESCR following the TCQSH is 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. The allocation exit modifies any fixed length field in DDESCR.

The fields that are referenced in SDESCR and DDESCR using displacement values are variable in length. Do **not** modify them with the allocation exit.

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 lack of expansion room means that this copy originated from a Connect:Direct node that did not build the copy control block with an expandable DSN field.

You can find the DSN field by adding D1DDSN to the address of the TCQSH. The DSN field contains a 2-byte length field followed by the DSN. Even though the field can be up to 100 bytes long, the 2-byte length field contains the actual length of the DSN. If you change the length of the

DSN, you must modify the 2-byte length field accordingly. 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 overrides the corresponding Type fields from the Type record.

## Copy Control Block Definitions

Copy control block definitions are generated in the allocation exit program by including the macro name followed by DSECT=YES, as follows:

- DMVSAMPL DSECT=YES
- DMTCQSH DSECT=YES
- DMDDESCR DSECT=YES
- DMSDESCR DSECT=YES

## Copy Control Block Modifications

Modify only the VSAMPL and DDESCR control blocks. For the VSAMPL control block, you are required to modify the VSRTNCD and VSMSGID fields as follows:

- Set the VSRTNCD field (and R15) to 0 to allow the Copy step to execute.
- Set the VSRTNCD field (and R15) to a nonzero value to fail the Copy step.
- Insert a message ID into the VSMSGID field, if the VSRTNCD is set to a nonzero value. Take precaution to avoid duplicating existing message IDs.
- Place the 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 DMDDESCR member of the SAMPLIB lists the DDESCR control block format. You can modify the control block fields listed in the following table. Do NOT modify any displacement fields listed in the member.

**Note:** Turning a flag *on* means setting the bit in the byte where the flag is located to one. Turning a flag *off* means setting the bit in the byte where the flag is located to a zero.

**Note:** 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) Connect:Direct can change the actual values. At the calculated address, you will find a halfword field representing the length of the data that follows. If you change the length of the data, you must also change this halfword 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 on the D1MEMNAM flag 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.

| Field | Explanation | Use Instructions |
|---|---|---|
| D1DTYPE | entry in the Connect:Direct type defaults file | N/A |
| D1BLKSIZ | 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 | 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 | 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 | 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 | 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 | relative key position | To use this value, set the D1RKYP flag on. |
| D1TRTCH | 7-track recording mode | To use this value, set the D1TRKTCH flag on. |
| D1LABTYP | label type | To use this value, set the D1LABEL flag on. |
| D1RETPD | 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 | retention period | To use this value, set the D1EXPDTE flag on. |
| D1PRILOC | 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. |

| Field | Explanation | Use Instructions |
|---|---|---|
| D1SECLOC | secondary allocation amount | To use this value, set the D1SECALL flag on. Set all of the following bits to off to use the space allocation values specified in the type record: D1TRK, D1CYL, D1BLK. |
| D1DIRBLK | number of directory blocks | To use this value, set the D1DIRBLK flag on. Set all of the following bits to off to use the space allocation values specified in the type record: D1TRK, D1CYL, D1BLK. |
| D1UNIT | 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. Set all of the following bits to off to use the UNIT value specified in the type record: D1UNCNT, D1P, D1DEFER, D1GRPTYP. |
| D1VOLSEQ | volume sequence number | To use this value, set the D1VOLSQ flag on. |
| D1VOLCT | volume count | To use this value, set the D1VOLCNT flag on. |
| D1PASSWD | data set password | To use this value, set the D1PWD flag on. |
| D1DMEMB | displacement to the member name field† | N/A |
| D1DDSN | displacement to the data set name field† | N/A |
| D1DVOLN | displacement to the volume serial number† | N/A |

# I/O Exits

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 whose organization Connect:Direct does not support or would improperly access. Examples are internal format access to CA-LIBRARIAN or CA-PANVALET files.

**Note:** Checkpoint/restart is not supported for I/O exits.

## Sample I/O Exit

The following sample I/O exit is provided in SAMPLIB.

| Exits | Description |
|---|---|
| NDMIOX01 | This exit processes external data sets whose formats are not supported by Connect:Direct for z/OS. It allocates, opens, reads, closes and deallocates a sequential data set. |

**Note:** Connect:Direct provides another I/O exit, DMDSSIOX, which enables you to copy SMS-compressed data without having to decompress the data and also provides support for copying wildcard-named files. However, you cannot modify this I/O exit. For more information, see *Utility Programs* in *Connect:Direct for z/OS User's Guide*.

## Implementing the I/O Exit

If you plan to use an I/O exit, consider the following items:

- All I/O exits must be re-entrant, follow IBM Assembler linkage standards, and reside in an authorized load library on the node where they are referenced. These exits must not alter any Connect:Direct control block fields (except in the EXTCB as indicated in *I/O Exit Access to Control Blocks* on page 237). If other Connect:Direct control block fields are altered, the results are unpredictable.
- If an ALLOCATION EXIT is specified, it is not 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. See *Adding Messages to Connect:Direct Message Library* on page 263 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 must 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.

After you write the I/O exit to satisfy your specific data set requirements, 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 in the COPY Statement

Include the IOEXIT keyword on the COPY statement to indicate that an I/O exit is used. The IOEXIT format on the COPY statement follows.

| Statement | Parameters |
|---|---|
| COPY | FROM (<br>IOEXIT= exitname\|<br>(exitname[,parameter,...])<br>) |
| | TO (<br>IOEXIT= exitname\|<br>(exitname[,parameter,...])<br>) |

The subparameters of the IOEXIT parameter of the COPY statement are:

| Parameter | Description |
|---|---|
| exitname | Name of the user-written program to receive control for I/O-related requests. |
| parameter | Parameter or list of parameters passed to the exit. Their format is the same format as those parameters which you can specify on the RUN TASK statement. |

The IOEXIT keyword is valid in either the FROM or TO areas of the COPY statement. This capability enables you to specify a different user-written I/O exit on each side as illustrated in following example.

```
COPY    FROM    (PNODE, -
                IOEXIT=(INEXT01,C'DB0A05',X'0E')) -
        TO      (SNODE, -
                IOEXIT=OUEXT03)
```

If you specify an exit, it can ignore the values of the other parameters in the COPY statement (the DCB information). This issue is beyond the control of Connect:Direct. For complete information on using the I/O exit through the COPY statement, see the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/Documentation/processes/processhome.html.

## Specifying the I/O Exit in the TYPE File

Another method of specifying 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 copy side (source or destination) that references it.

## 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 2-byte length followed by the value.

All parameter lists end with the high order bit on in the last address in the list. The macro DMEXITCB generates the EXTCB. DMEXITCB is supplied in the SAMPLIB.

## 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. The BEGIN is when the exit must allocate work areas in preparation for future requests and is the first request that an I/O exit receives.

**OPEN Request**

Connect:Direct makes an OPEN request to an I/O exit when the exit allocates and opens the file. EXTDIR contains either *S* (Send) or *R* (Received) to indicate whether the file is to be read or written. The I/O exit uses 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 retrieve 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.

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 read into the buffer. EXTINLNG is set to the length of the data. EXTINARA points 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. The EXTRCEOD in EXTRTNCD values enable Connect:Direct to issue messages that are added to the Connect:Direct message file.

The IOEXIT must allocate a buffer for the input record/block. To determine if Connect:Direct is expecting a record or block, the IOEXIT must examine the source LRECL, BLKSIZE, RECFM and destination LRECL, BLKSIZE, RECFM fields in the EXIT control block. If the source and destination data set attributes match, then Connect:Direct is expecting a physical block from the IOEXIT, otherwise a logical record is expected.

**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.

Connect:Direct always sends RECFM=VS and RECFM=VBS files in segments rather than records. If EXTSPAN is set to **Y**, the data pointed to by EXTOTARA has two segments. Refer to the IBM library of manuals for information on the format of spanned records. The buffer referenced by EXTOTARA contains a physical block if the source and destination data set attributes match; otherwise, it contains a logical record.

**CLOSE Request**

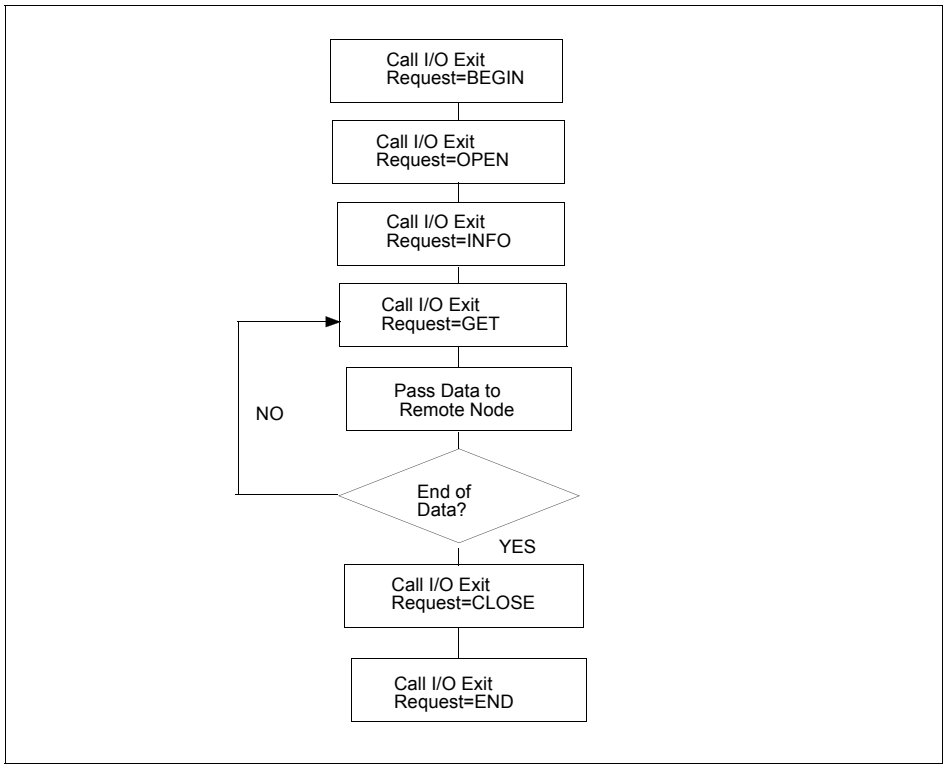
Connect:Direct makes a CLOSE request to an I/O exit when the file closes. Errors returned by the exit on this request are ignored. The EXTABN flag is activated if the CLOSE request is due to abnormal termination.

**END Request**

Connect:Direct makes the END request to an I/O exit to end communication with the exit. The exit releases any work areas allocated when it received the BEGIN request. This request is the last request an I/O exit receives.

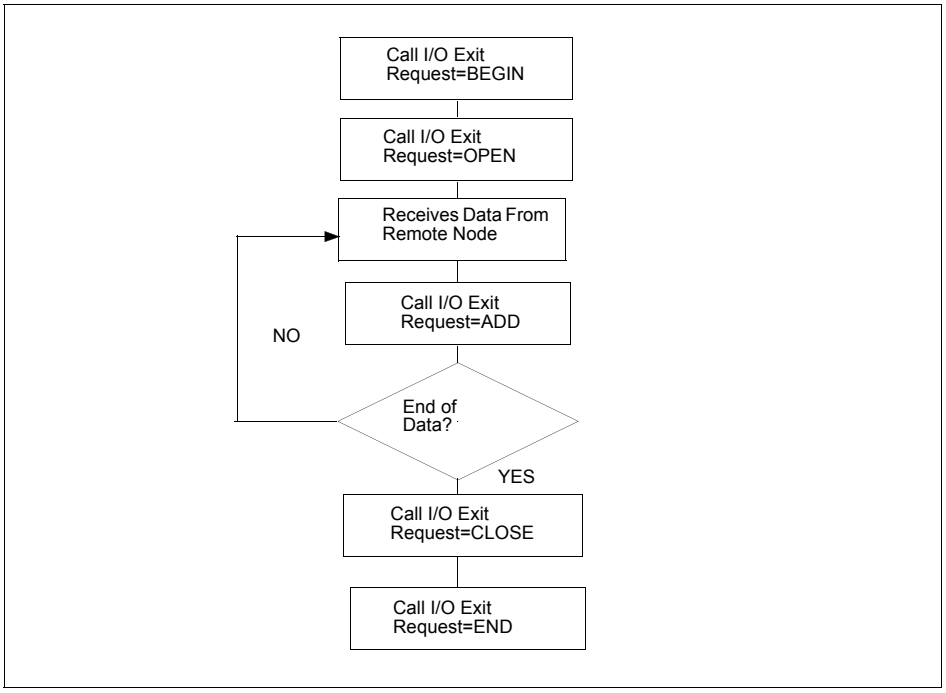
## Normal Input Calling Sequence

The following figure illustrates the normal call sequence for an I/O exit used for input.




## Normal Output Calling Sequence

The following figure illustrates the normal calling sequence of an I/O exit used for output.




# Data Exit

The Data exit functions similarly to the I/O exit; however, the Data exit does not require the I/O management that the I/O exit requires. The Data exit provides an interface to user-written programs, allowing them to add, delete, change, or insert records.

The Data exit is called through the DATAEXIT parameter in the COPY statement or a keyword parameter supplied within the SYSOPTS string.

**Note:** Checkpoint/restart is supported for Data exits.

## DATAEXIT Format

The DATAEXIT format in the COPY statement follows.

| Statement | Parameters |
|---|---|
| COPY | FROM (DATAEXIT= exitname\|(exitname[,parameter,...])) |
| | TO (DATAEXIT= exitname\|(exitname[,parameter,...])) |

The DATAEXIT subparameters are:

| Parameter | Description |
|---|---|
| exitname | The name of the user-written program that receives control for data requests. |
| parameter | A parameter or list of parameters that are passed to the exit. See the RUN TASK statement in the *Connect:Direct for z/OS User's Guide* for parameter formats. |

The following example shows the DATAEXIT parameter in the COPY statement.

```
COPY01   COPY     FROM     (PNODE DSN=GJONES1.FROM.DSN                      -
                             DATAEXIT=(CD$DXX01,CL6'WEEKLY')                -
                              )                                             -
                  TO      (SNODE DSN=GJONES1.TO.DSN                         -
                             DCB=(DSORG=PS,LRECL=80,BLKSIZE=32000)          -
                             DISP=(NEW,DELETE,DELETE)                       -
                             SPACE=(CYL,(1,0,0))                            -
                             DATAEXIT=(CDDATAEX,CL44'GJONES1.TO.DSN')         -
                             UNIT=SYSDA)
```

The following example shows DATAEXIT used as a SYSOPTS parameter.

```
COPY01   COPY     FROM     (PNODE DSN=GJONES1.FROM.DSN                     -
                              SYSOPTS="DATAEXIT=(CD$DXX01,CL6'WEEKLY')"    -
                              )                                            -
                  TO      (SNODE DSN=GJONES1.TO.DSN                        -
                             DCB=(DSORG=PS,LRECL=80,BLKSIZE=32000)         -
                             DISP=(NEW,DELETE,DELETE)                      -
                             SPACE=(CYL,(1,0,0))                           -
                             DATAEXIT=(CDDATAEX,CL44'GJONES1.TO.DSN')        -
                             UNIT=SYSDA)
```

## Sample Data Exits

The following DATAEXIT samples are provided in SAMPLIB.

| Exits | Description |
|---|---|
| CD$DXX01 | This sample Data exit examines or changes records from a COPY Process based on input data. You can use it to insert, replace, or delete records. |
| CD$DXX02 | This sample Data exit converts data from EBCDIC to ASCII if sending a file, or from ASCII to EBCDIC if receiving a file. |

## Implementing the Data Exit

If you plan to use a Data exit, consider the following items:

- All Data exits must be re-entrant, follow IBM Assembler linkage standards, and reside in an authorized load library on the node where they are referenced. These exits must not alter any

Connect:Direct control block fields (except in the EXTCB as indicated in *Data Exit Access to Control Blocks* on page 242). If other Connect:Direct control block fields are altered, the results are unpredictable.

- Add any message IDs specified by a Data exit to the Connect:Direct Message file. See *Adding Messages to Connect:Direct Message Library* on page 263 for instructions.
- Return from the Data exit in the AMODE under which it was called. For example, if the Data exit is called in 31-bit mode, the return must be in that mode. Return from a Data exit through a Branch Set Mode (BSM) instruction rather than a Branch (BR) instruction.

After you write the Data exit to satisfy your specific data set requirements, implement it by specifying the exit name on the DATAEXIT keyword on a COPY statement.

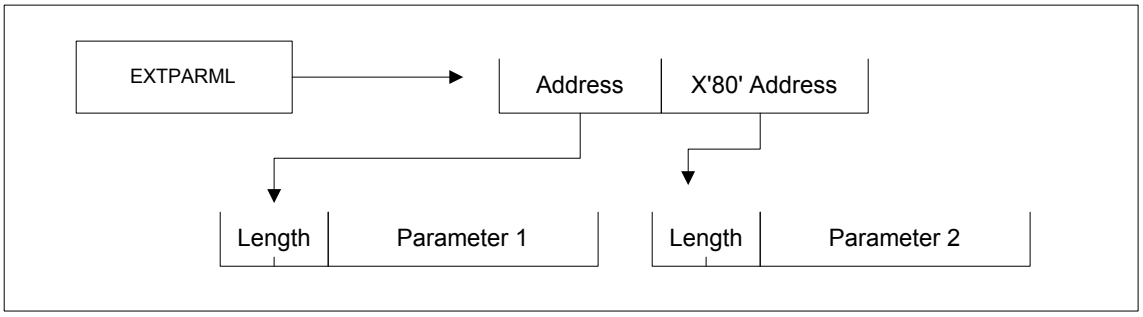
## Data Exit Access to Control Blocks

On entry to the exit, register 1 (R1) contains the address of the pointer to an 8-byte parameter list. The address consists of:

- +Pointer to a 4K storage area that is constant throughout the Data exit step. This area is mapped using the DXPARM label within the DMVSAMPL macro in SAMPLIB.
- +Pointer to the EXTCB (mapped by the DMEXITCB macro in SAMPLIB).

If parameters are passed to the Data exit, the EXTPARML field in EXTCB points to a standard z/OS parameter list, pointing to the parameters (half-word length followed by the parameter itself) passed to the Data exit. If no parameters are passed to the Data exit, EXTPARML points to a full-word field of binary zeros.

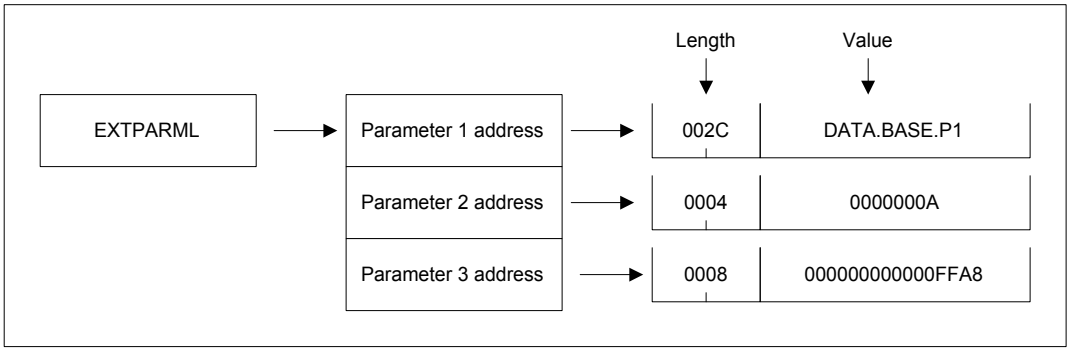
For example, if two parameters are passed to the Data exit, EXTPARML in EXTCB points to two full word pointers (the second pointer will have the high order bit on indicating the last parameter). Each of the pointers point to a half-word length followed by the parameter value as follows.




As another example, assume the following Data exit call.

```
DATAEXIT= = (MYTASK,CL44'DATA.BASE.PI'        -
             F'0010',XL8'FFAB')
```

Based on this call, the information passed to the exit program is displayed as follows.




To adhere to common linkage standards, Connect:Direct sets the list termination bit (X'80') in the Parameter 3 address.

## Data Exit Requests

The Data exit is called with Connect:Direct requests that are found in EXTOPER, which is a field in EXTCB. Input and output Data exits receive the following requests:

**Note:** Upon return from the Data exit, any nonzero value in EXTRTNCD causes the Process to terminate with RC=8 and an SCPA049I message.

**BEGIN Request**
Connect:Direct makes a BEGIN request to a Data exit when it begins communication with the exit. The exit is passed to a 4K work area that remains constant throughout this step in the Process. If additional storage is required, it can be obtained and anchored in the 4K work area.

**OPEN Request**
Connect:Direct makes an OPEN request to a Data exit after the file is open and before the first data record is read.

**GET Request**
Connect:Direct makes a GET request to a Data exit before adding the record to the buffer for transmission to the remote. At this point, the Data exit instructs Connect:Direct to pass the record unchanged, change the record, delete the record, or insert records by setting the appropriate flag bit in EXTEAI in the EXTCB.

| Flag bit setting | Action |
|---|---|
| All bits off | Pass the record unchanged |
| EXTEAIRR | Replace the record. Update EXTINARA to point to the new record and EXTINLNG with the new record length. |
| EXTEAIRD | Delete the record. |

| Flag bit setting | Action |
|---|---|
| EXTEAIRI | Insert a new record. (The next call present the original record again and you can continue to insert records.) If pointing to a new record, set EXTEAIRR. Update EXTINARA to point to the new record and EXTINLNG with the new record length.<br>If records are inserted in place of an original record, the original record delete the original record after the inserted records are passed. |

**PUT Request**

Connect:Direct makes a PUT request to the Data exit before sending the record to the z/OS I/O system (IOS). At this point, the Data exit instructs Connect:Direct to pass the record unchanged, change the record, delete the record, or insert records by setting the appropriate flag bit in EXTEAI in the EXTCB.

| Flag bit setting | Action |
|---|---|
| All bits off | Pass the record unchanged. |
| EXTEAIRR | Replace the record. Update EXTOTARA to point to the new record and EXTOTLNG with the new record length. |
| EXTEAIRD | Delete the record. |
| EXTEAIRI | Insert a new record. (The next call presents the original record again and you can continue to insert records.) If pointing to a new record, set EXTEAIRR. Update EXTOTARA to point to the new record and EXTOTLNG with the new record length.<br>If records are inserted in place of an original record, delete the original record after the inserted records are passed. |

**CLOSE Request**

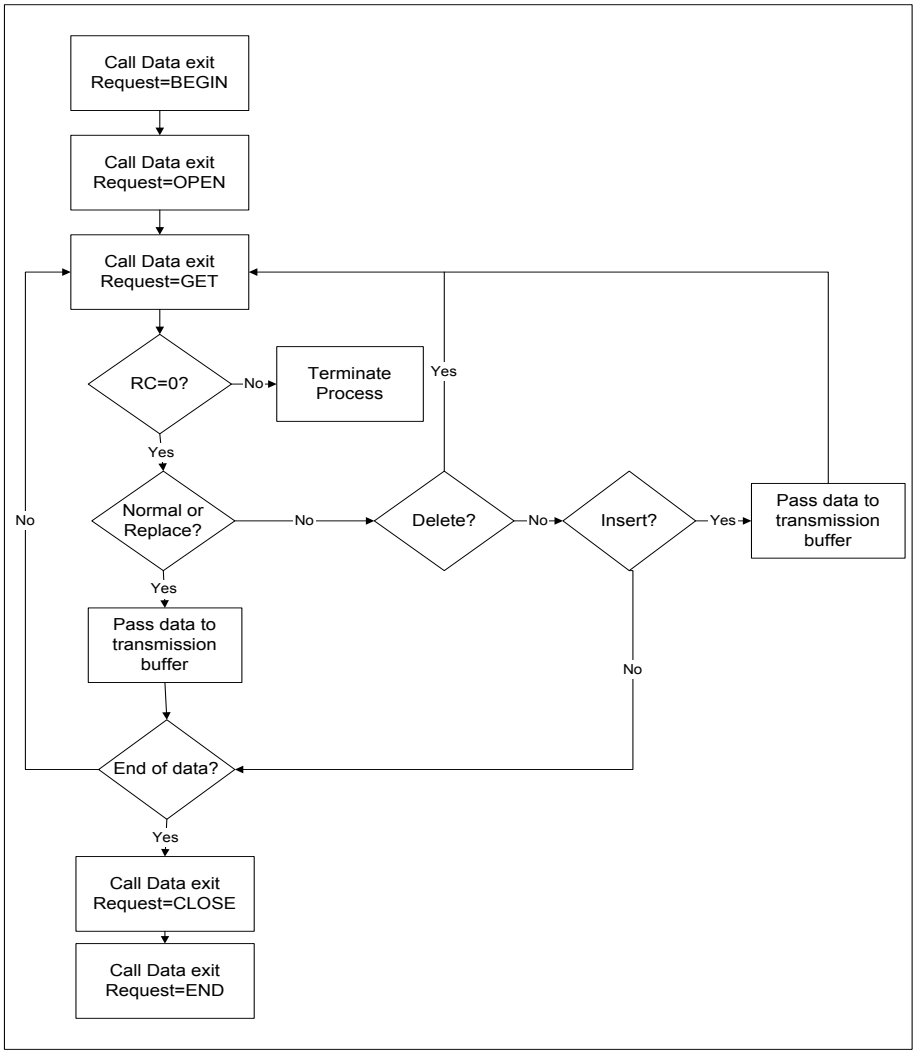
Connect:Direct makes a CLOSE request to a Data exit when the file is to close.

**END Request**

Connect:Direct makes the END request to a Data exit to end communication with the exit. The exit releases any work areas allocated when it received the BEGIN request. This request is the last request a Data exit receives.

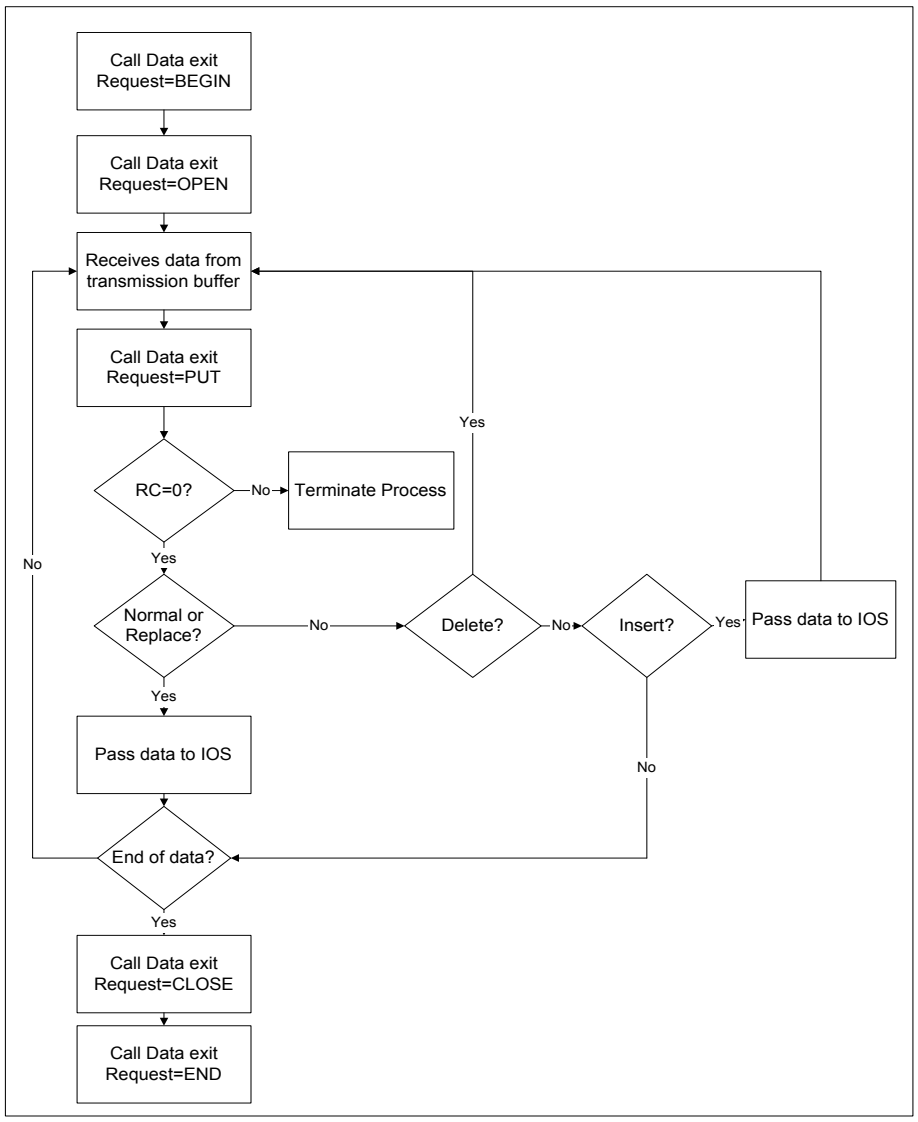
## Normal Input Calling Sequence

The following figure illustrates the normal call sequence for a Data exit used for input.



### Normal Output Calling Sequence

The following figure illustrates the normal calling sequence of a Data exit used for output.




# WLM Exit

If you code the initialization parameter CDPLEX.WLM.GOAL=YES, Connect:Direct invokes the IWMWSYSQ macro when necessary to query the status of systems in a sysplex. Connect:Direct uses the information returned from the query to determine which system can best handle additional Process work. If you want to override the decision Connect:Direct makes, you can specify a different choice with the WLM exit.

Activate the WLM exit by coding the following initialization parameter:

```
CDPLEX.WLM.GOAL=(YES,exitname)
```

Where exitname is the name of the WLM exit.

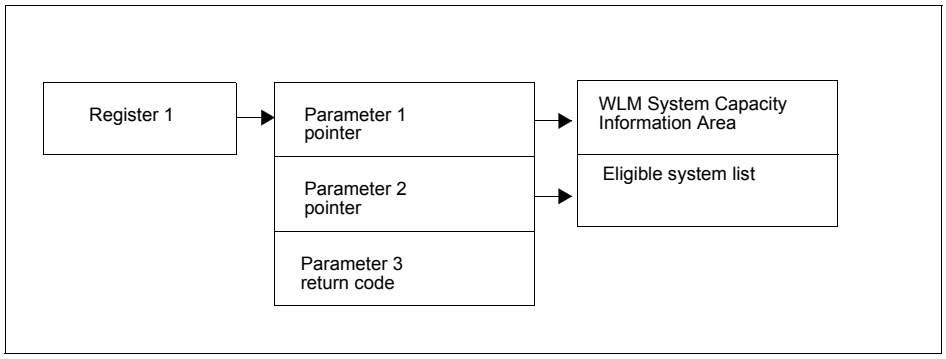
## Exit Calling Convention

Three parameters are passed to the WLM exit by Connect:Direct.

- The first parameter points to the WLM System Capacity Information Area (IWMWSYSI).
- The second parameter points to a list of system names that were found in goal mode by the WLM query. At least one of these systems must have a Connect:Direct/Plex server active.
- The third parameter is used for the WLM exit return code. The following describes the possible return codes:

| Return Code | Description |
|---|---|
| 0 | This indicates all systems are considered equal and no choice is made. |
| Negative value | This indicates that no systems are selected. |
| 1-n | This indicates the chosen system by an index into the eligible system list. |

The following figure depicts the information passed to the WLM exit:




## Sample WLM Exit

The following sample WLM exit is provided in SAMPLIB.

| Exit | Description |
|---|---|
| CDWLMEX | This exit analyzes the IWMWSYSI info returned from an IBM Workload Manager query and indicates the least busy system with a return code. |

# Tapemount Exit

The Connect:Direct tapemount exit provides an interface to StorageTek Tape Silo Software. If you supply a user exit in the initialization parameters, Connect:Direct invokes the exit prior to a tape VOLSER mount request. Using the return codes resulting from this exit, you can obtain the status of the volumes needed to satisfy the mount request prior to the Tape premount message being displayed. (The TAPE.PREMOUNT = YES | NO | LIST parameter determines if a tape premount message will be displayed or not.) If any volume is not available for the Silo to process, the tape mount request is automatically cancelled and an exit return code of 8 or higher is issued to indicate that the Process is being held in error.

| Exit Return Code | Explanation |
|---|---|
| 0 | The Tape Premount message will be suppressed. |
| 4 | The Tape Premount message will be issued. |
| 8 or higher | The Mount Process will be “Held in Error.” |

## Sample Tapemount Exit

Connect:Direct provides the following sample tapemount exit in the $CD.SAMPLIB.

| Exits | Description |
|---|---|
| DMGTAPEX | This exit interfaces with the StorageTek Tape Silo via SLA macro calls. You can modify this member to work with other vendors’ Tape Silo products. |

You can use the sample JCL, ASMTAPEX located in $CD.SAMPLIB, to assemble and link-edit the exit.

## Restrictions and Requirements

Observe the following restrictions and requirements:

- The name of the tapemount exit load module is user-defined, but it must not conflict with any other Connect:Direct load module names.
- Because the control blocks provided by Connect:Direct that the exit must access are located in storage requiring 31-bit addressability, you must link-edit the module with AMODE ANY to make it capable of executing in 31-bit mode.
- To activate the exit, specify TAPEMOUNT.EXIT=modname in the Connect:Direct initialization parameters file (see *TAPEMOUNT.EXIT = modname* on page 420 for more information). You must link-edit the tapemount exit as re-entrant and place in a load library that the Connect:Direct DTF can access.

The TAPEMOUNT.EXIT parameters are:

| Parameter | Explanation |
|---|---|
| NOVOLS | Used to define the return code if all volumes for a tape mount request are not in the silo. Code with a value of 00, 04, or 08. |
| VIRTVOL | Used to tell the exit how to treat a “virtual” tape VOLSER. Code with a value of OKAY or ERROR. |
| TEST | Used to supply diagnostic test messages to a DD statement named to match the assembled program name. Code with a value of YES or NO. |
| UID | Optional local identifier which appears in NDMLOG output along with the PTF maintenance listing. |

# Process Exit for Testing (NDMPRCXT)

The Process Exit for Testing (NDMPRCXT) allows you to perform the following functions:

- Test new applications and customer connections
- Prevent future production work from executing until testing is complete after you have terminated all active production work using the Flush Process command
- Resume regular production work after testing
- Control individual file transfers by application
- Enable and disable individual nodes and applications

While testing is being conducted, only Processes, particularly file transfers, involved with the testing activity are executed. No production data is transferred to applications being tested while at the same time no test data is transferred to production applications.

## Processing Flow of the NDMPRCXT Exit

First you tell NDMPRCXT which Connect:Direct Processes to run and not run by storing your preferences as text records in a parameter table stored as a Partitioned Data Set (PDS) member. You can specify the following criteria for NDMPRCXT to use to find matches for one or more Processes to include (using the “I” command code) or exclude (“X” command code) from execution:

- A partial or full Process name
- A partial or full remote node name
- A partial or full Connect:Direct submitter ID and submitter node combination
- A combination of Process name, remote node name and submitter ID/submitter node, all of which must match

In addition to telling Connect:Direct which Processes to run, you tell the system what to do with the Processes which do not get executed. You can specify the following dispositions for Processes not permitted to run:

- Place the Process in the Hold queue
- Place the Processes in the Timer queue for session retry
- Flush the Processes from the queue

To see different ways the NDMPRCXT exit can be used, see *Sample Test Scenarios* on page 256.

The NDMPRCXT exit is invoked by the Stage 2 Security exit before the security checks for a Connect:Direct Process about to be executed have been performed. For information on how the Stage 2 Security exit is processed, see *Stage 2 Security Exit* on page 58.

The Process Exit for Testing reads and validates the NDMPRCXT parameter table each time it is invoked when a Process is executed. If a syntax or other error occurs, Connect:Direct places the Processes in the hold queue and returns a non-zero return code and error message ID. If the table is valid, NDMPRCXT scans the parameter table looking for a pattern that matches the Process that is about to be executed. If a match is found, the Process is permitted to execute if the "I" (Include) command code is in effect. If command code "X" (Exclude) is in effect, the process is not permitted to execute. If a match is not found in the table, NDMPRCXT performs the opposite processing from the case where a match is found, that is, if no match is found and command code "I" is in effect, the Process is not permitted to execute, whereas if command code "X" is in effect, the Process is permitted to execute.

**Note:** To reverse an action taken, use the “R” (Reverse) command code. If a match is found in an Include list and the “R” command code is also in effect, the Process is excluded. Conversely, if a match is found in an Exclude list and the “R” command code is also in effect, the Process is included.

If a Process is not to be permitted to execute, NDMPRCXT uses the disposition specified in the NDMPRCXT parameter table to either hold, retry, or flush the Process after the NDMPRCXT exit returns with a non-zero return code.

**Note:** For Processes initiated on remote nodes, the NDMPRCXT exit functions in the same manner as it does for Processes submitted on the local Connect:Direct node. The NDMPRCXT Parameter Table is searched for a matching entry and the remotely initiated Process is either permitted to execute or excluded from execution. However, because the local node is the SNODE for this type of transfer, it cannot enforce the Process disposition setting in the NDMPRCXT Parameter Table. The remote PNODE determines how the Process is handled. Typically, the remote node places the Process in the Hold queue with a status of "HE" (Held in Error) if SECURITY.NOTIFY=HOLD is specified or the NDMPRCXT exit is supported on the remote node. If the remote node does not support SECURITY.NOTIFY=HOLD or the NDMPRCXT exit, the Process terminates.

When both the PNODE and the SNODE invoke the NMDPRCXT exit and the SNODE excludes a Process from executing, the PNODE automatically places the Process in the Timer queue for session retry regardless of the disposition setting in the NDMPRCXT Parameter Table on the PNODE. This processing is necessary because of technicalities in the handling of an SNODE error at the point in the Connect:Direct protocol.

## Setting Up and Using the NDMPRCXT Exit

To set up and use the NDMPRCXT Exit, complete the following steps. This roadmap is a high-level view of the procedure. For more information on each step, go to the section referenced in that step.

1. Review Chapter 3, *Implementing Security*. This chapter discusses a variety of topics and issues related to security including interfaces to other security software and sample exits.
2. Assemble the NDMPRCXT member in SAMPLIB.
3. To activate the stage 2 security exit which invokes the NDMPRCXT exit, specify SECURITY.EXIT=modname in the Connect:Direct initialization parameters file (see *SECURITY.EXIT = (module name, DATASET | ALL, PSTKT) | OFF SECURITY = (module name, DATASET | ALL, PSTKT) | OFF* on page 410 for more information). You can modify the sample security exit, DMGSAF, provided in the SAMPLIB, to use as the base code for your stage 2 security exit.
4. Change the PROCEXIT parameter in the DMGSECUR macro to NDMPRCXT. See *Enabling the NMDPRCXT Exit* on page 251.
5. Create a PDS member to store parameters specifying how you want to implement the NDMPRCXT Exit, that is, preferences such as which Processes to run and not run and which queue to place unexecuted Processes. See *Preparing the NDMPRXCT Parameter Table* on page 252.
6. Add DD statements to allocate the NDMPRCXT table and log information. See *Adding DD Statements* on page 252.
7. Reassemble and link-edit the Stage 2 exit source code. Because the control blocks provided by Connect:Direct that the exit must access are located in storage requiring 31-bit addressability, you must link-edit the module with AMODE 31 to make it capable of executing in 31-bit mode. You must also link-edit the stage 2 security exit as re-entrant and place in an authorized library that the Connect:Direct DTF can access.
8. Submit the startup jobstream to start Connect:Direct. (For more information on starting Connect:Direct, see the chapter on installing Connect:Direct for z/OS in *Connect:Direct for z/OS Installation Guide*.)

---

**Note:** It is not necessary to restart Connect:Direct when you modify the NDMPRCXT parameter table. The new settings are automatically in effect the next time a Process begins executing and invokes the NDMPRCXT exit, which reads the new values in the table. See *Reusing the NDMPRCXT Exit* on page 258 for more information.

---

### Enabling the NMDPRCXT Exit

The Stage 2 Security Exit executes the NDMPRCXT exit before it performs the necessary security checks for a Connect:Direct Process about to be executed. To enable the Security Exit to invoke the NDMPRCXT exit, you must change the PROCEXIT parameter in the DMGSECUR macro from NO to NDMPRCXT, and reassemble the exit. See *Parameters* on page 59 for more information on this and all parameters in the DMGSECUR macro.

### Adding DD Statements

Make sure your startup JCL includes the following DD statements:

```
//NDMPXTBL DD DSN=$CDPREF..PRCXTLIB(&NDMPXMEM),DISP=SHR
//USRINFO DD SYSOUT=*
```

The first DD statement allocates the parameter table PDS member while the second DD statement allocates a SYSOUT data set to which user-defined information from User Exits, such as error messages for parameter records incorrectly formatted and matching entries for Processes which run, is logged.

The CONNECTX JCL startup member contains these DD statements. For more information on the startup JCL members, see *DD Statements in Startup JCL* on page 373.

### Preparing the NDMPRXCT Parameter Table

You can use the ISPF text editor to create the NDMPRXCT Parameter Table which defines which Connect:Direct Processes can and cannot run. This table is stored as a Partitioned Dataset (PDS) member.

You must preallocate the data set with the following attributes:

```
DSORG=PO
LRECL=80
RECFM=FB
BLKSIZE=multiple of 80
```

Each table entry or record consists of a single-character command code in column one. Most command codes have a parameter which begins in column two and varies according to the command code function.

**Note:** The order of the entries in the table is important; the first match stops the table scan and the action requested is taken (allow the process to execute, flush the process, etc.).

| Command Code | Description | Subparameters/ Examples |
|---|---|---|
| * | Comment line. | * Only run the following processes. |

| Command Code | Description | Subparameters/ Examples |
|---|---|---|
| E | Enables NDMPRCXT. This command code must be the first non-comment entry in the table. | The second column in this entry must contain one of the following values which indicates the disposition of a process if it is not allowed to run.<br>H – Places the Process in the Hold queue with a status of HE (Held in Error)<br>R – Places the Process in the Timer queue in session retry until number of retries is exceeded. Once this number is exceeded, the Process is placed in the Hold queue with a status of RH (Restart Hold).<br>F – Flushes the process from the queue |
| D | Disallows NDMPRCXT execution and fails Process execution with a non-zero (error) return code and message NPRX003E. | You can also leave the disposition code in column two to make it easier to change from "E" to "D" and vice versa without having to change column two to a blank for command code “D.” |
| P | Matches Processes based on a full or partial Process name. Supports the wild card trailing asterisk (*). Can be used to enable or disable Process execution for a particular application by using naming conventions to match an application.<br><br>In addition, remote node and/or submitter@submitter-node can be specified to further qualify the match.<br><br>The combined format is: Pprocnam[,remote-node[,submitter@submitter-node]] | PCOPY – Matches a single Process<br><br>PEOM* – Matches all Processes beginning with “EOM” for the End of Month Processing application<br><br>P* – Matches all Processes\<br><br>PCOPY,RNODE,SUE@NODE1 – Matches Process COPY that was submitted by SUE on NODE1 and whose remote node is RNODE.<br><br>PCOPY,,SUE@NODE1 – Matches Process COPY that was submitted by SUE on NODE1.<br><br>PCOPY,RNODE – Matches Process COPY whose remote node is RNODE.<br><br>P*,*,SUE@NODE1 – Matches all Processes submitted by SUE on NODE1. |

| Command Code | Description | Subparameters/ Examples |
|---|---|---|
| R | Reverses the action to be taken on a match.<br><br>If the match is found in an include list, the Process is excluded.<br><br>If the match is found in an exclude list, the Process is included.<br><br>The combined format is:<br>Rprocnam[,remote-node[,submitter@submitter-node]] | The syntax is the same as for the "P" command code.<br><br>In this example, all Processes whose remote node begins with RNODE are excluded from execution except those whose remote node is RNODE3.<br><br>X<br>R*,RNODE3<br>NRNODE*<br>L |
| N | Matches Processes based on a full or partial remote node name. Supports the wild card trailing asterisk (*). | NCD.NODE1 – Matches a single remote node name<br><br>NCD.NODEA* – Matches all remote node names beginning with "CD.NODEA"<br><br>N* – Matches all remote node names<br><br>**Note:** P*,CDNODE1 is equivalent to NCDNODE1 and can be specified as R*,CDNODE1 to reverse the action. |

| Command Code | Description | Subparameters/ Examples |
|---|---|---|
| S | Matches Processes based on a full or wild card Connect:Direct submitter ID and a full or partial submitter node combination. The format is <id>@<node>. | SACTQ0ACD@TPM002 – Matches a specific submitter ID and node combination.<br><br>S*@TPM002 – Matches all submitter IDs from node TPM002<br><br>SACTQ0ACD@* – Matches submitter ID ACTQ0ACD from all nodes<br><br>SACTQ0ACD@TPM* - Matches submitter ID ACTQ0ACD from all nodes beginning with "TPM"<br><br>S*@* – Matches all submitter IDs from any node. This is another way to match all Processes.<br><br>**Note:** P*,*,SUE@NODE1 is equivalent to SSUE@NODE and can be specified as R*,*,SUE@NODE1 to reverse the action. |
| I | Includes Processes for execution that match the patterns in the table which follow this command code. Either “I” or “X” must be the second non-comment entry in the table. Processes which do not match a pattern in the table **are not** executed.<br>**Note:** To choose which command code to use to select Processes, determine which group is smaller and use the corresponding command Code. For example, if the number of Processes to be executed is smaller than the number of Processes to exclude from execution, specify “I” as the command code and add patterns to match that group of Processes. | ER<br>I<br>NCD.BOSTON<br><br>Includes for execution only those Processes whose remote node is CD.BOSTON. Excluded Processes are placed in the Timer queue in session retry |
| X | Excludes from execution those Processes that match the patterns in the table which follow this command code. Either “X” or “I” must be the second non-comment entry in the table. Processes which do not match a pattern in the table ***are*** executed. | EH<br>X<br>SDALLASOPS@*<br><br>Excludes Processes for execution submitted by DALLASOPS from any node |
| L | Last entry in table. | |

## Sample Test Scenarios

The following examples show different applications of the NDMPRCXT exit using NDMPRCXT Parameter Tables to define which Connect:Direct Processes to run and not run.

### Specifying Which Processes Run

In this example, Connect:Direct executes all Processes that start with ACH or are named DITEST01 or DITEST02. All other Processes are placed in the Hold queue.

```
* Enable processing. Only permit processes matching one of the patterns
* to execute. Hold processes that don't execute.
EH
I
PACH*
PDITEST01
PDITEST02
L
```

### Specifying Which Processes to Exclude

In this example, Connect:Direct does not execute any Process that starts with ACH or is named DITEST01 or DITEST02. All other Processes are executed.

```
* Exclude matching processes.  Permit all others to execute.
EH
X
PACH*
PDITEST01
PDITEST02
L
```

### Permitting Process Execution by Remote Node and Submitter User ID/Node

In this example, Connect:Direct executes all Processes that match one of the following criteria:

- The remote node name is DI.NODE1
- A remote node whose name starts with DI0017
- Any Connect:Direct submitter ID from node DI0049
- The specific Connect:Direct submitter ID ACHAPP from any node

All Processes not matching one of the above criteria are flushed from the queue.

```
* Only permit matching processes to execute.  Flush those that do not.
EF
I
NDI.NODE1
NDI0017*
S*@DI0049
SACHAPP@*
L
```

### Combining Matching Criteria for a More Specific Match

In this example, Connect:Direct executes all Processes that match one or more of the following criteria:

- Processes that begin with XYZ
- Processes that begin with ABC whose remote node is NODE1

All Processes not matching one of the above criteria are flushed from the queue.

```
* Only permit matching processes to execute.  Flush those that do not.
EF
I
PXYZ*
PABC*,NODE1
L
```

### Using the "R" (Reverse) Matching Criteria

In this example, Connect:Direct performs the following actions:

- Executes all Processes that begin with XYZ
- Executes all Processes that begin with ABC whose remote node is NODE1
- Excludes all Processes that begin with ABC and whose remote node is NODE1 when submitted by JOE@NODE2 from execution

All other Processes are flushed from the queue.

```
* Only permit matching processes to execute.  Flush those that do not.
EF
I
PXYZ*
RABC*,NODE1,JOE@NODE2
PABC*,NODE1
L
```

**Note:** When using the "R" (Reverse) matching criteria, always specify the most specific matching criterion first and the most generic matching criterion last.

### Stopping the NDMPRCXT Exit

In this example, the NDMPRCXT Exit will exclude Processes from being executed, and display a non-zero return code signifying an error along with message ID NPRX003E . The remainder of the table is ignored (including the “F” code to flush Processes from the queue) and all Processes are placed in the Hold queue.

To resume testing and use the NDMPRCXT exit again, change the "D" command code to an "E."

```
* Execute no processes at all.  Put them in the hold queue and return.
DF
I
PACH*
PDITEST01
PDITEST02
L
```

## Reusing the NDMPRCXT Exit

To facilitate the use of different testing scenarios, you can maintain multiple members in the NDMPRCXT Parameter Table PDS. To reuse the basic NDMPRCXT Parameter table but modify it to change the Processes which run and do not run, follow this procedure. It is not necessary to restart Connect:Direct; the next time a Process begins executing, the new settings will be in effect when the NDMPRCXT exit reads the table.

***Caution:*** To control the execution of the NDMPRCXT exit, make sure that only authorized operators can modify the NDMPRCXT Parameter Table PDS member using your security system.

1. Open the NDMPRCXT Parameter Table using the ISPF text editor.
2. Delete the current contents of the table member.
3. Copy another member for the next testing scenario you want to use and modify it as needed.
4. Save the NDMPRCXT Parameter Table PDS member.

   The next time the NDMPRCXT exit is invoked when a Process begins executing, the NDMPRCXT exit uses this updated table.

## NDMPRCXT Output

This section contains example JOBLOG and USRINFO data set output resulting from the execution of the NDMPRCXT exit.

### Example JOBLOG Output

This example shows JOBLOG output when the NDMPRXCT exit is executing.

```
SVTM055I SESSION (001) ESTABLISHED WITH       SNODE=WWW_TCP
NPRX000I ### Permitted:    TSTRUN  (00000007) SNODE=WWW_TCP
SVTM036I PROCESS STARTED   TSTRUN  (       7) SNODE=WWW_TCP
IGD103I SMS ALLOCATED TO DDNAME NDM00027
SVTM052I CO01     COPY     TSTRUN  (       7) SNODE=WWW_TCP
SVTM052I          COMPLETED  00000000/SCPA000I
SVTM052I      FROM /u/output/testfile7
SVTM052I      TO   DALLAS.O.TESTFILE.BENCH.M50
SVTM052I           COMPLETED  00000000/SCPA000I
SVTM037I PROCESS ENDED     TSTRUN  (       7) SNODE=WWW_TCP
SVTM056I SESSION (001) TERMINATED WITH        SNODE=WWW_TCP
SVTM055I SESSION (001) ESTABLISHED WITH       SNODE=WWW_TCP
NPRX109E ### Not executed: XSTRUN  (00000009) SNODE=WWW_TCP
SVTM056I SESSION (000) TERMINATED WITH        PNODE=WWW_TCP
```

### Example USRINFO Output

The examples in this section show USRINFO output that is written to the USRINFO dataset while the NDMPRCXT exit is executing. Each line of output has a timestamp and the hexadecimal address of the Connect:Direct Task Control Area (TCA) under which the NDMPRCXT exit is executing. The TCA address permits correlating output lines when Processes execute concurrently.

The following example shows NDMPRCXT output based on matching the specific Process name, TSTRUN.

```
965630E0  NDMPRCXT V1.14 ENTERED: SQCB@=16987688
965630E0  ############ APPLID TABLE ############
965630E0  EF
965630E0  I
965630E0  PTSTRUN
965630E0  L
965630E0  ############ END APPLID TABLE ############
965630E0  SETTING DISPOSITION TO FLUSH.
965630E0  TABLE MATCH: PNAME,PNUM=TSTRUN  ,00000007
965630E0               ENTRY=PTSTRUN
 NPRX000I ### Permitted:    TSTRUN  (00000007) SNODE=WWW_TCP
965630E0  NDMPRCXT EXITED:  SQCB@/RC/MSGID 16987688 /00000000 /NPRX000I
96561D60  NDMPRCXT V1.14 ENTERED: SQCB@=16990688
96561D60  ############ APPLID TABLE ############
96561D60  EF
96561D60  I
96561D60  PTSTRUN
96561D60  L
96561D60  ############ END APPLID TABLE ############
96561D60  SETTING DISPOSITION TO FLUSH.
96561D60  NO TABLE MATCH: PNAME,PNUM=XSTRUN  ,00000009
 NPRX101E ### No entry:     XSTRUN  (00000009) SNODE=WWW_TCP
96561D60  NDMPRCXT EXITED:  SQCB@/RC/MSGID 16990688 /00000004 /NPRX101E
```

The following example shows NDMPRCXT output based on matching the names of all Processes that begin with TSTRUN using the wildcard *.

```
96562720  NDMPRCXT V1.14 ENTERED: SQCB@=16990688
96562720  ############ APPLID TABLE ############
96562720  EF
96562720  I
96562720  PTSTRUN*
96562720  L
96562720  ############ END APPLID TABLE ############
96562720  SETTING DISPOSITION TO FLUSH.
96562720  TABLE MATCH: PNAME,PNUM=TSTRUN  ,00000010
96562720               ENTRY=PTSTRUN*
 NPRX000I ### Permitted:    TSTRUN  (00000010) SNODE=WWW_TCP
96562720  NDMPRCXT EXITED:  SQCB@/RC/MSGID 16990688 /00000000 /NPRX000I
96561D60  NDMPRCXT V1.14 ENTERED: SQCB@=16990688
96561D60  ############ APPLID TABLE ############
96561D60  EF
96561D60  I
96561D60  PTSTRUN*
96561D60  L
96561D60  ############ END APPLID TABLE ############
96561D60  SETTING DISPOSITION TO FLUSH.
96561D60  TABLE MATCH: PNAME,PNUM=TSTRUN2 ,00000011
96561D60               ENTRY=PTSTRUN*
 NPRX000I ### Permitted:    TSTRUN2 (00000011) SNODE=WWW_TCP
96561D60  NDMPRCXT EXITED:  SQCB@/RC/MSGID 16990688 /00000000 /NPRX000I
```

The following example shows invalid entries in the NDMPRXCT Parameter Table.

```
96562720  NDMPRCXT V1.14 ENTERED: SQCB@=16990688
96562720  ### INVALID TABLE ENTRY:XTSTRUN*
 NPRX005E ### Table format error encountered.
96562720  NDMPRCXT EXITED:  SQCB@/RC/MSGID 16990688 /00000008 /NPRX005E
```

# Special Considerations

The following special considerations apply to Connect:Direct for z/OS exits.

## Avoiding Out-of-Storage ABENDS

To avoid out-of-storage ABENDS in Connect:Direct, examine all user exits to verify that all obtained storage is freed. For each GETMAIN that an exit issues, the exit must issue a corresponding FREEMAIN to avoid accumulating storage. 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:

- Because information passed to the exit by Connect:Direct is located above the 16 megabyte line, you must link-edit the module with AMODE ANY to make it capable of executing in 31-bit mode. Refer to the section in this chapter describing the particular exit to see if this requirement 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. Link exits that run above 16 megabytes in 31-bit mode to AMODE=ANY, RMODE=ANY.
- Verify that your exits are coded to receive control and execute in the AMODE with which they are linked.
- Exits must return control to Connect:Direct in the AMODE in effect when Connect:Direct invokes the exit. Connect:Direct calls your exit through Branch and Save and Set Mode (BASSM), and you 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.

## Linkage Editor Attribute Requirements

You must code all exits that execute in the DTF address space and link-edit them as RENT and REUS. Only the Stage 1 exits (DMCXSIGN and DMCXSUBM) do not require re-entrancy.

# Chapter 11

# Customizing Connect:Direct

This chapter describes how to customize Connect:Direct for z/OS to meet your user needs. This chapter provides information on the following topics:

- Adding Messages to Connect:Direct Message Library
- Defining Message IDs
- 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 in the following figure is in the MSGSOURC member of the Connect:Direct sample library, $CD.SAMPLIB.

**Note:** Use the exact format as follows. You cannot use comments.

```
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 and INSERT variables are 1–8 characters.
- The MODULE variable is 1–64 characters.
- The STEXT and L01 through L12 variables are 1–63 characters.
- To insert a message ID, the INSERT, MODULE, STEXT, and L01 through L12 variables are required.
- To delete a message ID, the DELETE variable is required.
- To replace a message ID, the DELETE, INSERT, MODULE, STEXT, and L01 through L12 variables are required.

## Job Stream to Update Connect:Direct Message Library

The job stream in the following figure is in the MSGLOAD member of the Connect:Direct $CD.JCL library. After copying the message source into your message source library and making changes as needed, run this job stream to add your messages to the Connect:Direct message library.

```
//JOBNAME   JOB (ACCT),'NAME',CLASS=A,NOTIFY=TSOID,MSGCLASS=X
//MSGBUILD  EXEC   PGM=DMMSGLOD,PARM='$CDHLQ.MSGFILE'
//STEPLIB   DD  DSN=$CDHLQ.LINKLIB,DISP=SHR
//SYSOUT    DD  SYSOUT=*
//INPUT DD  DSN=$HLQ.MSGSRC,DISP=SHR
```

**Note:** For SMP/E installations, add the message source to the NMSGSRC target library as an SMP/E USERMOD. MSGLOAD JCL can then process it.

Make the following changes to the job:

- Change the PARM statement to reference the file name of the Connect:Direct VSAM message file. This value is the same value specified in the MSGDSN initialization parameter.
- Change the STEPLIB DD card to reference the LINKLIB.
- Change the INPUT DD card to reference the message source text. A sample format is in the member MSGSOURC in the Connect:Direct sample library, $CD.SAMPLIB.

# Defining Message IDs

In a Connect:Direct/Plex environment, messages can originate from any Connect:Direct/Server or from the Connect:Direct/Manager so you should define a 2-character message ID that identifies which Connect:Direct/Plex member originated the message. This message ID is displayed after the message number.

Define the message ID using the CDPLEX.MSGID initialization parameter, as described on page 437 of this guide.

The following example shows a message if the CDPLEX.MSGID value is set to S1. The message ID is highlighted in bold.

```
SVTM055I S1 SESSION (001) ESTABLISHED WITH       SNODE=SC.DUB.TPYLA2
SVTM055I S1 SESSION (001) ESTABLISHED WITH       PNODE=SC.DUB.TPYLA2
SVTM036I S1 PROCESS STARTED   MVSMVST3(       1) PNODE=SC.DUB.TPYLA2
SVTM036I S1 PROCESS STARTED   MVSMVST3(       1) SNODE=SC.DUB.TPYLA2
```

# Customizing Submit Screens

The following sections describe the Connect:Direct facilities and procedures you need to customize submit screens.

When you type variables 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, directly invokes the IUI submit panel, DMISUBMT, when the SB option is selected. See your *Connect:Direct for z/OS 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 permits. 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$SM03 is a submit menu panel which contains only one menu option that invokes DMISUBMT. You can modify DMI$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 in *Step 2 - Define a General Purpose Process* on page 267.
3. Provide custom submit screens. These screens:
   - Process variables that are resolved as SUBMITS occurs.
   - Build a command on each screen that communicates with the IUI dialog routines.

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. You are notified when the Process is complete.

## Step 1 - Modify the Existing Menu DMI$SM03

To modify the Connect:Direct Submit Menu, add the following in Submit Menu (DMI$SM03):

- 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 you add information. Only the elements necessary to modify an existing menu are displayed. 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
```

**Note:** For SMP/E installation, implement all ISPF panel changes as SMP/E USERMODs to the NISPPLIB target library.

## Step 2 - Define a General Purpose Process

The second step in creating a customized submit screen is to define a Process that the custom Submit screen invokes. 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, use the following information to help you design your screens:

1. Use the generic submit screen (CUSTSAMP) found in the sample library, $CD.SAMPLIB, 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 user ID |

| Variable | Explanation |
|---|---|
| &PNODEID | Security user ID at PNODE |
| &PNODEPW | Current security password at PNODE |
| &PNODENPW | New security password at PNODE |
| &SNODEID | Security user ID at SNODE |
| &SNODEPW | Current security password at SNODE |
| &SNODENPW | New security password at SNODE |
| &DSYMBPARM | Symbolic variable specification |
| &STIME | Start time value |
| &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:** Do not change the &CMD1, &CMD2, &CMD3, and &CMD4 ISPF variables. The SUBMIT command string is built into these four variables.

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 seen 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
                                                +
```

4. 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 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 are made in the INIT section:

- The cursor field is changed to FROMDSN.
- &R is changed to *Y* to indicate RETAIN=YES.
- &STIME is changed to *12:00* to indicate STARTT=(,12:00).
- &FROMDSN and &TODSN are added and initialized to blanks.
- Lines which carry a DELETED comment are removed because they are no longer necessary.

5. 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 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 are made in the PROC section:

- A verify is added for the &FROMDSN variable. It must be non-blank.
- A verify is added for the &SNODE variable. It must be non-blank.

- A test for blanks in &TODSN is added. If &TODSN is blank, it is set to &FROMDSN.
- &CMD1 is changed to contain the string SUB PROC=APROC, the command default.
- &CMD3 is added to contain the string &&DSN1=&FROMDSN. Symbolic substitution is accomplished with this addition. When APROC is submitted, &DSN1 is translated to whatever value is in the &FROMDSN file.
- The next-to-last line is added to concatenate the string &&DSN2=&TODSN to the contents of &CMD3. When APROC is submitted, &DSN2 is translated to the value in &TODSN.
- Lines which carry a DELETED comment are removed because they are no longer necessary.

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 12

# Administering Statistics

This chapter includes the following topics:

- Understanding the Statistics Facility
- Monitoring the Statistics Facility
- Optimizing 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 for z/OS 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.

### File Pair Configuration

The minimum configuration is two file pairs, or four files, however you can configure more than two file pairs. Specify the file pairs in the initialization parameters, STAT.DSN.BASE and STAT.FILE.PAIRS, as data set name high level qualifiers and the number of pairs. This specification determines the configuration of the statistics file pair list.

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.

### Retrieving Statistics with the SELECT STATISTICS Command

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. All the file pairs defined to the DTF are available to SELECT STATISTICS command processing. Connect:Direct searches any file pair that contains records satisfying the SELECT STATISTICS command, not just the files currently being written.

### 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 last. The system records each statistics record as a single VSAM record. The system does not compress the records or add control information.

When a file pair is full, the system switches to the next in the sequence, and begins writing to it. When the last file pair in the list is full, the system wraps back to the first pair in the sequence.

You can also specify the time of a file switch by using the STAT.SWITCH.TIME initialization parameter. For example, you can specify that statistics files switch at midnight every day, which limits 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.

When the system has written 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 for z/OS does a VSAM RESET. This VSAM RESET 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 tools for monitoring the statistics facility:

- INQUIRE STATISTICS command
- Type S2 statistics records
- Type SS statistics records
- SCCSTAT utility

### INQUIRE STATISTICS Command

The INQUIRE STATISTICS command gives a *snapshot* of the status of the facility. See page 291 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 that includes which file pair is active

- Date and time range covered by each file pair
- Size of each file
- Utilization percentage of the entry-sequenced files
- Count of SELECT STATISTICS commands active against each file pair
- Indication if logging is waiting for SELECT STATISTICS to finish so a file pair can be reset
- Indication if logging is waiting for archive to finish so a file pair can be reset
- 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
- Indication of whether archive notification was received for the nonactive files

## S2 Statistics Records

The S2 statistics records contain information about the statistics logging function. The system writes the records about once per hour when activity exists in the DTF. Each S2 record contains statistics about the period of time since the prior S2 record. 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 user ID 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 greatly reduce the search time. This is because the user ID is an indexed field. See the *Connect:Direct for z/OS User's Guide* for more information on how to use the SELECT STATISTICS command.

For example, if you code STAT.USER=statuser, a SELECT STATISTICS request to display all S2 records looks like the following figure.

```
SELECT STATISTICS WHERE (USER=statuser, TYPE=(S2)) TABLE
```

## 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 the index that Connect:Direct uses to search the files and the number of requests issued to the keyed and entry-sequenced clusters. The record also includes the number of records examined and rejected.

Use the SELECT STATISTICS command with the TYPE parameter to view the SS record. See the *Connect:Direct for z/OS 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 runs as a batch job step, and analyzes a single statistics entry-sequenced file.

To help with your estimates of statistics file usage, Connect:Direct provides the SCCSTAT utility program.

### SCCSTAT Utility

Use the SCCSTAT JCL to report on the statistics files. It calculates the average number of CIs used per day at one DTF. The job stream example in the following figure, SCCSTAT, is found in the $CD.JCL distribution library.

```
//JOBNAME  JOB (ACCT),CLASS=A
//*
//*************************************************************
//*                                                           *
//*       Connect:Direct for z/OS                             *
//*                                                           *
//*        This JCL will invoke the utility to produce        *
//*        a report for a 'new format' Statistics File;       *
//*        that is, v2.1+.                                    *
//*                                                           *
//*        Change "$cd" to the high-level qualifier(s)        *
//*        appropriate for your installation.                 *
//*                                                           *
//*************************************************************
//*
//STEP1    EXEC PGM=SCCSTAT
//STEPLIB  DD DISP=SHR,DSN=$cd.ndmlib
//SYSOUT   DD SYSOUT=*
//ESDS     DD DISP=SHR,DSN=$cd.stat.esdsxx    (ESDS of file pair)
```

# Optimizing the Statistics Files

This section describes how to determine the most efficient use of your Statistics file space.

**Note:** Connect:Direct does not support extended-format, extended-addressing ESDS Statistics data sets.

## Statistics Files Space Allocation Example

In this example, the Connect:Direct software is 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, you determine that the records log 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 seven 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 are 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 when 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.

## File Pair Verification

Connect:Direct performs a verification procedure at initialization, as follows:

- Within each file pair, Connect:Direct verifies the appropriate sizing, relative to each other, of the entry-sequenced cluster and the key-sequenced cluster. If the file pair is not relatively sized, then Connect:Direct issues a warning message and initialization continues.
- 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. Connect:Direct verifies that the key-sequenced file really does contain index information for the associated entry-sequenced file.

Connect:Direct uses control information maintained in the key-sequenced file to perform the verification. The software keeps a control record in the KSDS which 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.

### Changing the File Pair

Following are the implications of changing the file pair.

- 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. The two ways to resolve this statistics initialization failure are:
  - First, you can use the DMSTBKEY utility to rebuild the key-sequenced cluster. This utility recreates the KSDS control record so that it matches the new names or control interval size. The records in the file pair remain accessible when the DTF is available again.
  - The second solution is to empty the files. 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 is not a problem. The sizes of both files of a pair 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 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. See page 435 for an example of a file pair list.

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, unless STAT.INIT=COLD is specified in the initialization parameters, Connect:Direct verifies that the order of the file pairs is valid. This verification 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, except across the *wrap point*. Empty file pairs may appear anywhere in the list.

### Changing the Number of File Pairs

It can be useful to periodically change the number of file pairs in the list. To change the number of file pairs, change the STAT.FILE.PAIRS initialization parameter which specifies the number of file pairs. This action adds to or removes from the end of the list.

- Add empty file pairs to the end of the list unless you specify STAT.INIT=COLD.
- Remove records from the end of the list by reducing the STAT.FILE.PAIRS value. When you remove these records, they become unavailable and can 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. Each feature is described in the following sections.

## Archiving Using a Predefined Process

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 predefined 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, ARCHSTAT, is in the $CD.PROCESS distribution library.
- Alternatively, the Process can submit a batch job to archive the data using IDCAMS REPRO, or some other utility. Use the DMRTSUB utility to substitute the file data set name into the submitted job stream. You can invoke DMRTSUB through the RUN TASK statement. A sample archive Process, ARCHSTRJ, that submits a batch job using DMRTSUB, is in the $CD.DISTRIBUTION library. The job submitted is in SAMPLIB member ARCHJOB.

## 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 condition does not present a problem.

### 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. Connect:Direct is notified in several ways:

- 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 is archived.

- If you use a batch job to archive, then the job can send the notification by including a step to execute the DMSTARBT utility.
- Also, you can issue the API command STATISTICS ARCHIVED to inform Connect:Direct to reuse a file pair.

If no indication regarding the completion of the archive exists when Connect:Direct needs to reuse the files, the system issues a message, similar to the example shown below, to the operator console and waits for a reply indicating permission to reuse the file.

```
10.00.01 JOB82592  SSTL013I Statistics file pair switch from 02 to 01, code:TIMER
10.00.02 JOB82592  SSTL009I Arch notification required but not received for file pair 01
10.00.02 JOB82592 *93 SSTL008I Reply "GO" when file available, or "DISABLE" logging.
```

In this situation, all activity in the DTF ceases until a response to the message from the operator exists indicating that the statistics file can now be overwritten. This safeguard occurs as a result of the request that the DTF not erase statistics data unless it is certain that archiving the statistics is complete.

If you have not been requiring archival notification and decide to begin requiring it, you can avoid getting these messages by using the IUI command STAT, CF option and forcing all pairs not in use to indicate they have been archived.

### 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 operation 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 z/OS 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 *Planning DASD Requirements* section in the *Planning the Installation* chapter in the *Connect:Direct for z/OS Installation Guide* for information about the characteristics and relative sizes of the keyed and entry-sequenced clusters of a file pair. See page 286 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 z/OS 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 *VSAM Files DASD Requirement and Description* section of the *Planning the Installation* chapter in the *Connect:Direct for z/OS Installation Guide* for an explanation of estimating space requirements for allocating the directory file.

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 using the INQUIRE STATDIR command, described on page 292.

The archive notification utilities, DMSTARRT and DMSTARBT, write the directory records. If you want to use the directory feature, you must execute one of these utilities from the Process or batch job that archives the records. This condition is true even if you do not specify STAT.ARCH.CONFIRM=YES in the DTF initialization parameters. You must also use these utilities 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, DMSTARBT, 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
- By default, removes the oldest record from a full archive directory to make room for the newest record.

You can invoke DMSTARRT from within a Process through the RUN TASK statement. 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. When the copy operation successfully completes, the system can update the directory and send the archive notification.

The program accepts three parameters through the RUN TASK statement.

- The first parameter is required and is the data set name of the statistics entry-sequenced cluster that is archived.
- The second parameter is optional, and is the data set name of the archive file.
- The third parameter is optional, and specifies whether to age the archive directory.

DMSTARRT *always* sends archive notification to the DTF. If you specify STAT.ARCH.CONFIRM=NO and no notification requirement exists, the notification has no effect.

The addition of the entry in the directory of archive files depends 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 PNODE=primary.node                                -
                 SNODE=secondary.node                              -
                 PRTY=10                                           -
                 STARTT=(TODAY)                                    -
                 &EDSN=
ARC      COPY    FROM (DSN=&EDSN)                                  -
                 TO   (DSN=stat.archive.dsn(+1)                    -
                       DISP=(NEW,CATLG,DELETE)                     -
                 DCB=(DSORG=PS,RECFM=VB,LRECL=2048)                -
                       UNIT=CART -
                       LABEL=(1,SL) )
         IF      (ARC EQ 0) THEN
          RUN TASK (PGM=DMSTARRT,PARM=("&EDSN", -
                                       "stat.archive.dsn(+0)"))
         EIF
```

Whether the archive directory ages off the oldest record to make room for the newest depends on the third parameter, ARCAGE.

ARCAGE=Y - Requests that if the archive directory is full, the oldest record is deleted to make room for the newest. The user receives notification that the record is aged off the archive directory. The amount of time it takes to add a record to the archive directory is insignificant. But the amount of time it takes to age off the oldest record is noticeable and increases with the size of the archive directory, because all of the records in the ESDS must be rewritten. This is the default.

ARCAGE=N - Requests that if the archive directory is full, the utility stops without updating the archive directory.

The following example shows the ARCAGE parameter specified in the DMSTARRT utility for use within a Process through the Run Task statement:

```
ARCHSTAT PROCESS PNODE=primary.node                                  -
                 SNODE=secondary.node                                -
                 PRTY=10                                             -
                 STARTT=(TODAY)                                      -
                 &EDSN=
ARC      COPY    FROM (DSN=&EDSN)                                    -
                 TO   (DSN=stat.archive.dsn(+1)                      -
                       DISP=(NEW,CATLG,DELETE)                       -
                 DCB=(DSORG=PS,RECFM=VB,LRECL=2048)                  -
                       UNIT=CART -
                       LABEL=(1,SL) )
         IF      (ARC EQ 0) THEN
          RUN TASK (PGM=DMSTARRT,PARM=("&EDSN", -
                                       "stat.archive.dsn(+0)"))
         EIF
                "ARCAGE=Y"))
                EIF
```

To prevent aging, set ARCAGE=N. As a result, the newest record is discarded, rather than the oldest.

### DMSTARBT

The DMSTARBT utility (STatistics ARchive BaTch) has the following two functions:

- Notifies Connect:Direct that a statistics file pair is archived and is now available for reuse
- Optionally adds an entry to the directory of archive files
- By default, removes the oldest record from a full archive directory to make room for the newest record.

Execute DMSTARBT as a step within a batch job. Use DMSTARBT when submitting a job that archives the statistical data by executing IDCAMS or some other utility to COPY the data to another file at statistics file pair switch time. The system can update the directory and send archive notification upon successful completion of the copy operation.

DMSTARBT requires that the system allocate the archived statistics file with the data definition name (DDNAME) of STESDS.

DMSTARBT *always* sends archive notification to the DTF. If you specify STAT.ARCH.CONFIRM=NO and no notification requirement exists, then the notification has no effect.

If you want DMSTARBT to update the directory of archive files, the system must allocate the following DDNAMEs:

- STADIR, the directory file
- STARCH, the archive file

In the following example, the archive Process submits a batch archive job using DMRTSUB to substitute the statistics file data set name into the job stream. The system passes this data set name to the archive Process as the symbolic parameter &EDSN. Refer to the *Connect:Direct for z/OS*

*User's Guide* for information about how to use DMRTSUB. The submitted job uses the IDCAMS utility to copy the statistics records to an archive file. If the IDCAMS step is successful, Connect:Direct invokes DMSTARBT to both send the archive notification and update the directory.

The following figure is a sample archive Process.

```
ARCHSTRJ PROCESS PNODE=primary.node                                -
                 SNODE=secondry.node                               -
                 PRTY=10                                           -
                 STARTT=(TODAY)                                    -
                 &EDSN=
         RUN TASK (PGM=DMRTSUB,                                    -
            PARM=("DSN=SYS3.CONNECT.INSTALL.JCL(ARCHJOB),DISP=SHR",-
            "DSNAME &EDSN"))
```

The following figure is a sample archive job stream.

```
//ARCHJOB  JOB (ACCT),CLASS=A
//*
//* ------------------------------------------------------------- *
//*        IDCAMS step to archive the Statistics ESDS:            *
//* ------------------------------------------------------------- *
//*
//STEP1   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INPUT    DD DISP=SHR,DSN=&DSNAME
//OUTPUT   DD DISP=(NEW,CATLG,DELETE),
//            DSN=stat.archive.dsn(+1),
//            UNIT=CART,
//            LABEL=(1,SL)
//SYSIN    DD *
  REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*
//* ------------------------------------------------------------- *
//*        DMSTARBT step to notify DTF that ESDS was archived:    *
//* ------------------------------------------------------------- *
//*
//STEP2   EXEC PGM=DMSTARBT,COND=(0,LT,STEP1)
//STEPLIB  DD  DSN=prod.ndmlib,DISP=SHR
//SYSOUT   DD  SYSOUT=*
//STESDS   DD  DISP=SHR,DSN=*.STEP1.INPUT
//STARCH   DD  DISP=SHR,DSN=*.STEP1.OUTPUT
//STADIR   DD  DSN=stat.archdir.dsn,DISP=SHR
//
```

The DMSTARBT utility accepts the same ARCAGE parameter as specified for DMSTARRT. It is the first and only OS parameter for this utility.

The following example shows the ARCAGE parameter specified in the DMSTARRT utility to run as a step in a batch job:

```
ARCHJOB JOB (ACCT),CLASS=A
//*//
//* ------------------------------------------------------------------ *
//* IDCAMS step to archive the Statistics ESDS: *
//* ------------------------------------------------------------------ *
//*
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INPUT DD DISP=SHR,DSN=&DSNAME
//OUTPUT DD DISP=(NEW,CATLG,DELETE),
// DSN=stat.archive.dsn(+1),
// UNIT=CART,
// LABEL=(1,SL)
//SYSIN DD *
REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//*
//* ------------------------------------------------------------------ *
//* DMSTARBT step to notify DTF that ESDS was archived: *
//* ------------------------------------------------------------------ *
//*
//STEP2 EXEC PGM=DMSTARBT,COND=(0,LT,STEP1),PARM='ARCAGE=Y'
//STEPLIB DD DSN=prod.ndmlib,DISP=SHR
//SYSOUT DD SYSOUT=*
//STESDS DD DISP=SHR,DSN=*.STEP1.INPUT
//STARCH DD DISP=SHR,DSN=*.STEP1.OUTPUT
//STADIR DD DSN=stat.archdir.dsn,DISP=SHR
//
```

To prevent aging, set ARCAGE=N. As a result, the newest record is discarded, rather than the oldest.

### DMSTBKEY

The DMSTBKEY utility (STatistics Build KEYs) loads a statistics key-sequenced cluster with index information for an associated statistics entry-sequenced cluster. DMSTBKEY must execute as a batch job step.

DMSTBKEY enables 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 *Changing the File Pair Configuration* on page 278 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 is about 15% of the size of the associated ESDS. The KSDS must have the characteristics of a statistics key-sequenced cluster. Refer to *VSAM Files DASD Requirement and Description* in the *Connect:Direct for z/OS Installation Guide* for details about allocating statistics clusters.

The following is an example of a job stream to execute DMSTBKEY.

```
//DMSTBKEY JOB (ACCT),CLASS=A
//STEP1    EXEC PGM=DMSTBKEY
//STEPLIB  DD  DISP=SHR,DSN=prod.linklib
//ESDS01   DD  DISP=SHR,DSN=stat.esds01
//KSDS01   DD  DISP=SHR,DSN=stat.ksds01
//SYSOUT   DD  SYSOUT=*
//
```

PARM=CONSOLIDATE allows DMSTBKEY to build a KSDS for an ESDS that contains multiple archived statistics datasets. To use this parameter, create a large flat file of archived statistics, loading them in date order from the oldest to the newest. Next, load this file to an ESDS that will hold all the records and create a KSDS for this ESDS. Then execute DMSTBKEY using PARM=CONSOLIDATE to build the KSDS file. This file pair is available as an Archived Statistics Dataset.

## Sample Archiving Setup

Assume that you have the following requirements for archiving:

- Statistics records must remain available for seven days in the file pair list before being overwritten by new records. After seven days, they must be available in archive files.
- Each archive file can contain no more than one day of statistics records.
- Batch jobs executing the IDCAMS utility to copy the records to sequential files on magnetic tape must perform the archiving. The archive files must be available for 365 days.
- Maintain a directory of archive files.
- Ensure that statistics data is not overwritten before being archived.
- Establish a procedure for making archived statistics available to the SELECT STATISTICS command.

### Sample Statistics Configuration

This section describes how to configure the statistics facility to satisfy these requirements.

You must determine how to configure the statistics file pair list. The rate at which you log statistics records, availability of the statistics records for seven days before being overwritten, and each archive file containing no more than one day of statistics records determine the size and number of file pairs required.

For the statistics records to remain available for seven days after being generated, the total record capacity of all the entry-sequenced statistics files is seven times the average number of records generated daily. Run the SCCSTAT utility to determine how many records, on average, the system writes daily.

According to the sample requirements, a single archive file contains up to the same number of records as a single statistics entry-sequenced cluster, and an archive file contains no more than a day of records. Each ESDS also holds about a day of records, implying that seven statistics file pairs exist. Use the STAT.SWITCH.TIME initialization parameter to initiate a file pair switch every day at midnight rather than depend on a file pair switch occurring regularly as a result of file pairs

filling. To ensure that the switches do not occur before midnight as a result of a file pair becoming full, make each ESDS slightly larger than the daily requirement.

If you determine, using SCCSTAT, that the system writes statistics records at the rate of about 11,000 daily, define seven file pairs each with a capacity of about 12,000 records (RECORDS(12000)). This figure implies that the associated key-sequenced clusters are defined with RECORDS(9000).

RECORDS(365) defines the directory of archive files because the system generates the archive files at the rate of one daily and retains the files for one year.

SYSTEMS.CD.STATS is the data set name prefix for the statistics clusters. The data set name of the archive directory is SYSTEMS.CD.STATS.DIRECTRY. Member ARCHPROC in the data set SYSTEMS.CD.ADMINLIB contains the archive Process that is submitted at file pair switch time.

The following initialization parameters are necessary for the sample archive requirements.

```
STAT.DSNBASE = SYSTEMS.CD.STATS     /*  data set name base         */
STAT.FILE.PAIRS = 7                 /*  number of file pairs       */
STAT.SWITCH.TIME = ( 00:00 )        /*  switch at midnight         */
STAT.SWITCH.SUBMIT = SYSTEMS.CD.ADMINLIB(ARCHPROC) /* archive proc */
STAT.ARCH.DIR = SYSTEMS.CD.STATS.DIRECTRY  /* use directory        */
STAT.ARCH.CONFIRM = YES             /*  be sure archive completes  */
```

The archive Process in the member ARCHPROC follows.

```
ARCHIVE PROCESS  &EDSN=,              -  /* passed stats dsname    */
                 SNODE=CD.PROD,       -  /* PNODE=SNODE            */
                 PNODE=CD.PROD,       -  /*                        */
                 CLASS=1,             -  /* lowest class           */
                 PRTY=15,             -  /* highest priority       */
                 REQUEUE=YES             /* re-queue on error      */
  /*                                                               */
  /*  invoke DMRTSUB to submit the archive job...                  */
  /*                                                               */
       RUN TASK (PGM=DMRTSUB,           -  /* execute DMRTSUB,     */
 PARM=("DSN=SYSTEMS.CD.JCL(ARCHJOB),DISP=SHR", - /*job   */
  /* stream to sub*/
            "DSNAME  &EDSN"))             /* pass stat dsname      */
```

The archive job stream in SYSTEMS.CD.JCL(ARCHJOB) follows.

```
//ARCHJOB JOB (ACCT),ARCHIVE,CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1)
//******* archive the statistical data  ***************************
//ARCHIVE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INPUT    DD DISP=SHR,CD=&DSNAME              /* from DMRTSUB      */
//OUTPUT   DD DSN=SYS.NDM.ARCH.STATS(+1),DISP=(NEW,CATLG,DELETE),
//            UNIT=TAPE,DCB=(BUFNO=6)
//SYSIN    DD *
  REPRO INFILE(INPUT) OUTFILE(OUTPUT)
/*
//******* notify Connect:Direct that the file pair can be reused, ****
//******* and update the directory of archive files    *************
//NOTIFY  EXEC PGM=DMSTARBT,COND=(0,LT)            /* if no errors      */
//STEPLIB  DD DISP=SHR,DSN=SYS.CD.LINKLIB
//STESDS DD DISP=SHR,DSN=*.ARCHIVE.INPUT           /* stat file         */
//STDIR  DD DISP=SHR,DSN=SYSTEMS.CD.STATS.DIRECTRY /* archive directory */
//STARCH DD DISP=SHR,DSN=*.ARCHIVE.OUTPUT          /* archive file      */
//SYSOUT DD SYSOUT=*
//
```

The previous archive job stream indicates that the Connect:Direct administrator manages requests for access to archived statistics records. The submitted requests specify a range of dates and times for the necessary records.

The administrator issues the INQUIRE STATDIR command to determine which archive files contain records for the specified period. The administrator runs the following job stream to create a usable archived statistics file pair for each archive file that it finds.  The first step creates the archive file and copies the record to it. The second step builds the index information.

```
//RESTORE JOB (ACCT),RESTORE,CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1)
//ARCHIVE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INPUT    DD DISP=SHR,DSN=SYS.CD.ARCH.STATS.GnnnnVnn  /* arch seq  */
//SYSIN    DD *
   DEFINE CLUSTER                        - /* define archive KSDS       */
     (NAME(SYS.CDARCH.Dyymmdd)           - /* supply archive date yymmdd */
      VOLUMES(USRVOL)                    -
      INDEXED   NOIMBED                  -
      FREESPACE(0 0)                     -
      KEYS(27 0)                         -
      RECORDSIZE(32 78)                  -
      REUSE NOREPLICATE                  -
      SHAREOPTIONS(2))                   -
     DATA                                -
      (CONTROLINTERVALSIZE(4096)         -
       RECORDS(9000)                     -
       NAME(SYS.CDARCH.Dyymmdd.DATA))    -
     INDEX                               -
      (CONTROLINTERVALSIZE(512)          -
       NAME(SYS.CDARCH.Dyymmdd.INDEX))
   DEFINE CLUSTER                        - /* define archive ESDS */
     (NAME(SYS.CDARCH.Dyymmdd.ESDS)      -
      VOLUMES(USRVOL)                    -
      REUSE NONINDEXED NOIMBED           -
      RECORDSIZE(275 2048)               -
      SHAREOPTIONS(2))                   -
     DATA                                -
      (CONTROLINTERVALSIZE(4096)         -
       RECORDS(12000)                    -  /* same size as stats files */
       NAME(SYS.CDARCH.Dyymmdd.ESDS.DATA))
   IF MAXCC = 0                          -  /* if clusters allocated OK */
   THEN REPRO INFILE(INPUT)              -  /* then load with stats     */
        OUTDATASET(SYS.CDARCH.Dyymmdd.ESDS)
/*
//*******
//******* rebuild statistics index information
//*******
//BLDKEY  EXEC PGM=DMSTBKEY,COND=(0,LT)
//STEPLIB  DD DISP=SHR,DSN=SYS.CD.NDMLIB
//SYSOUT   DD  SYSOUT=*
//ESDSnn   DD DISP=SHR,DSN=SYS.CDARCH.Dyymmdd.ESDS  /* ESDS cluster */
//KSDSnn   DD DISP=SHR,DSN=SYS.CDARCH.Dyymmdd       /* KSDS cluster */
//
```

The archived statistics are now available and you can issue SELECT STATISTICS against the statistics by coding the name of the key-sequenced file with the ARCHDSN parameter, as follows.

```
SELECT STATISTICS WHERE  -
                (PNAME=USERPROC, ARCHDSN=(SYS.CDARCH.Dyymmdd))
```

# 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** | |

No parameters are required for the INQUIRE STATISTICS command.

## Statistics Inquiry through the Batch Interface

To use the INQUIRE STATISTICS command from the batch interface, perform the following steps:

1. Place your command in a batch job stream as described in the *Connect:Direct for z/OS User's Guide*.
2. Submit the job while Connect:Direct is running.

**Note:** 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.

3. Verify your results.

## Statistics Inquiry through the IUI Interface

To use the INQUIRE STATISTICS command from the Connect:Direct IUI, perform the following steps:

1. Access the statistics facility by selecting option **INQ** from the Connect:Direct Administrative Options Menu. The Inquire DTF Internal Status screen is displayed.
2. Type **ISTA** and press **ENTER** to display the status of the statistics logging facility.
3. Verify your results from the statistics logging facility display that is displayed. The report includes information such as the configuration of the statistics file pair list, the active file pair, file percentage utilizations, date and time ranges in the files, and additional information about the statistics facility.

The following figure shows a partial sample report.

```
============================================================
node name      *INQ STATS* DATE: mm/dd/yyyy  TIME: hh:mm:ss
============================================================

Status       => Enabled              Sec. Name  => USER01
Return Code  => 0                    Message ID => SSTL000I
Last "S2"    => 00:00:00             Que Wait   => No
Dsn Base     => USER01.STTX
Excluded     => MC
*****************  F I L E  P A I R  #01  *****************
Status       => Active
Start Date   => 01/20/2006           End Date   => 01/20/2006
Start Time   => 15:25:29             End Time   => 15:51:35
KSDS Size    => 196608               ESDS CIS   => 4096
ESDS Size    => 2211840              ESDS Loc.  => 1474
ESDS Util.   => 1%
Reset Pend.  => No                   Arch. Wait => No
Last Switch  =>                      Sel. Count => 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

No parameters are required for the INQUIRE STATDIR command.

## Optional Parameters

The following table describes the optional parameters used with the INQUIRE STATDIR command:

| Parameter | Description |
|---|---|
| STARTT = ([date \| day] [,hh:mm:ssXM]) | This parameter specifies that the 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 begins with the first directory entry.<br><br>date<br>This parameter specifies that the directory display is to start with this specific date. You can specify the day (dd), month (mm), and year (yy). Connect:Direct for z/OS is Year 2000 compliant and detailed information on Year 2000 compliance is presented in the *Connect:Direct for z/OS Release Notes*.<br><br>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 back slashes (/) to separate the components of a date value.<br>You can omit the separators only for transfers between mainframe nodes. However, you must use separators for transfers between mainframes and all other platforms.<br><br>After you designate the date order in your initialization parameters, you can use the following date formats:<br>**DATEFORM=MDY** specifies the date format as:<br>◆ mm/dd/yy *or* mm/dd/yyyy<br>◆ mm.dd.yy *or* mm.dd.yyyy<br>**DATEFORM=DMY** specifies the date format as:<br>◆ dd/mm/yy *or* dd/mm/yyyy<br>◆ dd.mm.yy *or* dd.mm.yyyy<br>**DATEFORM=YMD** specifies the date format as:<br>◆ yy/mm/dd *or* yyyy/mm/dd<br>◆ yy.mm.dd *or* yyyy.mm.dd<br>**DATEFORM=YDM** specifies the date format as:<br>◆ yy/dd/mm *or* yyyy/dd/mm<br>◆ yy.dd.mm *or* yyyy.dd.mm<br>The following Julian date formats are valid:<br>◆ yyddd *or* yyyyddd<br>◆ yy/ddd *or* yyyy/ddd<br>◆ yy.ddd *or* yyyy.ddd<br>If only date is specified, the time defaults to 00:00. |

| Parameter | Description |
|---|---|
| STARTT = ([date \| day] [,hh:mm:ssXM]) (continued) | day<br>This parameter 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.<br><br>hh:mm:ssXM<br>Requests the first archive file created after this time of day, specified in hours (hh), minutes (mm), and seconds (ss). XM can be 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, you can express 1:00 hours as 1:00AM, and you can express 13:00 hours as 1PM.<br>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 displays files created after noon, or MIDNIGHT, which displays archive files created after midnight. The default for the time is 00:00:00, the beginning of the day.<br>If you specify time of day but not date, the output shows the first available entry in the archive directory for files created after that time of day. Archive files from all later times and dates display up to and including the stop time. |

## Viewing the Statistics Archive Directory through the Batch Interface

To use the INQUIRE STATDIR command from the batch interface, perform the following steps:

1. Place your command in a batch job stream as described in the *Connect:Direct for z/OS User's Guide.*
2. Submit the job while Connect:Direct is running.

**Note:** 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.

3. Verify your results.

## Viewing the Statistics Archive Directory through the IUI Interface

To issue the INQUIRE STATDIR command in the Connect:Direct IUI, perform the following steps:

1. Select option **INQ** from the Connect:Direct Administrative Options Menu to display the statistics facility. The Inquire DTF Internal Status screen is displayed.

2. Type **IDIR** and press **ENTER** to display the directory. A sample of the screen follows.

```
node.name               INQUIRE STATISTICS ARCHIVE DIRECTORY
CMD ==>                                                                 hh.mm
                                                                     mm/dd/yy
                                                                     yyyy.ddd



          START DATE    ==> ________     (Gregorian or Julian)
          START TIME    ==> __________   (HH:MM:SSXM)
```

3. Supply the beginning date and time to limit the display for the INQUIRE STATDIR command. A report showing the results of the inquiry is displayed. 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:    03/02/1998  98.061  00:01:28
Oldest Record:            03/01/1998  98.060  00:00:06
Newest Record:            03/01/1998  98.060  23:59:54

Archival DSN:             USER01.STT.ARCHSTAT.G0009V00
Archival Notification:    03/03/1998  98.062  00:01:35
Oldest Record:            03/02/1998  98.061  00:00:11
Newest Record:            03/02/1998  98.061  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** | |

No parameters are required for the STATISTICS SWITCH command.

### Initiating a Statistics File Pair Switch through the Batch Interface

To use the STATISTICS SWITCH command from the batch interface, perform the following steps:

1. Place your command in a batch job stream as described in the in the *Connect:Direct for z/OS User's Guide.*
2. Submit the job while Connect:Direct is running.
3. Verify your results.

### Initiating a Statistics File Pair Switch through the IUI Interface

The IUI provides a formatted panel that facilitates the issuing of the STATISTICS SWITCH command. To issue the STATISTICS SWITCH command through the Connect:Direct IUI, perform the following steps:

1. Select the **STAT** option of the Administrative Options Menu to access the Statistics Command panel.
2. Select option **FS** on the panel to initiate 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. You can use the INQUIRE STATISTICS command to find out which types are currently disabled.

### Understanding the Use of STATISTICS ON/OFF Command

Use the STATISTICS ON/OFF command prudently when excluding Statistics records logging 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 WTO

Other record types are less critical and you can exclude them.

***Caution:*** Excluding record types can make problem analysis and resolution more difficult.

## 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:

| Parameter | Description |
|---|---|
| TYPE | This parameter specifies the list of statistics record types whose recording is enabled or disabled. Use the 2-character identifier to specify record types. These identifiers are in the table beginning on page 216. |

## Excluding Statistics Logging through the Batch Interface

To use the STATISTICS ON/OFF command from the batch interface, perform the following steps:

1. Place your command in a batch job stream as described in the *Connect:Direct for z/OS User's Guide*.
2. Submit the job while Connect:Direct is running.
3. Verify your results.

Refer to *Understanding the Use of STATISTICS ON/OFF Command* on page 296 for more information on what you do *not* exclude.

### Excluding Statistics Logging through the IUI Interface

To use the STATISTICS ON/OFF command from the IUI, perform the following steps:

1. Select the **STAT** option from the Connect:Direct Administrative Options Menu. The Statistics Command screen is displayed.
2. Select option **EN** to enable logging or option **DI** to disable logging. Supply the list of affected record identifiers in the area provided, and press **ENTER**.

Refer to *Understanding the Use of STATISTICS ON/OFF Command* on page 296 for more information on what you do *not* exclude.

## Notifying Connect:Direct of Statistics File Archival

The STATISTICS ARCHIVED command notifies Connect:Direct that the indicated statistics file is archived. This notification enables 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. The STATISTICS ARCHIVED command provides an additional 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:

| Parameter | Description |
|---|---|
| file pair number | This parameter specifies a number from 1–20 that identifies the statistics file for which archive notification is sent. This number 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. |

## Issuing Archive Notification through the Batch Interface

To use the STATISTICS ARCHIVED command from the batch interface, perform the following steps:

1. Place your command in a batch job stream as described in the *Connect:Direct for z/OS User's Guide.*
2. Submit the job while Connect:Direct is running.
3. Verify your results.

## Issuing Archive Notification through the IUI Interface

To use the STATISTICS ARCHIVED command from the IUI, perform the following steps:

1. Select the **STAT** option from the Connect:Direct Administrative Options Menu. The Statistics Command screen is displayed.
2. Select option **CF**, supply the number of the file pair for notification of archival, and press **ENTER**.

# Chapter 13

# Managing the Transmission Control Queue

This chapter describes the Transmission Control Queue (TCQ) and how to manage it. The following topics are discussed:

- Understanding the TCQ
- Configuring the TCQ
- Troubleshooting the TCQ

## Understanding the TCQ

Connect:Direct stores submitted Processes in the TCQ. The TCQ controls Process execution. The *Connect:Direct for z/OS User's Guide* contains information about how to submit Processes, how to control those Processes once they are in the TCQ, the logical queues that make up the TCQ, and the status values of Processes in the TCQ.

The TCQ consists of two interdependent VSAM data sets:

- The Transmission Control Queue, or TCQ, is a Relative Record Dataset (RRDS) which contains an internal form of the Process language of each Process and status flags.
- The Transmission Control Index, or TCX, is also an RRDS containing a single record. The record contains bitmaps, that indicate the availability of TCQ space.

The default size of the TCQ, as determined by the sample installation JCL, is 1000 records, but can be as large as 4016 records if the sample TCX is used. The size of a Process can range from 1 to 43 records, depending upon the number of steps it contains. The average Process size varies by installation. If the average Process size is 5 records, the sample TCQ can contain approximately 200 Processes.

In order to make use of a TCQ with a capacity that exceeds 4016 records, the TCX must be defined with a record size and control interval (CI) size that is larger than those specified by default in the installation JCL. The *Connect:Direct for z/OS Installation Guide* contains more information about planning your space requirements.

---

**Note:** Both the TCQ and TCX can be defined with a CISIZE (Control Interval size) of up to 30,720 bytes. The maximum number of TCQ records that can be mapped by the maximum-sized TCX is 122,804 .

To hold the maximum size Process (1 MB), the CISIZE of the TCQ must be at least 24 KB bytes.

---

# Configuring the TCQ

Connect:Direct provides several initialization parameters, that allow you to configure the TCQ. These parameters fall into three categories:

- Controlling startup—two parameters determine what the TCQ does with existing Processes:
  - TCQ = WARM | COLD, whose default value of WARM specifies that all existing Processes in the TCQ are retained. COLD requests that the TCQ be cleared of all processes
  - CONFIRM.COLD.START = YES | NO, whose default value you must change to force the operator to confirm the request for a COLD start before executing it.
- Controlling efficiency of the TCQ—the following parameters provide several flexible configuration options in this area.
  - MAX.AGE lets you specify the number of days to wait before purging a Process. With this parameter you can also manage the Wait and Hold queues by specifying which type of Process to purge (that is, those with a specific status) or the number of days to wait to purge for each status type.
  - MAX.AGE.TOD is an optional parameter, which you can use to change the system default of automatically purging the TCQ at midnight and whenever Connect:Direct is initialized.
  - TCQ.THRESHOLD specifies when Connect:Direct will issue a warning message that the TCQ is reaching its capacity. The default is that a message is not issued until the TCQ is completely full.
- Holding Processes—two parameters determine if submitted Processes are held.
  - QUIESCE specifies whether or not Connect:Direct holds Processes from execution.

**Note:** The QUIESCE parameter helps you in your efforts to clean up a TCQ, that has become corrupt. See *Using the TCQ/TCX Repair Utility (CDTCQFIX)* on page 303 for details.

  - REQUEUE specifies whether to requeue a Process, that ABENDS or results in a return code greater than 4.

Appendix A, *Global Initialization Parameters*, contains detailed information on these parameters.

# Troubleshooting the TCQ

Connect:Direct provides several ways to recover from a system malfunction associated with a TCQ problem. Refer to the *Connect:Direct for z/OS User's Guide* for the facilities available to resume Process execution, such as Checkpoint/Restart. Another TCQ recovery facility available is the TCQ/TCX Repair Utility.

The TCQ/TCX Repair Utility, CDTCQFIX, allows you to solve corruption problems related to the TCQ/TCX data sets without having to cold start the DTF and reinitialize the TCQ. The CDTCQFIX batch program retains the original TCQ/TCX data sets used in production, and creates a new validated copy of the TCQ by removing all invalid Processes. The CDTCQFIX utility can also be used to create a TCQ and TCX for pre-version 4.6 Connect:Direct. Processes that are larger than 64 KB (the pre-version 4.6 Process limit) are removed from the new TCQ and TCX when the BACKLEVEL parm is used.

**Note:** The CDTCQFIX utility can be used to build a new TCX/TCQ pair with increased CISIZEs to hold larger Processes.

## Using the TCQ/TCX Repair Utility (CDTCQFIX)

You can run the CDTCQFIX utility in one of the following ways:

- *Rebuild TCX* mode, which creates a new TCX by using the current TCQ to indicate the existence of Processes.
- *Use TCX* mode, which creates a new TCQ by using the current TCX to indicate the existence of Processes. The Use TCX mode is recommended for pre-version 4.4 TCQ/TCX migration use only.

**Note:** It is recommended that you use different names to distinguish the original and new TCQ/TCX data sets in case you need to go back and reuse the original data sets.

The Rebuild TCX mode is the preferred mode for rebuilding the TCQ/TCX after encountering TCQ corruption problems, which cause U3083 abends.

***Caution:*** When CDTCQFIX runs in Rebuild TCX mode, the TCX is rebuilt based upon the TCQ contents, and any pre-version 4.4 Processes previously completed but found intact may be restarted when the DTF initializes. To avoid this potential problem with pre-version 4.4 TCQs, first run the CDTCQFIX utility in Use TCX mode, and then use the Rebuild TCX mode for all future rebuilds of the TCQ/TCX.

The Use TCX mode is recommended for TCQs associated with systems running versions of Connect:Direct prior to Version 4.4. Prior to Version 4.4, completed Processes were retained in the TCQ and were not deleted.

The CDTCQFIX program, located in $CD.LINKLIB, has one execution parameter for specifying the report type.

| Parameter | Description |
|---|---|
| PARM=<br>SUMMARY \|<br>DETAIL \|<br>BACKLEVEL | Specify SUMMARY to produce a report at the Process level.<br>Specify DETAIL to produce a report, which shows steps within each Process, such as RUN TASK, RUN JOB, SUBMIT, and COPY.<br>Specify BACKLEVEL to create a TCQ and TCX for pre-version 4.6 Connect:Direct systems. |

Normally, you run CDTCQFIX with Connect:Direct shut down but you could run it in production. The data sets and reports created will be correct as long as no update activity to the input TCQ takes place while the utility executes. The program issues a warning message if the VSAM timestamp for the input TCQ is changed during execution.

The return codes associated with the CDTCQFIX utility are described in the following table.

| Return Code | Meaning |
|---|---|
| 0 | No errors were found in the input TCQ |
| 4 | At least one error was found and removed or a warning message was issued |
| 8 | A severe error occurred during execution and the utility was terminated |

## Initializing the DTF After CDTCQFIX Has Found Errors

If errors were found and corrected when you ran the CDTCQFIX utility, you should replace the original corrupted data sets currently in use with the new data sets created by CDTCQFIX. To allocate the new data sets to Connect:Direct, you can either use IDCAMS ALTER or regenerate the network map. Use one of the following procedures.

### Using IDCAMS ALTER

1. Shut down the DTF, if necessary.
2. Rename the old TCQ and TCX to save the original data sets.
3. Using ALTER, rename the new TCQ and TCX data sets using the original data set names.
4. Initialize the DTF and specify the QUIESCE=YES initialization parameter.
5. Use the SELECT PROCESS command to display the TCQ contents, and then delete any unwanted processes.
6. Issue the MODIFY Sessions command to resume DTF operation.
7. After you are confident that Connect:Direct is operating normally with the new TCQ and TCX data sets, delete the original TCQ and TCX datasets.

### Regenerating the Netmap

1. Shut down the DTF, if necessary.
2. Execute the Unload Netmap utility, DMCNTMPL.
3. Change the names of the TCQ and TCX data sets in the unloaded member. The names are defined within the LOCAL.NODE definition.
4. REPRO the old network map data to preserve a copy of it for fallback purposes.
5. Delete, define, and reload the netmap.
6. Initialize the DTF and specify the QUIESCE=YES initialization parameter.
7. Use the SELECT PROCESS command to display the TCQ contents, and then delete any unwanted processes.

8. Issue the MODIFY Sessions command to resume DTF operation.
9. After you are confident that Connect:Direct is operating normally with the new TCQ and TCX data sets, delete the original network map, TCQ and TCX data sets.

## CDTCQFIX Examples

The following three JCL samples are provided in $CD.SAMPLIB:

- TCQFIX1, which runs CDTCQFIX in *Rebuild TCX* mode. CDTCQFIX will create new TCQ and TCX data sets and print a detailed report.
- TCQFIX2, which runs CDTCQFIX in *Use TCX* mode. CDTCQFIX will use the current TCX data set to indicate the existence of TCQ Processes. The program will print a summary report.
- TCQFIX3, which runs CDTCQFIX in *BACKLEVEL* mode. CDTCQFIX will create new TCQ and TCX datasets and print a summary report. Processes larger than 64 KB are removed.

The following sample shows part of the JCL within SAMPLIB member TCQFIX1.

```
//*************************************************************
//*                                                           *
//*   Run the TCQ FIX utility in "Rebuild TCX" mode.  Create  *
//*    new TCQ and TCX with any invalid contents of input     *
//*    TCQ removed.  Request detail-level reporting for       *
//*    input and output TCQ:                                  *
//*                                                           *
//*************************************************************
//STEP2    EXEC PGM=CDTCQFIX,PARM=DETAIL
//STEPLIB  DD DISP=SHR,DSN=CD.LOADLIB
//SYSOUT   DD SYSOUT=*
//TCQIN    DD DISP=SHR,DSN=CD.OLD.TCQ
//TCQOUT   DD DISP=SHR,DSN=CD.NEW.TCQ
//TCXIN    DD DISP=SHR,DSN=CD.OLD.TCX (not needed in this mode)
//TCXOUT   DD DISP=SHR,DSN=CD.NEW.TCX
//TCQINRPT DD SYSOUT=*
```

You can accomplish various tasks depending upon which DD statements are present in the batch JCL:

- If TCXOUT is present, *Rebuild TCX* mode is requested; otherwise, *Use TCX* mode is requested.
- If TCQINRPT is present, an input TCQ report is generated; otherwise, it is not.
- If TCQOUT is present, a new TCQ is created; otherwise, it is not. You can therefore analyze and report on the input TCQ without creating any data sets.

**Note:** TCQOUT and TCXOUT must both be pre-allocated empty VSAM data sets.

## CDTCQFIX Output

The sample reports shown in this section are a result of running CDTCQFIX in *Use TCX* mode. The first report shows two invalid TCQ Processes, which were detected and skipped during the copy of

the input TCQ to the output TCQ. The second report lists all Processes in the input TCQ in sequential Process number order.

### CDTCQFix Output TCQ Report (Summary)

```
Connect:Direct for z/OS

CDTCQFIX execution on 18 Mar 2003 14:58:16
Mode: Use TCX


Output TCQ Summary

PName     PNum   Cur Step Submitter Node   Other Node       Stat  UserID   Submitted
-------- ------- -------- ---------------- ---------------- ----- ------   ----------
D3103UPR       4 STEP0108 PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:08
D3104UPR       5 STEP0103 PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:14
D3105UPR       6 STEP0103 PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:17
D3106UPR       7 STEP01   PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:21
D3107UPR       8 STEP01   PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:24

Processes Skipped Summary

PName     PNum   Cur Step Submitter Node   Other Node       Stat  UserID   Submitted
-------- ------- -------- ---------------- ---------------- ------------   ---------
D3101UPR       2          PLEX.JOE         PLEX.TOM.TCP     HO HI CBENN1   19 DEC 2002  11:08:45
D3102UPR       3          PLEX.JOE         PLEX.TOM.TCP     HO HI CBENN1   19 DEC 2002  11:08:59


Totals:

Processes found in Input TCQ:                          7
Processes written to Output TCQ:                       5
Processes not copied to Output TCQ (skipped):          2

STQF001E  CDTCQFIX ended; RC=04
```

### CDTCQFIX Input TCQ Report (Summary)

```
Input TCQ Summary

PName     PNum   Cur Step Submitter Node   Other Node       Stat  UserID   Submitted
-------- ------- -------- ---------------- ---------------- ----- -------- ------------------
D3101UPR       2          PLEX.JOE         PLEX.TOM.TCP     HO HI CBENN1   19 DEC 2002  11:08:45
D3102UPR       3          PLEX.JOE         PLEX.TOM.TCP     HO HI CBENN1   19 DEC 2002  11:08:59
D3103UPR       4 STEP0108 PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:08
D3104UPR       5 STEP0103 PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:14
D3105UPR       6 STEP0103 PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:17
D3106UPR       7 STEP01   PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:21
D3107UPR       8 STEP01   PLEX.JOE         PLEX.TOM.TCP     EX EX CBENN1   19 DEC 2002  11:09:24
```

# Chapter 14

# Managing Files with Connect:Direct File Agent

Connect:Direct File Agent is a component of Connect:Direct that provides unattended file management. Before using Connect:Direct File Agent, you must plan how to configure it to automate file management for your site. After planning what you need to accomplish, configure File Agent to connect to a Connect:Direct server, watch the directories that files of interest will be added to, and submit a specified Connect:Direct Process to the server when a file is detected.

## Connect:Direct File Agent

Connect:Direct File Agent provides monitoring and detection capabilities that enhance the automation you accomplish with Connect:Direct Processes. You cannot create Processes with File Agent; however, File Agent variables can pass arguments to a Process. File Agent does not delete, copy, or move files directly, but it helps you accomplish such tasks by submitting the Process you specify in the configuration to the Connect:Direct server. Before you configure File Agent, you must create and test the Connect:Direct Process that you intend to specify as the default Process in the File Agent configuration.

Using the File Agent configuration interface and Help system, you define the *default configuration file* (Default_Config.ser). The default configuration file defines the Connect:Direct server that Connect:Direct communicates with; the directory, or directories, that File Agent monitors; and how a file added to a watched directory or a detected system event is processed.

You can configure Connect:Direct File Agent to operate in either of the following ways:

- Watch for any file to appear in one or more *watched* directories and submit the default Process after detecting the newly added file.
- Override the default Process specified and apply either *watched file event* rules (Submit Process rule) or *system event* rules that are enabled for the configuration. File Agent applies a *watched file event* rule to a detected file by checking file properties to determine whether criteria specified by the rule are met. A *system event* rule checks whether a system event meets criteria specified by the rule. If all criteria for a rule are met, File Agent submits the Connect:Direct Process associated with that rule.

You can create File Agent rules based on the following properties:

- ✦ Full or partial name of the file detected in a watched directory
- ✦ Size of the file detected in a watched directory
- ✦ System event title
- ✦ System event contents (as included in a stack trace)

You can specify more than one rule in a File Agent configuration; each rule can have File Agent submit a different Process.

**Note:** Although you can create multiple rules as part of a File Agent configuration, File Agent processing ends when all criteria for a rule are met. Therefore, you should specify rules so that those with more specific criteria (properties) are listed first in the configuration.

For optimum performance, you should configure Connect:Direct File Agent to communicate with the Connect:Direct node where it is installed. You can configure File Agent to use continuous signon and remain connected to the API port for the Connect:Direct server at all times, or configure it to connect to the port only when it needs to. File Agent is available on UNIX, Windows, and OS/390 (and z/OS) operating systems. When you use Connect:Direct for UNIX or Windows, the watched directory is a UNIX path name or a Windows path to the directory. When you use Connect:Direct with OS/390 and z/OS, the watched directory can be a fully specified HFS path name for a file or a directory, a fully specified MVS data set name, a partial MVS data set name, or the name of a partitioned data set (PDS) or partitioned data set extended (PDSE).

File Agent can monitor multiple directories, including local and network directories. File Agent scans the watched directories you specify in the configuration for newly added files (unless you specify a rule to force other operation). By default, Connect:Direct File Agent scans a watched directory once each minute. For example, if you start File Agent at 1:00 p.m., a file added to that watched directory at 12:55 a.m. is not detected. If you start File Agent at 1:00 p.m., and a file is placed in the watched directory at 1:01 p.m., then File Agent detects this newly added file. File Agent detects a file only one time, unless the file is accessed and saved with a later timestamp.

Using Connect:Direct File Agent requires an understanding of Connect:Direct Processes, operating systems, and scripting (for regular expression operator use with File Agent rules).

## Running File Agent

You can run File Agent from a UNIX or MS-DOS command line, configure it to start automatically as a Windows Service at system startup, or configure it to run from a Windows shortcut. Use the command line to verify that File Agent is working correctly or to specify an alternate configuration file. After you run File Agent from the command line to verify that File Agent is operating correctly, run it using the method that requires the least user intervention.

When File Agent runs as a Windows service, it is fully automated, requiring little user intervention. On UNIX, you can modify the initialization sequence of the computer to call the cdfa.sh script and run Connect:Direct File Agent whenever you restart the computer. On OS/390 or z/OS, you must run the appropriate job to start the File Agent configuration interface, or start or shutdown the File Agent.

Depending on your network and how you use Connect:Direct, there might be more than one File Agent running (on different hosts). The first File Agent that connects to a Connect:Direct server becomes the File Agent *gate keeper*. The gate keeper port is used to keep track of locations monitored in case multiple File Agents are configured to watch a single directory. The gate keeper ensures that only one File Agent detects a file that appears in a watched directory.

---

***Caution:*** When running multiple File Agents (on multiple hosts), each File Agent must use a unique gate keeper port number. If multiple agents use the same gate keeper port number, the first File Agent connects, becomes the gate keeper, and operates as configured. However, if another File Agent with the same gate keeper port number connects subsequently, monitoring and rule processing for that File Agent may not occur as configured.

---

### File Agent Logging Capabilities

File Agent logging capabilities vary by platform. Running File Agent from a command line using the verbose argument turns on File Agent logging when it is available. When you run File Agent as Windows service, no information is logged unless an error occurs.

Connect:Direct File Agent provides the following levels of status information when logging is available on a platform:

- System log—Shows all system activity. This log is created only when you specify verbose mode or if an error occurs. If you are running verbose mode from the command line, this log information is shown in the command line window. If you are not running in verbose mode, the system log appears in the snaps directory (located in the File Agent installation directory), which is created when the first event occurs.
- First Failing Status (FFS) log—One or more logs created when an error occurs. The snaps directory is created as needed and contains the FFS logs for any errors encountered by File Agent.

Using trace commands provided for your platform also captures details related to File Agent operation.

### File Agent Configuration Interface and Help

Instructions for configuring File Agent are available in the online Help system that you access from the configuration interface. Field-level Help is displayed in the bottom pane of the configuration interface. Clicking **Help** displays the online configuration procedures.

## Planning the File Agent Configuration

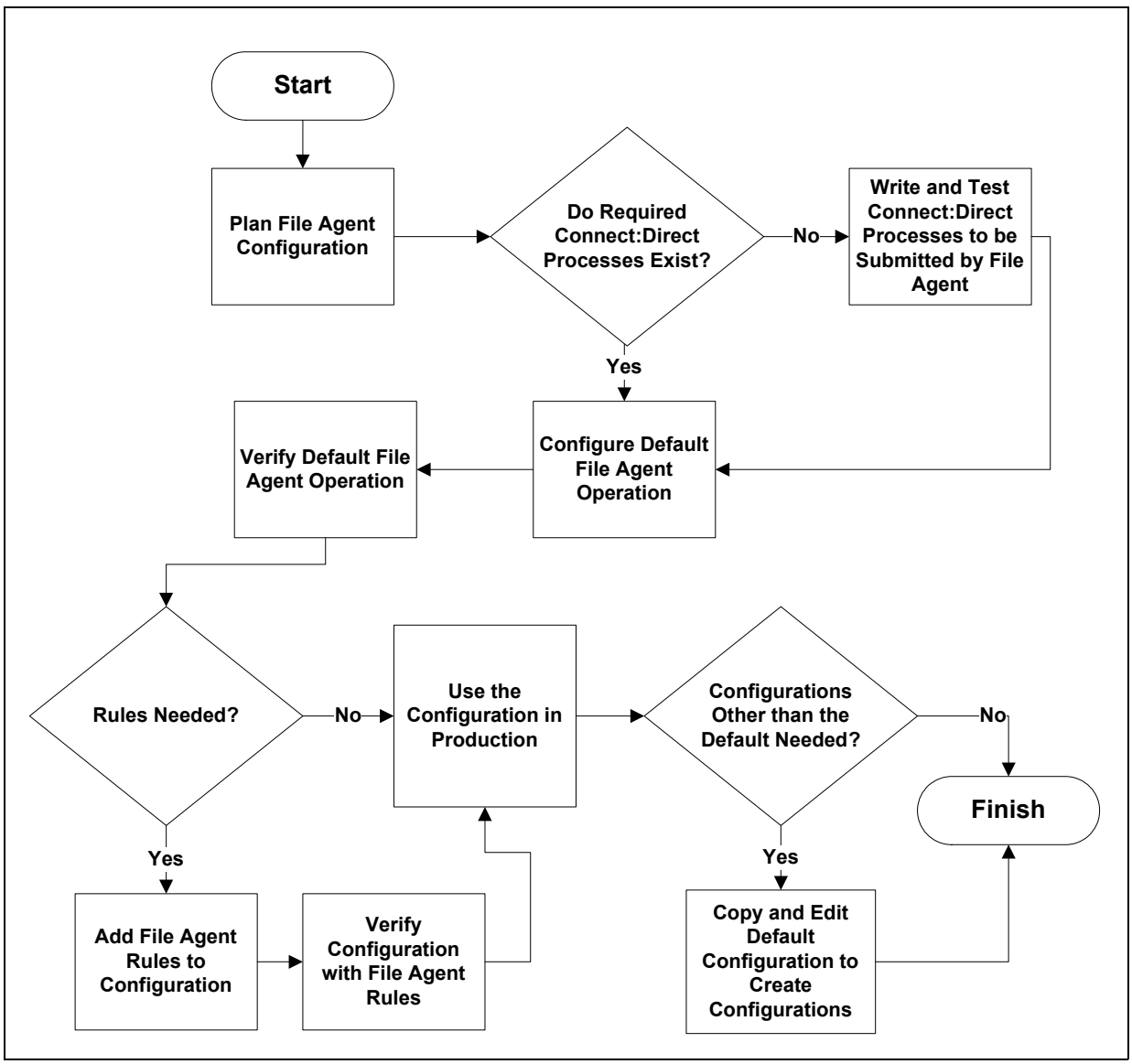
Before you begin configuring File Agent, you must use Connect:Direct to choose or create the Processes that perform the actions you want to automate. You will need to carefully configure File

Agent to connect to the Connect:Direct server and to monitor and detect conditions (such as a file addition to a directory). At detection, Connect:Direct File Agent submits the Process for executing actions that need to be performed in response to those conditions.

Refer to *Connect:Direct File Agent Configuration Examples* to review some configuration scenarios that can help you plan your File Agent configuration. When you configure File Agent, it is best to take an incremental approach; that is, first specify the server connection, a default Process, and the watched directories. Run a test from the command line to ensure that the default File Agent configuration is working correctly. After a successful test of the default configuration, you can run the File Agent Configuration Interface again to start building and testing any File Agent rules that you want to apply, one by one. After you successfully create a default configuration, you can use the file as the basis for other configuration files.

Use the *File Agent Worksheet* to gather the information you need to configure File Agent. Contact your system administrator for the site-specific information you will need to establish a connection to the Connect:Direct server. As you complete the worksheet, run the File Agent Configuration Interface and use the File Agent Help system to learn about entering parameters. The Help system and field-level help provide complete descriptions of parameters and arguments you can specify in the configuration file. Make copies of this worksheet if you have to configure File Agent on multiple Connect:Direct servers.

The following diagram illustrates the flow of steps for setting up File Agent for use in a production environment.



## File Agent Worksheet

| **Connect:Direct Server Connection Information** | |
|---|---|
| Userid for API (for connecting to the Connect:Direct server) **Required**<br>Must match the user ID used to submit the default Process. | _____ |
| Password for API (for connecting to the Connect:Direct server) **Required**<br>Must match the password used to submit the default Process. | _____ |
| API host DSN name (name of the host on which the Connect:Direct server is located) **Required** | _____ |
| API port (default =1363)<br>1–5 digit port number that Connect:Direct File Agent uses to connect to the Connect:Direct server API. **Required** | _____ |
| Gatekeeper port (default=65530)<br>Port used to track directory monitoring and ensure that multiple File Agents do not monitor a single directory. **Required** | _____ |
| Gate keeper DNS name (optional)<br>(default=127.0.0.1) | _____ |
| **Default Process and Watched Directory Information** | |
| Watched directories:<br>For Windows and UNIX, one or more valid specifications of paths (Windows) or pathnames (UNIX). For z/OS, one or more fully specified HFS pathnames of a file or directory, or a full or partial MVS data set name. **Required**<br>List one valid entry per line. | |
| | |
| | |
| | |
| | |
| | |
| **Default Process and Watched Directory Information** | |

| | |
|---|---|
| Default Process:<br>Windows and UNIX: Valid path and name of the file that contains the default Process on the Connect:Direct server.<br>z/OS: Member Name in DMPUBLIB<br>**Note:** If you do not specify a default Process or create a rule, no processing is performed when a file or event is detected. | _____<br>_____ |
| Default arguments (See Help for complete list.)<br>Argument string to pass to the default Process in the following format:<br>&FA_*XXXX_XXX*.<br>**Note:** The ampersand sign (&) and period (.) are required. | _____ |
| Error Process: | _____ |
| Error arguments | _____ |
| Process class (default=1) **Required** | _____ |
| Process priority (default=1) | _____ |
| Watched file interval (default=1 minute) | _____ |
| File completion delay (default=1 minute) | _____ |

**Note:** If you are using X Windows, the X11 display variable is used to connect to the GUI server for terminal emulation. The File Agent Configuration Interface will display on the monitor specified for the X11 display variable. If you want to display the File Agent Configuration Interface on a Windows computer, you must specify the network ID of the terminal you want to use for displaying the File Agent Configuration Interface.

# Connect:Direct File Agent Configuration Examples

The following examples illustrate three typical scenarios for using Connect:Direct File Agent. Fields that are not required to be set for the operation demonstrated in the example are not included in the tables of configuration parameters. Required fields are indicated by an asterisk(*) in the File Agent Configuration Interface, and all fields are described in field-level Help. You can use the sample scenarios to become familiar with settings for parameters you must set using the configuration interface in order to accomplish watched directory monitoring and file detection needs.

See the *File Agent Worksheet* on page 312 for a description of the parameters required to establish the connection. The tables of sample data for these scenarios assume that you have already configured the site-specific parameters required to establish a connection to the Connect:Direct server where File Agent is installed. The sample scenarios also assume that the Connect:Direct Processes that will perform tasks associated with Connect:Direct File Agent file detection activities have been created.

## Detecting a File Added to a Watched Directory on a z/OS System

Some users need to access a report file that is expected to be transferred to a location that only administrators can access. File Agent can be configured to perform the processing on a z/OS system:

- Monitor the watched data set called EASTERN.Q1.REPTS.
- Submit a default Process called DEFPROC. The default Process has been created to copy a file detected in the watched data set to a specified location for access by users.

| Tab | Field | Sample or Description |
|---|---|---|
| **File agent** | Watched directories | Type EASTERN.Q1.REPTS to specify the fully qualified MVS data set name to watch. |
| | Default Process | Type DEFPROC, the member name for the Process in DMPUBLIB.<br>**Note:** If no default Process is specified and the file does not match a rule, then no processing occurs. |

## Detecting a VSAM Data File Added to a Watched Directory on a z/OS System

Each month, users in the accounting department need to access a VSAM data file that contains their company's monthly payroll information. The name of the data file containing this information is VSAM.*mm.yy*.PAYCHECKS.DATA where *mm* is the month and *yy* is the year. The data file is expected to be transferred to a location that only administrators can access.

The Connect:Direct administrator configured File Agent to watch for any file containing the string, VSAM.**.**.PAYCHECKS, and then to copy it to the directory location the accounting users could access. When the administrator tested File Agent, she discovered that the Process had been submitted three times because File Agent was triggered for the following VSAM files when the VSAM cluser was created:

- VSAM.*mm.yy*.PAYCHECKS
- VSAM.*mm.yy*.PAYCHECKS.INDEX
- VSAM.*mm.yy*.PAYCHECKS.DATA

To configure File Agent to watch only for VSAM data files and not other VSAM-related files, the administrator modified the match string and specified VSAM.**.**.PAYCHECKS.DATA as the the VSAM data set to watch. She configured File Agent to perform the following processing:

- Monitor the watched data set called VSAM.**.**.PAYCHECKS.DATA
- Submit a default Process called DEFPROC. The default Process has been created to copy a file detected in the watched data set to a specified location for access by users.

| Tab | Field | Sample or Description |
|---|---|---|
| **File agent** | Watched directories | Type VSAM.**.**.PAYCHECKS.DATA to specify the fully qualified VSAM data set name to watch. |
| | Default Process | Type DEFPROC, the member name for the Process in DMPUBLIB.<br>**Note:** If no default Process is specified and the file does not match a rule, then no processing occurs. |

## Detecting a File by File Size on a Windows System

Customer transaction files are regularly transferred into the Windows directory c:\monthend\datafile. Files larger than 1 MB require special processing that will automatically be performed on files in a certain directory.

The sample values in the following table accomplish the following processing:

- Monitor the watched directory c:\monthend\datafile
- Apply the rule titled Find big file to detect files larger than 1 MB.
- Override the default Process and submit a Process that is associated with the Check file size rule. This fixbigfile.cdp Process will copy a file larger than 1 MB from the c:\monthend\datafile directory to the c:\reprocess directory.
- Pass the path and file name of the file that meets the criteria for the find big file rule to the Process fixbigfile.cdp.

| Tab | Field or Dialog Box | Actions and Sample Entry |
|---|---|---|
| **File agent** | **Watched directories** field | Specify the directory to watch. Type the directory path for the directory that File Agent will watch:<br>**c:\monthend\datafiles** |
| **Rules** | **Create rule** dialog box | Click **New** and type the name you want to give the rule in the field:<br>**find big file**<br>Click **find big file** in the list of rules and click **Edit**. |
| | **Match criteria list for rule "find big file"** | Specify the criterion to check for a detected file. Select the default criterion name, **Not enabled: system event title matches " "** and click **Edit match**. |
| | **Edit match criterion for rule "find big file"** dialog box | ◆ Click **Enabled** to enable the criteria you are about to specify.<br>◆ Click **Size of the newly arrived file**<br>◆ Click **Matches** to display the options for the comparison. Click **Greater than** to define the how the file size should compare.<br>◆ Type **1024** in the **Compare size** field and click **OK**. |
| | **Process name** field | Scroll down to view the **Submit Process information for watched file event rule "find big file"**. Type the directory path and filename for the Process that File Agent is to submit when the rule Find big file results in a match:<br>**c:\processes\fixbigfile.cdp**<br>Click **Done** and then click **Save**. |

## Detecting a System Event by Title on a Windows System

IndexOutOfBoundsException is the title of an event that indicates a number is outside of an expected range. In the following example, File Agent is used to detect an event with IndexOutOfBoundsException in the title, pass a string (the event title) to a Connect:Direct Process, and then submit a Process to the Connect:Direct server that will perform actions the environment requires for this type of event. In this scenario, the event IndexOutOfBoundsException could indicate activity that a network administrator should investigate. Because the site uses a Connect:Direct mailbox system, the configuration will include the administrator's account to be notified when Connect:Direct File Agent submits a Process for the IndexOutOfBounds rule.

The sample values in the following table accomplish the following processing:

- Override the default Process and submit \processfolder\oo_boundserrproc.cdp
- Send a message to the administrator's mailbox system account after submitting the oo_boundserrproc.cdp Process for the rule.

| Tab | Dialog Box, Window, or Field | Description/Example |
|---|---|---|
| **Rules** | **Create rule** dialog box | Type **index out of bounds** as the name of the rule you are creating. |
| | **Match criteria list for rule "index out of bounds"** window | Select the default criteria **Not enabled: System event title matches " "** and click **Edit match**. |
| | **Edit match criterion for rule "index out of bounds"** dialog box | ◆ Click **Enabled** to enable the criteria you are about to specify.<br>◆ Click **System event title** as the criterion to match for the rule.<br>◆ Click **Matches** on the drop-down field to see the options for comparison to a string.<br>◆ Click **Contains** to specify how the compare string should relate to a system event title that File Agent detects.<br>◆ Type **IndexOutofBounds** as the **Compare String** to indicate that the system event title should include this string.<br>◆ Click **OK**. |
| | **Submit Process information for system event rule "index out of bounds"** window | Type information into the fields that will define the Process to submit and the mailbox user to notify after the Process is submitted. |
| | **Process name** field | Type **c:\processfolder\errproc.cdp** to specify the path and file name for the Process File Agent submits when a file meets the rule criteria. |
| | **Notification userid** field | Type **adminjim@company.com** to specify the user to notify when File Agent submits the Process. |

## Passing the UNIX Pathname for a Detected File to a Process

Because Connect:Direct File Agent can watch multiple directories for the appearance of a new file, the Connect:Direct Process that File Agent is to submit to the server at the appearance of a new file might need to reference the Windows Path or UNIX pathname for the detected file as part of commands and statements in the Process.

In the following example, a UNIX pathname is passed to the default Process, copynewfile.cdp.

| Tab | Dialog box, Window, or Field | Sample Entry |
|---|---|---|
| **File agent** | **Watched directories** | Type one UNIX pathname per line for each location File Agent is to watch for the appearance of files:<br>**user/bin/monthend/**<br>**quartend/easterndiv/errorfiles**<br>**managers/special/reports**<br>File Agent checks these pathnames for new files. |
| | **Default process** | Type the UNIX pathname and filename for the Connect:Direct Process that File Agent is to run when a file appears in any watched directory specified:<br>**user/bin/admin/copynewfile.cdp**<br>The pathname where File Agent detected a new file is passed to this Process. |
| | **Default arguments** | Type the File Agent variable for passing a UNIX pathname or Windows path, including the leading percent sign (%) and the ending period (.):<br>**&FAP=%FA_PATH_FOUND.**<br>In this example, **&FAP** is the variable to which File Agent will pass the UNIX pathname where the file was detected. **%FA_PATH_FOUND.** is the File Agent variable used to indicate the information to pass to the Connect:Direct Process. |

# Chapter 15

# Supporting DBCS

This chapter describes the following topics:

- Overview of DBCS
- Translation Tables
- Customizing the Translation Tables

## Overview of DBCS

Some languages have too many symbols for all 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'. The Korean and other ideographic languages contain several thousand characters. To create these coded character sets, two bytes are needed for each character.

The Connect:Direct 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 character representation differs between operating systems. 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. SO/SI combinations are not required 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 323 for correct mapping of DBCS characters by language.

# Translation Tables

Connect:Direct provides the following translation tables in both load module and source form. The executable tables are located in $CD.LINKLIB and the source tables are in $CD.SAMPLIB. You can copy and customize the source code format for your processing environment. For more information on how to use these tables with the SYSOPTS parameter in the COPY statement, refer to the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.

| Table Name | Description |
|---|---|
| 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 |
| JEFXJIS | Japanese host EBCDIC Katakana to ASCII |
| JISXJEF | ACSII to Japanese host EBCDIC Katakana |
| GBKXNHC | GBK to Chinese new host code |
| NHCXGBK | Chinese new host code to GBK |

# Customizing the Translation Tables

You can create and update the translation tables through a preprocessor that takes simple batch input in a predefined format and creates output compatible with the assembler. You can then assemble and link-edit the output to produce a translation table you can load.

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 TRANSLATIO N 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

creates the double-byte character set translation table. This table translates 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.

The following 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 you use NAME, it must be the first parameter defined. If you use NAME 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 in *HLQ*.SAMPLIB.

### 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 you use TITLE, it must be the first parameter defined. If you use TITLE with NAME, NAME and TITLE must be the first two parameters defined. The default for TITLE is DBCS TRANSLATION TABLE.

The following example shows the syntax for the TITLE parameter.

```
TITLE=HOST EBCDIC TO ASCII KS5601 TRANSLATION
```

The EBCXKSC table is provided in *HLQ*.SAMPLIB.

### DEFAULT=nnnn

contains the hexadecimal representation you define as the replacement for invalid DBCS code points. This default character is displayed wherever a nontranslatable character is displayed in the data being received. The default is 0000.

**nnnn** denotes the hexadecimal character defined to replace an invalid DBCS code point.

The following example shows the syntax for the DEFAULT parameter.

```
DEFAULT=FFFF
```

## RULES

defines what constitutes a double-byte character for the defined language. RULES is only used when receiving a file from a platform other than z/OS or MVS, 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.

The following table identifies valid language options for the RULES parameter.

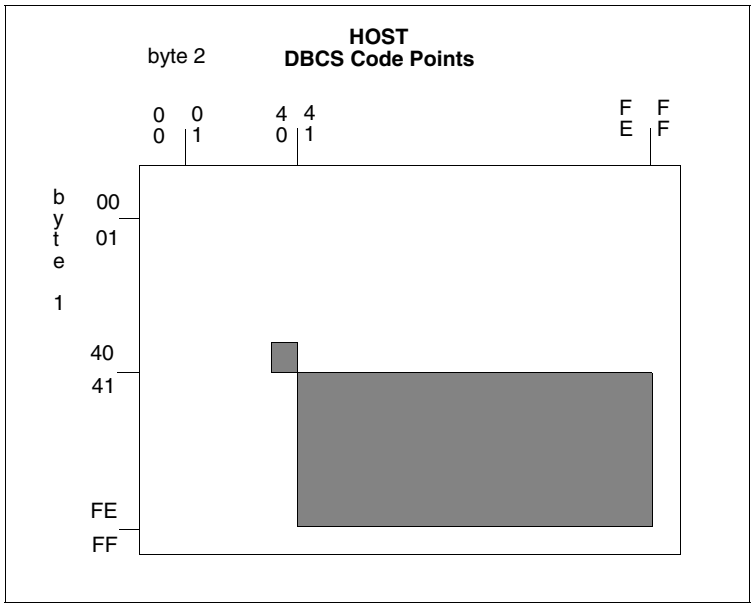
| Language Option | Range |
|---|---|
| KS5601 (Korean Standard) | x'A1'-x'AC'<br>x'B0'-x'FD' |
| KOREAN (Old Style) | x'81'-x'BF' |
| JAPANESE | x'81-x'9F'<br>x'E0'-x'FC' |
| CHINESE (Traditional/Simplified and 5550) | x'81'-x'FC' |
| BIG5 (Chinese) | x'A4'-x'C6'<br>x'C9'-x'F9' |
| x'01'-x'FF' | user selectable |

The following example shows the syntax for the RULES parameter.

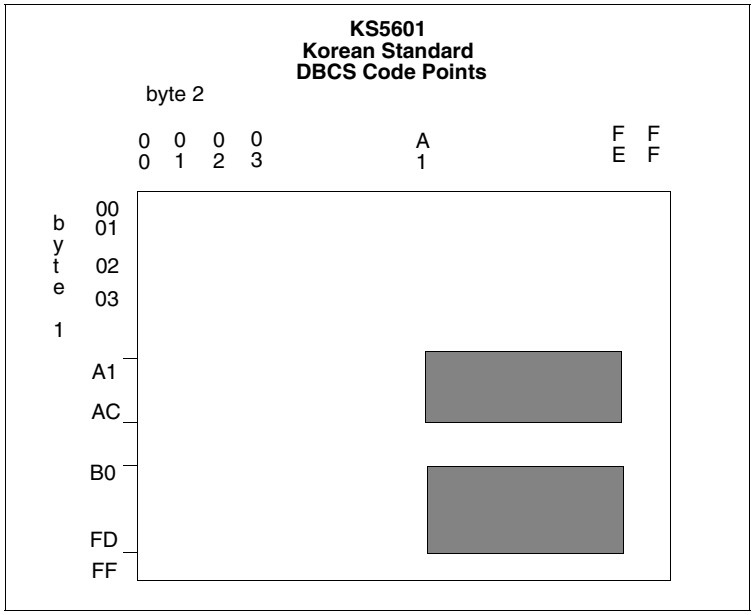
```
RULES
KS5601
END
```

The KS5601 language option is in the EBCXKSC table, which is provided in *HLQ*.SAMPLIB.

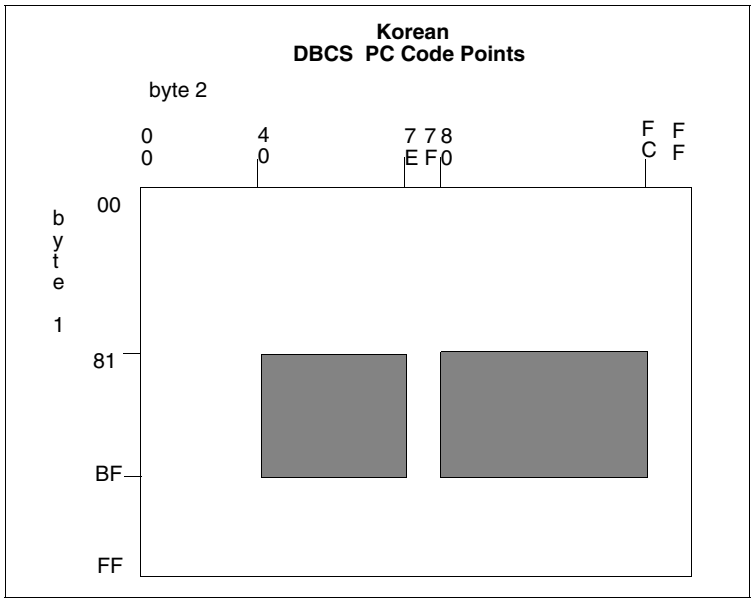
The following graphic represents the Connect:Direct hexadecimal DBCS code points.



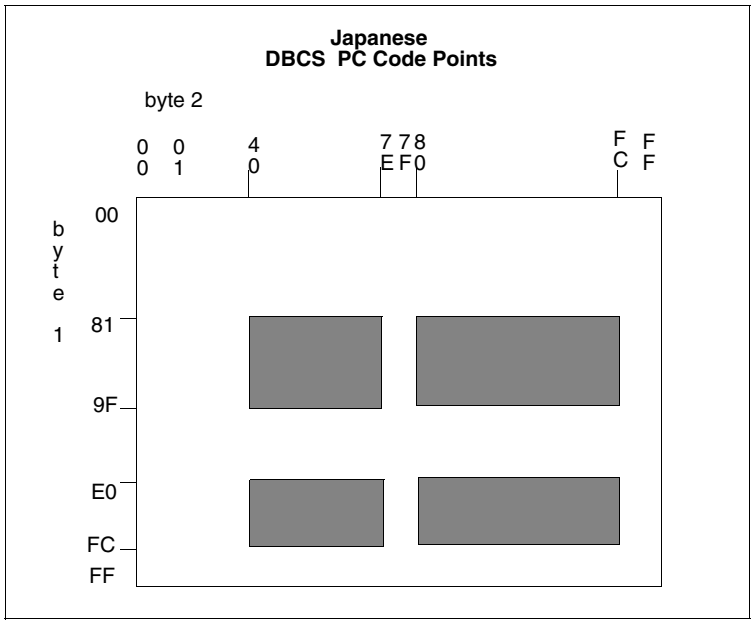

The following graphic represents the Korean Standard (KS5601) hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 323.



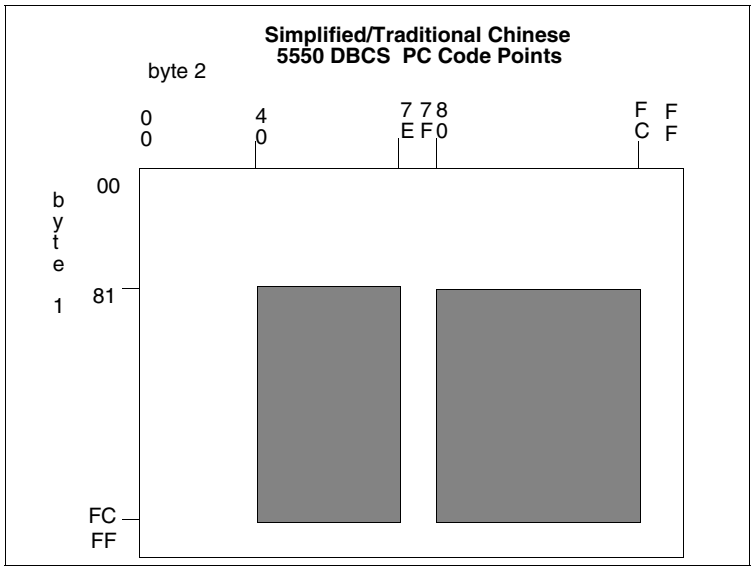
The following graphic represents the Korean hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 323.



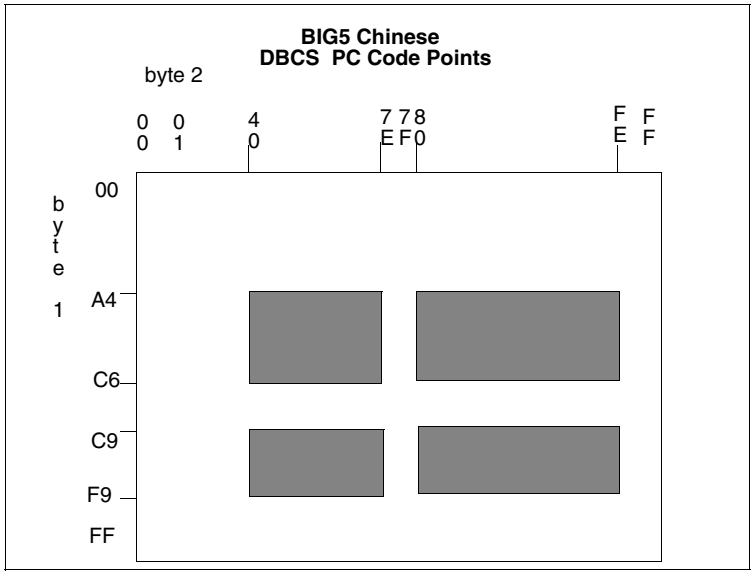

The following figure is a graphic representation of the Japanese hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 323.



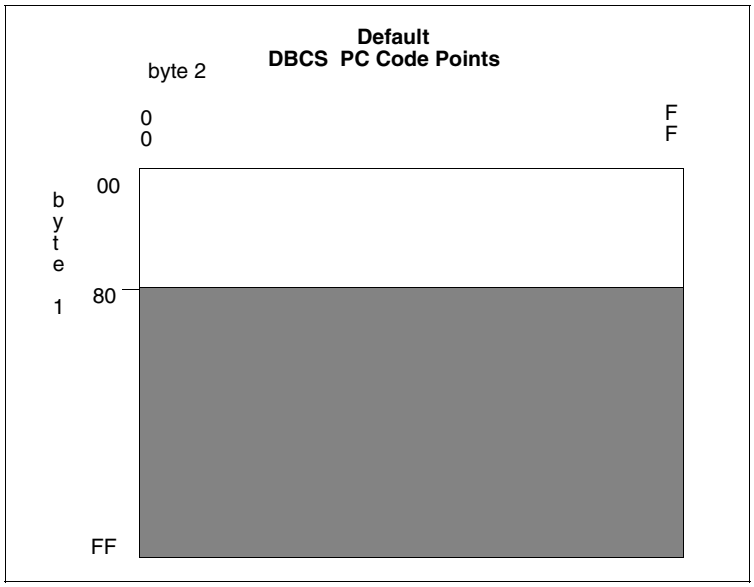
The following graphic represents the Traditional Chinese hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 323.




The following graphic represents the Chinese (BIG5) hexadecimal DBCS code points. The first character of each code point coincides with the range values in the table on page 323.



The following graphic represents the default hexadecimal DBCS code points.




## SBCS

creates the single-byte character set translation table. This table translates 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 you define SBCS, you must begin all data in column one and only one hexadecimal character pair is allowed per line.

The following 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 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 the 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 translates 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 translates 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. Following are some sample tables. Use these sample tables as a reference only. They do 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
D377,C8EF
D37B,C8F0
```

## Sample JCL to EXECUTE the Preprocessor

The following sample JCL executes the preprocessor against the input source. The output produced by the preprocessor is in assembler CSECT form and is input to the assembler. The assembled object is then link-edited to produce a load module.

The following JCL is contained in the member DBCSBLD $CD.JCL

```
//JOBNAME  JOB (ACCTN),'ADMINISTRATOR',CLASS=A,
//    REGION=4098K,MSGLEVEL=(1,1),MSGCLASS=X
//*****************************************************************
//*
//* JCL TO CREATE DBCS TRANSLATE TABLE
//*
//* REPLACE THE FOLLOWING ENTRIES IN THE PROCEDURE STATEMENT
//* BELOW WITH SITE DEPENDENT INFORMATION
//*
//* TABLE=    NAME OF THE SOURCE TRANSLATE TABLE
//*
//*****************************************************************
//BLDDBCS  PROC TABLE=XXXXXXXX,CDPREF='$CD',TEST=NOTEST,RENT=RENT
//*****************************************************************
//* STEP1 CREATE ASSEMBLER OUTPUT FROM PRE-PROCESSOR INPUT
//*****************************************************************
//STEP1    EXEC PGM=DMDBCSPR
//STEPLIB  DD  DSN=&CDPREF..LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//CDTABIN  DD  DSN=&CDPREF..SAMPLIB(&TABLE),DISP=SHR
//CDTABOUT DD  DSN=&&SRC,DISP=(,PASS),SPACE=(CYL,(1,1)),
//             DCB=(BLKSIZE=1600,LRECL=80,RECFM=FB,DSORG=PS),
//             UNIT=SYSDA
//*****************************************************************
//* STEP2 ASSEMBLE OUTPUT CREATE BY PRE-PROCESSOR
//*****************************************************************
//STEP2    EXEC PGM=ASMA90,
//             PARM='OBJECT,NODECK,XREF(SHORT),&RENT,&TEST,USING(WARN(0X
//             ),NOMAP),FLAG(NOCONT)'
//SYSLIB   DD  DSN=&CDPREF..SAMPLIB,DISP=SHR
//         DD  DSN=SYS1.MACLIB,DISP=SHR
//         DD  DSN=SYS1.MODGEN,DISP=SHR
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSLIN   DD   DSN=&&OBJ,UNIT=SYSDA,DISP=(,PASS),SPACE=(TRK,(5,5)),
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD  DSN=&&SRC,DISP=(OLD,DELETE)
//*****************************************************************
//*
```

Information on Connect:Direct Processes, including how to use the table created from the previous JCL sample using the COPY statement, is available from the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.

# Chapter 16

# Performance Tuning

This chapter provides general guidelines to help you evaluate the performance of your Connect:Direct system, along with other factors you can review and test to see the effects on the overall efficiency of your system. As in most computing environments, Connect:Direct is only one piece of your system consisting of other communication and network components along with other applications–all of which impact how the overall system runs. In addition, performance and security goals differ for each organization and reflect the tradeoffs an organization is willing to take.

The following topics are covered in this chapter:



## Analyzing TCP/IP Performance

Performance problems occur when a system and its network are not operating as effectively as they should as indicated by slow response times and a decrease in users' productivity. These problems can be intermittent or can indicate a growing strain pointing to capacity issues. Causes can be multifaceted and include both hardware and software origins or can be quickly solved with proper configuration settings.

Because of the multi-faceted nature of TCP/IP issues, this section serves as an analysis tool by providing a checklist of possible factors. By properly analyzing and defining the problem in terms of a set of symptoms and potential causes, you can either solve the performance problem yourself or can provide Sterling Support with documentation to pinpoint the problem and devise an action plan.

When you analyze performance, you must also consider your organization's priorities, such as the following goals:

- To maximize throughput in order to achieve the maximum data transfer rate
- To "fill the pipe" and run at full capacity

As you test and fine-tune settings to address specific factors in a problem area, record the results and note any unusual interactions or behavior. You may want to change company standards and create a checklist to accommodate new procedures or settings.

## General TCP/IP Problems

This table lists general TCP/IP problems and factors to consider

| Problem | Factors to Consider |
|---|---|
| Host Issues | ◆ Inadequate memory<br>◆ Slow disk speed/contention<br>◆ Slow channel speed/contention<br>◆ Excessive workload<br>◆ Inadequate processors/slow processor speed<br>◆ Inefficient performance groups and dispatch priorities<br>◆ Resource competition among applications on same system<br>◆ Resource competition among LPARs |
| Network issues | At the link level, look for:<br>◆ Link errors<br>◆ Hardware or interface errors<br>◆ Latency problems<br>**Note:** UDT is an alternative transport layer protocol to TCP that is designed for high latency with high-bandwidth connections. If you transfer files with a trading partner who is geographically distant from you and determine that the data congestion is caused by a high-latency, high-bandwidth connection, you may want to analyze your system and see if you would benefit from UDT. For additional information, see the *Connect:Direct for z/OS Release Notes*.<br>◆ Collisions<br><br>At the IP layer, look for:<br>◆ Discarded packets<br>◆ Reassembly failures<br>◆ Whether the DoNotFragment Bit is set<br>◆ TOS (TCP/IP Type of Service) such as Telnet with low delay (interactive priorities overriding batch transmissions)<br>◆ Small MTU/MSS (Maximum Transmission Unit/Maximum Segment Size)<br>**Note:** If the MTU is too small, inefficiency results whereas if it is too large, datagram fragmentation may result. |

| Problem | Factors to Consider |
|---|---|
| Network issues (cont'd) | At the TCP layer, look for:<br>◆ Segments retransmitted<br>◆ Connections reset<br>◆ Frequency of ACKs<br>◆ Window size too small<br><br>In the TCP/IP stack, look for:<br>◆ The maintenance level of the two TCP stacks involved<br>◆ The use of PORT and 1364 TCP CDSTC NODELAYACKS which may delay ACKS<br>◆ The values of TCPSENDBFRSIZE and TCPRCVBFRSIZE in the TCP/IP PROFILE data set<br>**Note:** These values affect all applications using the TCP/IP protocol whereas the V2.BUFSIZE initialization parameter (see below) affects the operation of the Connect:Direct application only.<br>◆ Whether the value set for PATHMTUDISCOVERY is an MTU size of 8992 (the default)<br><br>In the Connect:Direct global initialization parameters file, look at:<br>◆ V2.BUFSIZE–The default value of 4096 is too low for most communications lines which have greater bandwidth and speed.<br>◆ DEBUG–Make sure this setting is 00000000 so that internal traces are turned off. |

## Problems Involving Executing Connect:Direct Processes

This table lists factors to consider if a particular Connect:Direct Process is executing inefficiently.

**Note:** Because FTP is a utility program integrated into the TCP stack (thus running at the dispatch priority of the TCP stack), FTP may transfer data at a faster rate than an external application when sending a single file from the same source to the same destination. The only time an external data transfer application exceeds the transfer rate of FTP is when parallel data transfers take place between the same source and destination. Using parallel data transfers between the same source and destination is how most large production environments operate.

| Factors to Consider | Suggestions |
|---|---|
| Are you using compression? If so, what type? | ◆ Compression is generally unnecessary unless you use a slow line. Send data in its original state.<br>◆ If you use extended compression and want to see the effects of changing the default values for the parameters related to extended compression, see *Increasing Throughput and Decreasing CPU Utilization* on page 336. |
| Are you using checkpoints? If so, what is the interval? | On a fast link make this interval large, for example, 100M |
| Are you sending text or binary files? | When comparing Connect:Direct with FTP, send files ***only*** in Binary mode with both Connect:Direct and FTP. Binary mode must be used with Text files because FTP strips trailing blanks and sends only a partial file.<br>**Note:** Connect:Direct only sends complete files unless sending HFS files, where trailing blanks can be stripped.. |
| What does the file structure look like? | To speed up the transfer, use a larger blocksize. |
| Are you using striped extended-format data sets for files that have large amounts of data or in which time is of the essence? | Depending on the number of stripes, you could see a dramatic increase in the I/O rate. |
| Can you break down the Process so as to send multiple files at once using Connect:Direct's parallel session capabilities? | For testing purposes, set the PARSESS parameter in the network map to at least 10 then submit 10 file transfers in Connect:Direct and 10 in FTP. After verification, change the PARSESS value to fit your environment. |
| Are you changing DCB attributes? | Avoid giving the sending and receiving data sets different DCB attributes since that forces the transfer to "record mode," which increases CPU utilization and TCP or SNA I/O. |

# Improving BSAM Data Transfer Rates

To optimize BSAM sequential data set transfer rates, you can take one or more of the following approaches:

- If MAXSTGIO is currently defined in the initialization parameter file, review the setting and consider setting it to the 1 MB default or greater. To fine-tune and set the Number of Channel Programs (NCP) in the DCB parameter of the COPY statement, include the second positional parameter as well.
- Increase the block size when it is advantageous to do so. Make the block size of a disk data set close to (but not more than) half-track blocking (27998 for non-extended 3390 disk data sets or 27966 for extended 3390 data sets). This improves performance by increasing the number of bytes transferred per I/O. For example, when transferring a data set with an LRECL of 80, it takes much longer to transfer 27920 bytes in 349 blocks (BLKSIZE=80) than in 1 block (BLKSIZE=27920).

**Note:** Exceeding half-track blocking on disk wastes a significant amount of storage capacity without improving the transfer rate.

For tape-to-tape transfers, a larger block size improves both performance and capacity. For disk-to-tape transfers, the I/O performance benefit of reblocking to an LBI block size (> 32760) may be outweighed by the CPU performance hit of transferring the data set in "record mode."

## Troubleshooting BSAM Data Transfers

Data transfer rates using BSAM vary significantly from run to run even on the same system. For example, even with a high NCP, the transfer rate deteriorates when the I/O subsystem is moderately busy. If problems should occur, review these factors:

- The REGION specified on the job card. With the higher number of I/O buffers comes the risk that if too many Processes run simultaneously, the above-the-line storage can be exhausted. To prevent out of storage abends, review both the MAXSTGIO initialization parameter and the job's region.
- The number of concurrent transfers occurring on one DTF, even when all system components (CPU, DASD, CU, CHPID or network) run below capacity. For example, where a single transfer attains a transfer rate of 76 MB per second, two concurrent transfers potentially reduce it to 66 MB per second for each transfer, three transfers to 58 MB per second, and 4 transfers to 52 MB per second.
- Hardware caching. When you run the same test case multiple times, usually the first runs slower than subsequent runs. For example, the initial transfer rate might be 38 MB/second, which increases to 75 MB/second for the second and subsequent runs.

- Network. The transfer rate varies widely according both to the speed of the network and the volume of traffic on it.
- Data set's device, CU, and CHPID configuration, speed, and how busy they are.
- CPU speed, and how busy it is (sometimes a limiting factor).
- Compression. Compression sometimes slows down the transfer due to extra CPU use.

### Problems Involving Checkpoints

If the value specified for the checkpoint interval either as an initialization parameter or in the TO clause of a COPY statement is too small, it can significantly reduce transmission speed. Since the purpose of checkpointing is to save time in a restart, it is usually unnecessary to have a checkpoint interval that translates to less than a second of transmission time. A checkpoint interval that translates to 5 seconds of transmission time between checkpoints is normally more than adequate. For example, if the transmission rate to the other node is typically 10 MB/second, and you desire to lose no more than 5 seconds of transmission time in a restart, then you would set CKPT=50M.

**Note:** For additional information on the CKPT parameter in the COPY statement, see the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.

# Increasing Throughput and Decreasing CPU Utilization

Depending on your company's computing environment, extended compression can be used to increase throughput while decreasing the amount of time it takes to transfer data. In addition, if you are using Secure+ Option, there are other factors you should take under consideration to fine-tune performance in this area.

## Using Extended Compression

TCP/IP connections accommodate greater bandwidth to transfer large files than older technologies eliminating the need to compress data. Data can be sent in its original state saving both time and the need to decompress data once it has been transferred. However, if you have a slow line and high CPU capacity, compression may be warranted. Under other conditions, compression consumes a lot of CPU and slows transfer rates considerably.

### Different Methods of Using Extended Compression

You can compress and store files in ZLIB-compressed format using one of the following methods:

- On a global basis using the ECZ initialization parameters. If you always transfer the same type of data, you may benefit by changing the global default values of the extended compression initialization parameters. See *Changing the Values of Extended Compression Initialization Parameters* on page 337.

- On a Process basis using the EXT parameters in the COPY statement. If you send a variety of data types, it is probably more efficient to retain the default values of the initialization parameters and override them on a Process-by-Process basis using the COPY statement. For information on overriding the extended compression parameters in the COPY statement, refer to the Process Guide Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.
- Through the CDSACOMP batch utility, which does not consume as many CPU resources as the online methods.

You can also use the CDSACOMP batch utility to decompress the data and store it in the original format if you are on the remote node to which compressed data has been sent. The CDSACOMP utility also produces a report, which shows how much the data was read, written, compressed, and how long it took to compress so you can determine the benefits of changing the default values of the extended compression parameters.

### Changing the Values of Extended Compression Initialization Parameters

The effects of changing the default values for the ECZ.COMPRESSION.LEVEL, ECZ.MEMORY.LEVEL, and ECZ.WINDOWSIZE extended compression parameters are not always predictable and can significantly increase CPU utilization. The default values for the three parameters produce very good results for a wide variety of data types. Typically, it is only beneficial to change these default values if line speeds are limited, data is repetitive, and CPU is available.

## Considerations When Using Secure+ Option

When using Secure+ Option, be aware of the following :

- CPU utilization increases dramatically with every increase in the length of the encryption key. Use the lowest level of encryption allowed by your security policy.
- Whenever possible, use an encryption key that is supported in the z/ hardware (3DES or AES128).
- Even though extended compression is not recommended for high speed networks, using extended compression with files that compress well (80-90%) can reduce total CPU utilization, especially if the encryption key is not implemented in the hardware.
- If Secure+ Option is being used between two Connect:Direct for z/OS nodes (Version 4.7 or later), but not all files must be encrypted, consider using OVERRIDE=YES on the remote node record in the Secure+ Option parameter file and SECURE=OFF in the PROCESS statement.

# Chapter 17

# Isolating Problems

This chapter helps you isolate problems or errors found while using Connect:Direct for z/OS. The following topics are discussed:

- Problem reporting procedures including supporting documentation you need to diagnose problems and provide Sterling Commerce Customer Support
- Basic troubleshooting methods
- Common error types along with causes of the errors, types of data to collect and action to take for problem resolution
- Diagnostic tools for determining and resolving problems including the use of debug settings and DD statements in the startup JCL to produce traces

## Problem Reporting Procedures

This section describes the following:

- Contacting Customer Support
- Gathering data for Customer Support
- Resolving the problem
- Escalating a problem situation

### Contacting Customer Support

Sterling Commerce provides a central Customer Support department responsible for Connect:Direct technical support. Refer to the Sterling Commerce Web site at www.sterlingcommerce.com for information about how to obtain assistance. You must have a Support On Demand user name and password for access to the information and services provided on the Sterling Commerce Customer Support Web site. For information on obtaining a user name and password, see the *Connect:Direct for z/OS Release Notes*. We suggest that one person from each customer site serve as the contact for all Connect:Direct problem reporting.

## Gathering Data to Isolate the Problem

To expedite the diagnostics process, prepare the following information for the Customer Support representative before calling Sterling Commerce:

- Version, release, maintenance level, and operating system for the Connect:Direct products being used on both nodes, for example, Connect:Direct for z/OS 4.6.00 PUT4601 with Connect:Direct for HP NonStop 3.4.00.
- Release levels of the operating system, VTAM, and other software involved
- Details describing the complete scenario, including:
  - Commands that are issued, with exact syntax and order
  - Files or devices that are involved. Note the file contents and the type of file (such as DCB file attributes, GDG, Tape, SMS, striped, compressed)
  - Interface that you used, such as batch interface, IUI, or operator interface
  - Error messages for *both* nodes
  - System logs for *both* nodes, including any messages generated while the problem occurred. Check the SYSLOG for Connect:Direct for z/OS nodes
  - Which side is PNODE/SNODE
  - The direction of the data transfer
  - What connection protocol you are using (LU0, LU6.2, IBM TCP/IP, TCPAccess)
  - If a Connect:Direct for z/OS node is a Connect:Direct/Plex or stand alone server
  - Connect:Direct Process including the FILE attributes for the FROM and TO files for both nodes
  - I/O device types
  - Network map information
  - Statistics information for both nodes including ALLOC information

**Note:** Be prepared to recreate the problem. Problem recreation is the responsibility of the customer as Sterling Commerce cannot duplicate all customer environments.

### Additional Supporting Data

Problem determination can be involved and can require extensive research. We ask you to gather some of the following supporting documentation to help analyze the problem. Not all of this information is applicable to all problems or all operating environments.

- SYSLOGs
- Connect:Direct statistics for both nodes
- RPLERRCK output
- ESTAE output
- VTAM definitions

- APPLID definitions
- Logmode table entry
- Class of Service (COS) entry
- D NET VTAM display
- Connect:Direct traces
- Network map information

You can use your network management software or VTAM commands to isolate any session-related problems.

## Providing Dumps to Customer Support

A Connect:Direct ABEND can occur when a system failure or system error exists or when the FORCE parameter is used with the STOP CD command.

When an ABEND is reported, Customer Support searches the problem tracking database for any similar problems. Often the ABEND is a known bug or a common error, and a solution is readily available. Otherwise, they need a full SVC dump for diagnosing ABENDs.

The Sterling Commerce Customer Support Web site at www.sterlingcommerce.com is the doorway to Web support, information, and tools. Refer to the Customer Support Reference Guide available from the Sterling Commerce Customer Support Web site for specific information on getting support for Sterling Commerce products. Another Self Support Tool you can use to exchange diagnostic information and solutions with Customer Support is SupportXChange, which is also available on the Sterling Commerce Customer Support Web site.

When you call to report an ABEND, Sterling Commerce Customer Support attempts to determine the type of ABEND and locate the module in which the ABEND occurred. If support personnel cannot locate a reference to the ABEND, they might request the following to diagnose the ABEND:

- Send a complete SVC dump, not a snap dump.
- If the ABEND is caused by a specific Connect:Direct Process, support personnel may request a copy of the Process and the statistics associated with that Process. If the Process has symbolics, include the symbolic substitution data. Also, ensure that the statistics records include WTO records.

**Note:** Include statistics from both Connect:Direct nodes.

If the ABEND occurred while executing a Process that has previously executed successfully, determine what changes were made, either in the operating system or within Connect:Direct, since executing and do the following:

- Send console logs for both Connect:Direct nodes.
- Note whether the ABEND caused either Connect:Direct node to terminate.
- If the ABEND can be recreated, provide details.

- Send a copy of the system log and network error log for both operating systems, which can indicate any unusual situations occurring with the operating system or network at the time of the ABEND.

**Note:** A system log is always required when analyzing an ABEND. It is preferable to review the log for both systems; however, it is essential for the system reporting the ABEND. When one of the nodes is not in a z/OS, check the output files for Connect:Direct.

- Send RPLERRCK DD output to review I/O errors and other information. See *DD Statements in Startup JCL* on page 373 for more information.
- Send ESTAE DD output to review ABEND conditions and some special I/O errors. See *DD Statements in Startup JCL* on page 373 for more information.

If several ABENDs occur simultaneously, send the dump from the first ABEND only. Subsequent ABENDs are usually a result of the original ABEND.

When sending a dump on tape, send the JCL that created the tape. DSN, VOLSER, LABEL, and DCB attributes are needed to facilitate tape unloading. If available, send a printout of the tape management product display of the VOLSER.

**Note:** If you are sending a tape with more than one file, ensure that the JCL that created the tape references the correct file in the LABEL= parameter. This reference ensures that a previous file is not inadvertently overlaid.

You can send dumps to Sterling Commerce on tape or over the Internet. To send dumps over the Internet, contact Sterling Commerce Customer Support for instructions.

The remainder of this section discusses the various dumps that might be requested by Sterling Commerce Customer Support for problem determination and resolution. Also provided are instructions for producing these dumps. Dumps described in this section include:

- Connect:Direct Data Transmission Facility dumps, which include SYSMDUMP and CDSVCDMP dumps
- Connect:Direct Interactive User Interface dumps, which include z/OS Time Sharing Option (TSO) address space dumps
- Batch dumps
- IBM VSAM file dumps

### Connect:Direct Data Transmission Facility (DTF) Dumps

A Connect:Direct Data Transmission Facility (DTF) dump is generated when an ABEND occurs. The dump contains the contents of the Connect:Direct address space, which is copied into one or more of the data sets that you specify by ddname in the DTF JCL (CDSVCDMP, SYSABEND, or SYSMDUMP).

When an ABEND occurs, Connect:Direct produces an SVC dump with all of the information for the address space, regardless of whether the ABEND occurred in the main task or a User Exit subtask. Unless otherwise specified in the JCL, the dump is written to the standard SVC dump data

set, SYS1.DUMPxx. You can specify an alternate data set by using the CDSVCDMP DD statement in the JCL.

**Note:** If the system attempts to take an SVC dump and fails with: *ERROR* Unable to take an SVC dump; reason: 0B, the Dump Analysis and Elimination (DAE) component of the operating system found an earlier dump of this problem already exists.

You can turn off the SVC dump by modifying your JCL. For more information, see *Turning Off the SVC Dump* on page 343.

### Capturing Multiple DTF SVC Dumps Using CDSVCDMP

You can use CDSVCDMP to capture multiple DTF SVC dumps. A dump is created for each ABEND and written to a separate data set. You can specify whether the dumps are written to the SYS1.DUMPxx data sets or to unique user-specified data sets.

To write dumps to user-specified data sets, you must create a data set name for each dump that can occur. The first data set name must end with .SYSMDP00. Each additional data set name must end with .SYSMDPnn, where nn is a consecutive number up to a value of 99. For example, if you want to create enough data sets to write five dumps, create five data set names beginning with xxxx.yyyy.SYSMDP00 and ending with xxxx.yyyy.SYSMDP04.

You must define the data sets with the same attributes as your SYS1.DUMPxx data sets. The data sets must be preallocated and on the same disk volume.

To record multiple SVC dumps, set the JCL statement for CDSVCDMP as follows.

```
//CDSVCDMP DD DSN=XXXX.YYYY.SYSMDP00,DISP=SHR
```

The first dump is written to the .SYSMDP00 data set. When an additional ABEND occurs, a dump is written to the next data set, .SYSMDP01. Each additional ABEND creates a dump to the next .SYSMDPnn data set as long as ABENDs occur and enough .SYSMDPnn data sets are available. If an ABEND occurs and all data sets are full, the dump is not created and a message is issued stating that all .SYSMDPnn data sets are full.

You do not have to empty or reset these dump data sets. When the DTF is initialized, and you are using the .SYSMDPnn data sets, Connect:Direct writes over the existing data in the data sets. If you want to save the existing data, save the data sets using a different data set name before you restart Connect:Direct.

### Turning Off the SVC Dump

If you want to turn off the SVC dump, place the following statement in your DTF JCL.

```
//CDSVCDMP DD DUMMY
```

### Changing Dump Options

For the SDATA parameter specify SDATA=(ALLSDATA). If SDATA=(ALLSDATA) is not an acceptable default for your system, ensure that the PARMLIB member corresponding to the ddname in the JCL for the DTF specifies, at a minimum, the following.

```
SDATA=(NOSUM,PSA,RGN,SQA,SWA,TRT,LPA,GRSQ,CSA,NUC)
```

For the PDATA parameter for IEADMPxx and IEAABDxx, specify PDATA=(ALLPDATA).

**Note:** PDATA is not an option for member IEADMRxx.

If you cannot specify ALLPDATA, at a minimum, include PSW, REGS, SA, JPA, SPLS, and SUBTASKS. Refer to the IBM manual *MVS Initialization and Tuning Reference* for the release of z/OS being used.

If you are unable to change the PARMLIB member, issue an operator command to change the dump options.

The following steps guide you in issuing operator commands to change dump options:

1. Issue the command DISPLAY DUMP,OPTIONS to list the dump options currently in effect.
2. Issue the command CHNGDUMP SET to change the options.
3. Issue the CHNGDUMP DEL or CHNGDUMP RESET command to reset the options after recreating the dump.

If the correct dump options are specified in one of the PARMLIB members, change the ddname in the JCL for the DTF to reference the ddname corresponding to that member.

For further information on changing dump options, refer to the IBM manual *MVS System Commands* for the z/OS being used.

## Connect:Direct Interactive User Interface (IUI) Dumps

When the Connect:Direct IUI ABENDs, a system dump might be required for problem resolution. To obtain a dump of the TSO address space, allocate a SYSMDUMP DD to an appropriately defined data set, and run the ISPF with the TEST option. To send a dump of the IUI to a data set, allocate SYSMDUMP with DISP=MOD to account for the two dumps that Connect:Direct produces.

To recreate an ABEND of the IUI to produce a dump, perform the following steps:

1. Ensure that the SYSMDUMP DD is allocated.
2. Reinvoke ISPF with the **ISPF TEST** command.

3. To allocate a SYSMDUMP DD, type the following TSO command where *filename* is a valid DSN name that has LRECL=4160, BLKSIZE=0, and RECFM=FBS.

```
TSO ALLOC F(SYSMDUMP) DA(filename) MOD
```

   The system will determine blocksize, and will prevent short blocks. Generally, 110 cylinders on a 3390 device is enough space for an IUI dump.

4. When the ABEND message is displayed, press **ENTER** to produce a dump. Two dump messages (and dumps) are produced. Press **PF3**/**END** to bypass the dump.

### DMBATCH Dumps

For DMBATCH dumps, ensure the JCL that executes DMBATCH contains a SYSMDUMP DD statement. Refer to the discussion on DTF dumps beginning on page 342 to ensure that the appropriate dump options are specified for the corresponding ddname in the JCL for DMBATCH.

### VSAM File Dumps

If a problem occurs when using a Connect:Direct VSAM file, you might need a dump of the VSAM file for problem resolution. To dump a Connect:Direct for z/OS VSAM file, use the IBM Access Method Services (IDCAMS) PRINT command. If you send an IDCAMS PRINT of the TCQ, include the TCX file.

Refer to the sample job stream in the following example for creating a Connect:Direct for z/OS VSAM dump.

```
//XXXXXX   JOB  (1111),PROGRAMMER,NOTIFY=TSOID,CLASS=D,MSGCLASS=X,
//             MSGLEVEL=(1,1)
//PRINT    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  PRINT INDATASET(DUMP1.ZOS.VSAM1)                                -
           DUMP
```

To copy a VSAM file to tape, use the IDCAMS REPRO command. You must include the DCB parameters, RECFM=VB, DSORG=PS for the data set created on the tape. In the LRECL and BLKSIZE parameters for the dataset created, you must reflect the size specified for the RECORDSIZE parameter used when you defined the file.

For example, if the Connect:Direct statistics file was initially defined with RECORDSIZE (4089 4089), copy the Connect:Direct statistics file by typing the following DCB parameters in the DD statement in the JCL.

```
//OUTDD DD DSN=TAPE.STAT.FILE,
  DCB=(LRECL=4089,BLKSIZE=4089,RECFM=F,DSORG=PS)
```

### Suppressing Dumps for Specific ABEND Codes

Use the following procedure if you want to suppress the dump for specific ABEND codes.

1. Add the ABEND code for which you want to suppress dumps to the ABEND codes list defined in the ABEND.CODES.NODUMP parameter.
2. Set the ABEND.RUNTASK parameter to ABEND.CODES.NODUMP. Following is an example:

```
ABEND.CODES.NODUMP => (SX37 SX13 U0728 SXD9 S9FC)
ABEND.RUNTASK => ABEND.CODES.NODUMP
```

## Resolving the Problem

If Sterling Commerce Customer Support cannot resolve the problem during your initial call, the problem is documented as a case in a problem-tracking system and is assigned a reference number. Refer to this number when forwarding any supporting documentation or calling for updates.

| Severity Level | Definition |
|---|---|
| Severity 1 | Production system is down; requires immediate attention. For example, excessive abnormal termination. |
| Severity 2 | A major problem or question; the product operates but is severely restricted. For example, an incorrect response from a frequently used command. |
| Severity 3 | A non-critical issue is encountered with the product, but the majority of functions are still usable. For example, an incorrect response from an infrequently used command. |
| Severity 4 | A minor problem or question that does not affect product function. For example, the text of a message is misspelled. |

All cases are prioritized based on severity, as follows: During or immediately after the initial issue is reported, a Sterling Commerce Customer Support representative requests appropriate information needed to research your case. Every effort is made to minimize requests for more information. Trained resources research and resolve cases on a timely basis. You are informed of the resolution status at agreed-upon intervals while the case is still open.

When the problem is resolved and you have accepted the resolution, the case is closed. Brief service evaluation surveys are sent to you occasionally so that we can obtain your feedback.

## Escalating a Problem Resolution

If our normal support cycle does not produce the results you require or your issue has changed in severity, you can escalate the case. To escalate a case, contact the technician responsible for your problem, state your concerns, and request that the case be escalated.

# Basic Troubleshooting Methods

This section describes the basic troubleshooting methods that you can use for problem isolation. Preliminary actions you can take to isolate the problem include:

- Reviewing Connect:Direct messages
- Examining output from Connect:Direct SELECT commands
- Verifying file attributes
- Overriding Connect:Direct initialization parameters

## Reviewing Connect:Direct Messages

As you research a problem, note any messages issued by Connect:Direct. Look specifically for messages displayed by the API through the IUI or batch interface (DMBATCH). If an error occurred in a file transfer, look at messages for any Processes executing at that time for both nodes.

Also, check for messages in the JES log for z/OS.

Connect:Direct messages contain short and long text to explain the error. Connect:Direct for z/OS displays the short text when an event occurs. The following information is displayed for each message if you request the long text:

- Module issuing the message
- Short and long message texts
- System action as a result of the message situation
- User response to correct the situation

**Note:** For z/OS, press **PF1** to display the longer explanation if the message is issued in an ISPF panel.

### Interactive Use of the Connect:Direct Message Facility

For Connect:Direct for z/OS, display the long text message through the IUI by either of the following methods:

- Use the SELECT MESSAGE command in the command line interface.
- Type **M** at the CMD prompt and press **Enter**.
- Access the PRIMARY OPTIONS MENU and type the MSG option at the CMD prompt.
- Type **=M** from any Connect:Direct screen at the CMD prompt in the ISPF Interface and press **Enter**. The long text for the current message is displayed.

### ABEND Messages

Some user ABEND messages are stored in the Connect:Direct message file. To access them, insert the ABEND message ID in place of a Connect:Direct message ID.

For example, to display the description of user ABEND U0075 perform the following:

1. Access the PRIMARY OPTIONS MENU in the IUI.
2. Select the MSG option.

3. Select option 1.
4. Type **U0075** on the message ID line and press **Enter**.

```
                    Connect:Direct MESSAGE DISPLAY
                                                              DATE => mm.dd.yyyy
                                                              TIME => hh:mm
MESSAGE ID==> U0075
MODULE    ==> DMINIT
================================================================================

SHORT TEXT==> Connection services initialization failure.
LONG TEXT:
   LINE 1 ==> An error has occured while initializing VTAM connection
   LINE 2 ==> services. ESTAE output should give a return code and error
   LINE 3 ==> flag.
   LINE 4 ==>
   LINE 5 ==> Verify the applid does not have a password associated with it
   LINE 6 ==> in the LOCAL.NODE definition of the Network Map.
   LINE 7 ==>
   LINE 8 ==> SYSTEM ACTION: ABEND the intialization of the DTF.
   LINE 9 ==>
   LINE 10==> RESPONSE: Verify that the APPLID is varied active and there is
   LINE 11==>           not a VTAM password specified in the Network Map.
   LINE 12==>

COMMAND  ===> _________  ENTER 'DIR' TO DISPLAY THE DIRECTORY
```

System ABEND messages appear in the:

- JES log for z/OS

They do not appear in the Connect:Direct message file. Refer to the appropriate operating system manuals if you require further details.

## Examining Output from Connect:Direct Select Commands

Problem determination and resolution is often as simple as gathering output from the Connect:Direct SELECT commands and examining the output for obvious errors. A description of the commands follows:

| Command | Description |
|---|---|
| SELECT NETMAP | Displays or prints the definitions of nodes with which Connect:Direct can communicate. |
| SELECT PROCESS | Displays or prints information about Connect:Direct Processes in the TCQ. You can use this command before or during Connect:Direct Process execution. |
| SELECT STATISTICS | Displays or prints statistics from the statistics log. Use this command only after executing a Connect:Direct Process. |
| SELECT TASK | Selects, displays, or prints the list of all active users, Processes, and tasks within Connect:Direct. |
| SELECT TYPE | Displays or prints type records from the Type file. |

All of the Connect:Direct SELECT commands are documented in the *Connect:Direct for z/OS User's Guide*.

## SELECT STATISTICS Command

The SELECT STATISTICS command is one of the most useful commands for problem determination. This section provides a brief overview of the SELECT STATISTICS command. For more detailed information, refer to the *Connect:Direct for z/OS User's Guide*.

If an error occurred during a file transfer, issue the SELECT STATISTICS command on both the PNODE and SNODE to review the statistics for the Process. Process statistics are stored in the Connect:Direct statistics log. You can request them by either using the IUI Select Statistics (SS) screen, if applicable, or issuing the Connect:Direct SELECT STATISTICS command.

The statistics log records the following types of information for Connect:Direct Processes:

- Function requested (COPY, RUN TASK, RUN JOB, SUBMIT)
- Process name and number
- Start and stop times, and date of the function
- Completion code
- Messages associated with the Process
- Location and ID of the user requesting Connect:Direct services
- Sending and receiving file names
- Amount of data sent and received
- Security violations
- All Connect:Direct WTO messages, allocation information, and mount requests

---

**Note:** WTO messages are created during Connect:Direct Process execution to document the execution steps and are stored in the Connect:Direct statistics file. This type of message is an excellent debugging tool for determining a Connect:Direct Process execution failure.

---

The optional parameters associated with the SELECT STATISTICS command enable you to define the search criteria and the form in which the report is presented.

### Connect:Direct IUI Select Statistics Screen

The Select Statistics screen provides a convenient, easy method for issuing the SELECT STATISTICS command. Use the fields to specify statistics selection criteria. The selection criteria enable you to determine what records to select from the statistics log, limit the statistics to a certain period of time, or limit the statistics to a certain Process. The selection criteria also allow Connect:Direct to select the requested statistics from the file more efficiently.

The SELECT STATISTICS screen displays general Connect:Direct Process information.

- To display the requested information in a formatted report, type a D in the CMD field and press Enter.
- To print the formatted report, type a P in the CMD field and press Enter. The output is sent to your TSO sysout.

- For a summary report display, type an S in the CMD field and press Enter.
- To view your statistics file in an unformatted display, type FIL or F in the CMD field and press Enter.
- The SELECT STATISTICS screen also enables you to exclude certain types of information from being displayed. Type a Y in any of the following fields and press Enter.
  - **MEMBERS** - Statistics generated for each partitioned data set (PDS) member
  - **WTO** - WTO (Write to Operator) messages
  - **WTO** - All statistics other than WTO messages

  When statistics are requested by Sterling Commerce Customer Support, do not exclude any information. Set all the fields listed to **N**. Customer Support must receive all the information available to solve the problem.

### Verifying File Attributes

If a problem occurs during a file transfer, verifying the accuracy of the file attributes can often resolve the problem. Review the following file attribute parameters for the file being transmitted when the error occurred. Check both the input and output files specified in your Connect:Direct Processes:

- Logical record length (LRECL)
- Block size (BLKSIZE)
- Record format (RECFM)
- File organization (DSORG)
- File disposition (DISP)
- Unit containing the file (UNIT)
- Volume serial number (VOLSER)
- Storage for allocating new files (SPACE)
- Optional processing code (OPTCD)
- Length of keys used in file (KEYLEN)
- Number of blocks or tracks to search for available space (LIMCT)
- SMS options and ACS rules

These parameters are described in detail in the information available from the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.

## Common Errors

This section describes the following types of common errors:

- Signon and IUI/API errors

- Connect:Direct DTF session establishment errors
- Connect:Direct DTF out-of-storage ABENDS
- Allocation and open errors
- Transmission errors
- Operator interface errors

**Note:** For information on initialization and license key errors, see the appendix in *Connect:Direct for z/OS Installation Guide* that deals with errors that can occur when you start up. For errors related to security, see *Troubleshooting Security Errors* on page 91.

For each type of error, information on probable causes, actions to take, and data to collect is provided.

## Signon and IUI/API errors

Signon errors keep you from accessing Connect:Direct. IUI/API problems prevent you from successfully submitting a Process or executing a command.

This section describes the following types of signon and IUI/API errors:

- ISPF signon failures can be caused by a variety of problems. Errors signing onto the IUI through the ISPF interface can include VTAM problems, security problems, or ISPF problems. Refer to *Troubleshooting Security Errors* on page 91 for a description of common security errors.
- IUI/API Connect:Direct session failures occur when the IUI/API cannot establish a session with the DTF. A session failure usually means you will be unable to sign on to Connect:Direct.
- SELECT command errors occur while issuing SELECT STATISTICS, SELECT PROCESS, SELECT NETMAP, SELECT USER, SELECT TYPE, and SELECT TASK commands from the IUI, batch, or operator interface

**Note:** You can diagnose most IUI/API problems by running an API-to-DTF session trace. See information on the NDMCMDS trace in *Connect:Direct Automatic Traces* on page 373.

### Condition: Signon to IUI Denied

Your signon to IUI is denied with the message *Error during ACB open*.

| Error | Cause | Action | Collect |
|---|---|---|---|
| SVTB002I | The interactive applid that Connect:Direct is trying to use for signon is not active, is in an unacceptable state, or is not correctly defined to VTAM. | Review both the short text and long text Connect:Direct messages. Allocate NDMCMDS to display error messages to your terminal. NDMCMDS shows all actual Connect:Direct commands issued to the DTF, including resolution of symbolics. It can be particularly helpful to debug Connect:Direct commands through the IUI or through DMBATCH if you are having signon problems, syntax errors, and so forth.<br>◆ For Connect:Direct for z/OS, type the following on your command line:<br>`TSO ALLOC FI(NDMCMDS) DA(*)` | ◆ NDMCMDS output<br>◆ Applid status display<br>◆ Applid definitions<br>◆ Network map |

### Condition: Signon to IUI Denied - No Error Message

Your attempt to sign on to the IUI is denied, but no error message is displayed.

| Error | Cause | Action | Collect |
|---|---|---|---|
| None | Your TSO or TSS profile specifies the NOWTPMSG option, which inhibits some error output from being displayed to the terminal. | Review both the short text and long text Connect:Direct messages. For z/OS, change to the WTPMSG option by typing TSO PROF WTPMSG. With the WTPMSG option, error messages are displayed at the terminal. Retry the operation (sign on). | ◆ None |

### Condition: Signon Denied - Connect:Direct not Active

Your signon is denied.

| Error | Cause | Action | Collect |
|---|---|---|---|
| SVTB004I<br>SCIA011I | The Connect:Direct you are attempting to sign on to is not active. | Review both the short text and long text Connect:Direct messages. Ensure that Connect:Direct has completed initialization before attempting a signon. Allocate NDMCMDS to display additional information about the session failure. NDMCMDS shows all actual Connect:Direct commands issued to the DTF, including resolution of symbolics. It can be particularly helpful to debug Connect:Direct commands through the IUI or through DMBATCH if you are having signon problems, syntax errors, and so forth.<br>For z/OS, check to see that the network map is specified correctly on the ISPF menu and that the network map is correctly loaded, as described in the installation guide for your product.<br>Try to sign on through the DMBATCH interface to isolate the problem | ◆ NDMCMDS output<br>◆ Network map<br>◆ Connect:Direct initialization parameters<br>◆ For z/OS the ISR@PRIM panel |

### Condition: Signon Denied - Users Exceeds Limit

Signon is denied.

| Error | Cause | Action | Collect |
|---|---|---|---|
| STAA004I | The number of interactive users on Connect:Direct has reached the limit set in the MAXUSERS parameter. | Review both the short text and long text Connect:Direct messages. Check the MAXUSERS parameter in the Connect:Direct initialization parameters data set. If it is commented out, the default is six users. Report this error to your Connect:Direct administrator, and determine whether you need to increase the value of this parameter. | ◆ Connect:Direct initialization parameter |

### Condition: SELECT Command Issued Successfully - No Output Produced

The SELECT command is issued successfully and completes with a successful return code and message, but no output is produced.

| Error Messages | | | | | |
|---|---|---|---|---|---|
| SAFF000I | SAFF014I | SAFK000I | SAFL000I | SAFL010I | SCBB000I |
| SCBL000I | SCBO000I | SCBP000I | SCBQ000I | SCBX000I | SOPA000I |
| SOPA011I | SOPE000I | SOPS001I | | | |

| Cause | Action | Collect |
|---|---|---|
| It is likely that Connect:Direct is having trouble allocating the temporary data set that contains the output from the SELECT command. | Review both the short text and long text Connect:Direct messages. For Connect:Direct for z/OS, specify a UNIT and VOLSER for the temporary data set, and SPACE information. You can specify UNIT and VOLSER on the SIGNON DEFAULTS panel of the IUI or use the TEMPDSN parameter on your signon command for batch. Note that a UNIT type of VIO is not acceptable. | ◆ List of the SELECT commands that produce output and those not producing output<br>◆ Any error messages |

### Condition: SELECT Commands - No Output Available

SELECT commands return with no output and a message indicating no output was available from the command.

| Error | Cause | Action | Collect |
|---|---|---|---|
| SOPA010I<br>SOPB012I | For the SELECT PROCESS and SELECT STATISTICS commands, it is likely that the userid issuing the command is defined as a general user by the stage 2 security exit, indicating that the user is only allowed to see the command output for Processes submitted by that same userid. | Review both the short text and long text Connect:Direct messages. Check to see that the userid is defined with the ability to select Process/statistics for Processes not submitted by that userid. | ◆ Authorization for the userid<br>◆ Statistics file |

## Connect:Direct DTF Session-Establishment errors

Connect:Direct DTF session-establishment errors prevent a successful connection between two Connect:Direct systems. This section explains the most common causes of DTF session-establishment errors, actions to take, and the types of data you need to collect to troubleshoot the error.

### Condition: Cannot Establish a Session with Another Connect:Direct Session

The table describes the four possible causes and the courses of action to take.

| Error | Cause | Action | Collect |
|---|---|---|---|
| SVTM026I<br>SVTM045I<br>SVTM053I<br>SVTM104I | The links that connect the two Connect:Direct systems are not active, or an error has occurred on the links. | Review both the short text and long text Connect:Direct messages. If you are unable to determine the problem, check the RPLERRCK DD for possible clues. Use your network management software to determine the status of the links used for system-to-system communication. Reactivate the links. | ◆ None |
| SVTM026I | The cross-domain resource definition for the remote Connect:Direct system is not active. | Review both the short text and long text Connect:Direct messages. If you are unable to determine the problem, check the RPLERRCK DD for possible clues. Use your network management software to determine the status of the Connect:Direct cross-domain manager and cross-domain resource definitions used in communicating with other Connect:Direct locations. Reactivate the cross-domain resource manager or cross-domain resources | ◆ None |

| Error | Cause | Action | Collect |
|---|---|---|---|
| SNAS0801<br>SVTM026I | The VTAM applid for the remote Connect:Direct system is not active. | Review both the short text and long text Connect:Direct messages. If you are unable to determine the problem, check the RPLERRCK DD for possible clues. Use your network management software to determine the status of the applid for the remote Connect:Direct system. Ensure the remote Connect:Direct has initialized. Reactivate the VTAM applid, or initialize the remote Connect:Direct. system. | ◆ None |
| SCCS028I<br>STAA005I | The maximum number of secondary sessions is reached on the secondary Connect:Direct system. | Review both the short text and long text Connect:Direct messages. If you are unable to determine the problem, check the RPLERRCK DD for possible clues. Determine the number of active VTAM sessions for the secondary location. Use your network management software to issue **D NET,ID=applid,E** (VTAM applid) at the secondary location site. Ensure that the maximum number of secondary sessions is sufficient for your requirements. | ◆ None |

## Connect:Direct DTF Out-of-Storage ABENDS

DTF out-of-storage ABENDS occur during heavy Connect:Direct activity or during phases when the DTF has run for a long period of time. This chapter explains the most common causes of DTF out-of-storage ABENDs, actions to take, and the types of data to collect.

### Condition: Out-of-Storage ABEND Occurs in the DTF

The following table describes two possible causes and associated actions to take.

| Error | Cause | Action | Collect |
|---|---|---|---|
| user ABEND U0500<br>user ABEND U0501 | If this condition occurs only during heavy Connect:Direct activity, you may need to modify the initialization parameters or the DTF REGION parameter. | Review both the short text and long text Connect:Direct messages. Limit the number of Processes that can run at one time using the MAXPRIMARY, MAXSECONDARY, and MAXPROCESS initialization parameters. Also, examine the MAXSTGIO initialization parameter to determine if it can be decreased. The REGION size in the DTF JCL may need to be increased to allow more concurrent Processes. | ◆ TSubpool that is growing<br>◆ Dump taken after controlled tests when all DTF activity has ended<br>◆ The Connect:Direct log<br>◆ Connect:Direct initialization parameters<br>◆ Connect:Direct STC (started task) JCL<br>◆ Source for any user exits |
| System ABEND Sx0A<br>System ABEND Sx78 | If this condition appears to be "storage creep" and occurs after the DTF is active for a long time (not necessarily running many Processes immediately), you can take several actions. | ◆ See *Possible Actions* to troubleshoot this problem. | ◆ Dump taken after controlled tests after all DTF activity ended<br>◆ The Connect:Direct log<br>◆ Connect:Direct initialization parameters<br>◆ Connect:Direct STC JCL<br>◆ Source for any user exits |

#### Possible Actions

Use the following list to help you troubleshoot DTF out-of-storage ABENDs:

- ✦ Examine all RUNTASK programs; ensure that for every file opened, a CLOSE and a FREEPOOL is also done.
- ✦ Examine any user exits for GETMAIN macros; verify that FREEMAIN macros are issued for each of those areas.
- ✦ Examine any RUNTASK programs for GETMAIN macros; ensure FREEMAIN macros are issued for each area.
- ✦ Try to pinpoint the type of Processes or other DTF activity that causes the problem:
- ✦ Does this occur only during a COPY?
- ✦ Does this occur when a specific Process is run? What does the Process do?
- ✦ Does this occur when a certain command is issued? What is the command?

**Note:** For the next two suggestions, you can issue an MVS DUMP command for the Connect:Direct for z/OS address space to help in your investigation. Make sure that the SDATA includes the following parameters:

```
RGN,LSQA,SUM,PSA,GRSQ,SQA,SWA,TRT
```

See the *IBM MVS System Commands* manual for the release of z/OS in use.

- If you cannot determine which Process, command, or other activity is causing the storage creep, run a typical batch of Processes or commands that runs when the ABEND occurs. Before the out-of-storage ABEND occurs, go to the ADMIN MD panel and QUIESCE Connect:Direct for z/OS. For example, if the ABEND usually occurs after 10 hours of activity, quiesce after about 8 hours.
- If you did determine that a certain Process or command causes the problem, submit that Process or issue the command several times (the number of times depends on how long it takes before you get to the ABEND). For example, if it occurs after the Process runs 100 times, run it 90 times in your tests. Get a dump of the DTF address space after all DTF activity is finished.

## Allocation and Open Errors

Allocation and open errors involve the source or destination files. This chapter describes errors in which the allocation or opening of a file fails, and the action to take and types of data you need to collect to troubleshoot the error.

### Condition: Allocating a User File Fails

Use the following table to troubleshoot this error.

**Note:** Connect:Direct initialization parameters (ALLOC.CODES and ALLOC.RETRIES) determine which allocation errors, if any, cause a Process that fails on an allocation error to be retried.

| Error | Cause | Action | Collect |
|---|---|---|---|
| SDAA001I<br>SDAA004I<br>SDAA005I<br>SDAA048I | Connect:Direct received an error while allocating a file or data set. | Review both the short text and long text Connect:Direct messages. Check the SYSLOG, console, or Connect:Direct statistics for the text of the SDAA004I and SDAB005I messages. The SDAA004I message contains the allocation parameters used by Connect:Direct. If an error exists, the ERR=nnnn field of the SDAB005I message contains non-zeroes, and error text follows. Use the Connect:Direct message facility to look up the error, which has a format of SDEnnnnI, where nnnn is the number in the ERR field. The operating system dynamic allocation routine returns the ERR value.<br>**Note:** Connect:Direct initialization parameters (ALLOC.CODES and ALLOC.RETRIES) determine which allocation errors, if any, cause a Process that fails on an allocation error to be retried. | ◆ Connect:Direct allocation string (found in the WTO records in the statistics file)<br>◆ Connect:Direct Process involved<br>◆ SDAA004I message output<br>◆ SDAB005I message output |

### Condition: TCQ File Below Defined Threshold Value

Use the following table to troubleshoot this problem.

| Error | Cause | Action | Collect |
|---|---|---|---|
| SPQL003I<br>TCQ file is now below the defined threshold of &VAR. | The number of VSAM file CIs used has reached the defined threshold. | Reduce the number of Processes in the TCQ,or increase the size of the TCQ. Refer to the *Global Initialization Parameters* appendix of the *Connect:Direct for z/OS Administration Guide* for more information. | ◆ None |

## Tranmission Errors

Transmission errors include consistency problems within communication components that can occur during Process execution. The errors can occur within communication components such as VTAM, IBM Network Control Program (NCP), or links.

This section lists possible transmission errors, error messages, probable causes, actions to take, and data to collect to troubleshoot an error.

### Condition: Error During Process Execution Initiation

| Error | Cause | Action | Collect |
|---|---|---|---|
| SVTM041I (SNASYNC1: Session abnormally terminated) | The session or link was lost before Process execution began. | Review both the short text and long text Connect:Direct messages. Use your network management software to determine the status of the link, cross-domain definitions, and applids used in the system-to-system communication. Activate the link, cross-domain definitions, or applids as required, and restart the Process | ◆ None |
| SVTM053I (Session acquire failure) | A protocol error occurred within the Connect:Direct system. | Review both the short text and long text Connect:Direct messages. Also, request a session manager trace and an RPL trace. | ◆ Output from session manager<br>◆ Output from RPL traces |

### Condition: Unrecoverable Error Occurs while a Process Executes

When an unrecoverable send or receive error occurs within the system-to-system session while a Process executes, the three likely causes of the problem are detailed in the following table. You may need a backup copy of the file if a file I/O error caused a send or receive error. If the error is temporary, retrying the Process might clear up the difficulty.

| Error | Cause | Action | Collect |
|---|---|---|---|
| SVTM045I | An I/O error within the primary or secondary node causes Connect:Direct to send a negative response to the other location. | Review both the short text and long text Connect:Direct messages. Accompanying Connect:Direct messages indicate the type of error that caused the send or receive session to fail. Check your network management software for VTAM sense codes, then find the reason the sense code was issued. Correct the problem if possible and retry the Process.<br>In some cases, you need a VTAM buffer or an I/O trace of the error. You may need a backup copy of the file if a file I/O error caused send or receive error. If the error is temporary, retrying the Process might clear up the difficulty. | ◆ Output from the VTAM buffer or I/O trace |
| SVTB020I, followed by a U4095 ABEND | A Connect:Direct system shuts down with either the IMMEDIATE or FORCE parameter specified on the STOP CD command. | Review both the short text and long text Connect:Direct messages. Restart the Connect:Direct system. | ◆ None |
| SVTM042I<br>SVTM043I<br>SVTM044I<br>SVTM045I<br>SVTM046I<br>SVTM047I<br>SVTM048I<br>SVTM049I<br>VTAM sense code 0870<br>VTAM sense code 800A | An error occurs within one of the communication components (VTAM, NCP, or link) | Review both the short text and long text Connect:Direct messages. The communication component containing the error issues error messages. Various VTAM and NCP definitions are incompatible with Connect:Direct operations. Refer to the Selecting RU Sizes section in the appropriate Connect:Direct installation guide for more information. | ◆ VTAM definitions<br>◆ NCP definitions |

## Operator Interface Errors

Operator interface errors occur while you are using the operator interface to issue commands to Connect:Direct. You can find more information on the operator interface in the *Connect:Direct for z/OS Facilities Guide*.

This section lists possible operator interface errors, error messages, possible causes, actions to take, and data to collect to troubleshoot an error.

### Task Busy Message

Use the following table to troubleshoot the problem when you receive a Task Busy message after issuing an Operator Interface command.

| Error | Cause | Action | Collect |
|---|---|---|---|
| IEE342I Modify rejected - task busy | An error in the MCS.SIGNON parameter in the Connect:Direct initialization parameters. | Ensure that the MCS.SIGNON parameter reflects a valid userid-password combination with Connect:Direct operator authority and that the network map is correctly specified on the MCS.SIGNON parameter. Remove comments from this parameter. | ◆ Connect:Direct initialization parameters<br>◆ SYSLOG output |

### User Not Authorized Messages

Use the following table to troubleshoot authorization errors you receive when you issue operator interface commands.

| Error Messages | | | | | |
|---|---|---|---|---|---|
| **SCBB001I** | **SCBC030I** | **SCBD001I** | **SCBE001I** | **SCBF001I** | **SCBF063I** |
| **SCBF064I** | **SCBG001I** | **SCBH001I** | **SCBI001I** | **SCBJ001I** | **SCBK005I** |
| **SCBL001I** | **SCBN001I** | **SCBO001I** | **SCBP001I** | **SCBR002I** | **SCBS001I** |
| **SCBT005I** | **SCBU003I** | **SCBV001I** | **SCBW001I** | **SCBX001I** | **SCBY001I** |
| **SCPA008I** | **SFIA002I** | **SFIA003I** | **SRJA014I** | **SRTA008I** | **SSUB100I** |

| Cause | Action | Collect |
|---|---|---|
| The userid attempting to issue the operator interface commands is not authorized to issue them. | Review both the short text and long text Connect:Direct messages. Check the userid specified in the MCS.SIGNON parameter of the Connect:Direct initialization parameters file. Determine whether that userid has the authority to issue the command. If you believe it does, run a security trace to determine why the user cannot issue the command.<br>See *Security Traces* on page 364 for information on how to run the trace. | ◆ Connect:Direct initialization parameters<br>◆ Security trace |

# Diagnostic Tools

You can perform some problem isolation and diagnostics by running traces. A trace is a sequential recording of program events during execution. Generally, trace output is useful only as a diagnostic tool for the Sterling Commerce Customer Support staff because most of the output is meaningful only with access to the source code.

The following sections describe how to run the different types of traces:

- ✦ Guidelines for running traces
- ✦ Security traces
- ✦ Connect:Direct function traces
- ✦ Connect:Direct automatic traces

The last section, *DD Statements in Startup JCL* on page 373, lists the DD statements in the sample JCL members you can use to run both function and automatic traces.

## Guidelines for Running Traces

When running traces for Sterling Commerce Customer Support personnel, it is important to remember the following:

- ✦ Limit Connect:Direct activity while running traces.
- ✦ Trace the simplest case possible.
- ✦ Disable traces upon completion. They generate considerable overhead.
- ✦ Use traces judiciously. They are for diagnostic purposes only and there is potential security exposure.

## Security Traces

Sterling Commerce Customer Support staff use a security trace to debug security problems. The trace shows:

- Fields from the security control block
- Messages
- Return codes from the security system itself such as RACF, CA-TOP SECRET, and ACF2
- Data set names, if verifying data sets
- Exact userids and passwords

---

**Note:** To prevent a remote node's security from discovering and then using Signon dummy passwords to gain access to a primary node, you can use the REMOTE.DUMMY.PASSWORD initialization parameter. See Appendix A, *Global Initialization Parameters,* in the *Connect:Direct for z/OS Administration Guide.*

---

To start a security trace, complete the following steps:

1. Specify TEST=YES as a parameter in your security exit source and reassemble and link-edit the exit.
2. For Connect:Direct for z/OS in the Connect:Direct startup JCL, allocate a DD for SECURITY, either to SYSOUT or to a data set on DASD. SYSOUT is preferred.
3. To allocate the security trace from the IUI, enter the following command.

```
TSO ALLOC FI(APISECUR) DA(*)        /* FOR ZOS */
```

4. If Connect:Direct is active, stop Connect:Direct. Restart it with the modified JCL startup job, and recreate the problem.

   For details on security, refer to the Connect:Direct installation guide for your operating environment.

## Connect:Direct Function Traces

Connect:Direct contains various internal traces for diagnosing problems and recording events. Based on the trace specified, the Connect:Direct trace output is directed to various ddnames. You can enable these traces using one of the following methods:

- Modify the Connect:Direct startup job stream to include the trace files and ddnames for trace output and add the DEBUG=nnnnnnnn parameter to the initialization parameter data set. The traces are turned on during Connect:Direct initialization and continue running until turned off by the MODIFY command or until Connect:Direct is terminated.
- Issue the Connect:Direct MODIFY command to set DEBUG bits. The trace starts when you issue the MODIFY command. See *Connect:Direct MODIFY Command* on page 368.
- Reduce the amount of trace information by restricting a trace to a Process (MODIFY Debug = parameter) or a specific node (NODETRACE.ON parameter). See *Connect:Direct MODIFY Command* on page 368 for more information about these parameters.

## Debug Settings

Use the following debug settings with the DEBUG=nnnnnnnn initialization parameter, and the BITS.ON=X'nnnnnnnn'and BITS.OFF=X'nnnnnnnn' MODIFY command parameters.

For each debug bit turned on for a trace, you must allocate the equivalent DD names in the Output DD column to the Connect:Direct started task. If you do not specify these DD names in the started task JCL of Connect:Direct, you must allocate them using either the DYN (batch) or DYN (IUI) option described in this section. The one exception is separate trace per task (Rnnnnnnn), which is dynamically allocated by Connect:Direct as required. This trace output is directed to SYSOUT.

The following table shows the available function traces for Connect:Direct for z/OS, with their respective DEBUG settings, and the DD names (or filenames) used for output. Specify these bits using hexidecimal notation, for example, X'80' plus X'10' would result in X'90' while X'08' plus X'04' would result in X'0E'.

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

| DEBUG Setting | Trace Type | Output DD |
|---|---|---|
| 00001000 | Workload Balancing trace | CDPLXWLB |
| 00000100 | In-storage tracing only<br>**Note:** The size of this in-storage table is controlled by the TRACE.BUFFER initialization parameter. | N/A |
| 00000080 | RPL trace - long<br>**Note:** To avoid generating excessive output when you use this trace with a large value for the V2.BUFSIZE initialization parameter, use the short RPL trace. | 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 | SNMP trap trace | SCTRAPDD |

## Displaying DEBUG Settings

Use the INQUIRE DEBUG command to display the current DEBUG settings.

### Command Format

The INQUIRE DEBUG command has the following format and parameter.

| Label | Command | Parameter |
|---|---|---|
| (optional) | INQuire DEBUG | WHERE (SERVER=server name) |

The parameter for the INQUIRE DEBUG command is:

| Parameter | Description |
|---|---|
| WHERE (SERVER=server name) | This parameter is optional. This parameter specifies the Connect:Direct/Server whose DEBUG settings you want to view. The server name parameter is the 1–8 character name assigned to a Connect:Direct/Server by the CDPLEX.SERVER initialization parameter. If you omit this parameter in a Connect:Direct/Plex environment, the DEBUG settings for the Connect:Direct/Manager are displayed. You cannot use this parameter in a Connect:Direct/Stand-alone Server. See *Debug Settings* on page 365 for a descriptions of these settings. |

### Using the INQUIRE DEBUG Command from the Batch Interface

To use the INQUIRE DEBUG command from the batch interface:

1. Place your command in a batch job stream.
2. Submit the job while Connect:Direct is running.

**Note:** 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.

3. Verify the results.

The following figure shows a partial sample report:

```
================================================================================
node.name        *INQ DEBUG*     DATE: yyyy.mm.dd TIME: hh:mm:ss
================================================================================

DEBUG            => '00000001'
SESSIONS QUIESCED=> USER01
NODE.TRACE.ON    => (SC.DUB.TEST1   00000010)
TCQ DSN          => USER01.CD.TCQ
TCX DSN          => USER01.CD.TCX
TCQ THRESHOLD    =>        5%
TCQ File            0% Full. Max.# CI       104  # Used CI     0
```

### Issuing the INQUIRE DEBUG Command through the IUI

To display the DEBUG settings from the IUI:

1. Select option INQ from the Connect:Direct for z/OS Administrative Options Menu.

   The Inquire DTF Internal Status screen is displayed.

2. If you want to view the DEBUG settings for a specific Connect:Direct/Plex member, type the server name in the Server field. If you want to view the DEBUG settings for a Connect:Direct/Manager, leave the Server field blank.

   Leave the Server field blank in a Connect:Direct/Stand-alone Server.

3. Select the IDBG option.
4. Press ENTER.

   The current DEBUG settings are displayed.

### DEBUG Initialization Parameter

Various settings on the DEBUG=*nnnnnnnn* initialization parameter turn on a specific trace option or any combination of options. In the syntax for the DEBUG initialization parameter, *nnnnnnnn* represents the DEBUG setting in hexadecimal.

You can place trace DD statements in the system without slowing down Connect:Direct performance if you do not turn on the trace by specifying the DEBUG parameter.

For problems with SNA connections, use the following four function traces merged into a single output file:

- Session manager trace
- Separate trace per task trace
- Long RPL trace
- COPY routine trace

**Note:** If the problem occurs during file transfer or session establishment of node connections, run the trace on both the sending and receiving nodes.

## Connect:Direct MODIFY Command

The Connect:Direct MODIFY command yields the same types of traces as the DEBUG initialization parameter. Unlike the initialization parameter, however, the MODIFY command does not require you to bring down and restart the DTF. The trace starts when you issue the MODIFY command to turn on the trace bits, provided the DD is allocated.

**Note:** When JES data sets are dynamically allocated, specify FREE=CLOSE to ensure that the DD is deallocated and tracing stopped when you close the trace DD using the MODIFY command.

You can issue the MODIFY command through the batch interface, DMBATCH, or interactively through the IUI. See *Issuing the MODIFY Command Through the Batch Interface* on page 370 or *Issuing the MODIFY Command through the IUI* on page 371. To see the current DEBUG setting, see *Displaying DEBUG Settings* on page 366.

The MODIFY command has the following format and parameters. None of these parameters are required.

| Label | Command | Parameters |
|---|---|---|
| (optional) | MODIFY | BITS.OFF= X'nnnnnnnn' |
| | | BITS.ON= X'nnnnnnnn' |
| | | CLOSE= ddname |
| | | DDNAME= (ddname, nn) |
| | | DEBUG= nnnnnnnn |
| | | DYN (batch)= 'dynamic allocation string'<br>**Note:** Use an equal sign before and quotes around the dynamic allocation string. |
| | | DYN (IUI) dynamic allocation string<br>**Note:** Do not use an equal sign before or quotes around the dynamic allocation string in the IUI screen unless you are entering it on the Command Line Interface.. |

| Label | Command | Parameters |
|---|---|---|
| | | MODDIR.TRACE= YES |
| | | NODE.TRACE.ON= (node name, debug bits) |
| | | NODE.TRACE.OFF= node name |
| | | WHERE(SERVER= server name) |

The following table describes the parameters of the MODIFY command.

| Parameter | Description |
|---|---|
| BITS.OFF = X'nnnnnnnn' | Turns individual trace bits off. Refer to *Debug Settings* on page 365 for the 'nnnnnnnn' value. |
| BITS.ON = X'nnnnnnnn' | Turns individual trace bits on. Refer to *Debug Settings* on page 365 for the 'nnnnnnnn' value. |
| CLOSE = ddname | This parameter specifies the DD name that is closed in the Connect:Direct DTF. |
| DDNAME = (ddname, nn) | This parameter 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 DDNAME provides a consolidated trace of all activity associated with the task. The DDNAME format is R00000nn, where nn is the TASKID. |
| DEBUG = nnnnnnnn | Replaces the system-wide debugging bits with the specified debug bits. Refer to *Debug Settings* on page 365 to see all possible debug bit values. |
| DYN = 'dynamic allocation string' (batch) | This parameter specifies that dynamic allocation is invoked in the DTF using a specified allocation string. Use an equal sign before and quotes around the dynamic allocation string in the batch DYN parameter.<br>You must use an equal sign before and quotes around the dynamic allocation string in the Command Line Interface for the IUI. |
| DYN dynamic allocation string (IUI) | This parameter specifies that dynamic allocation is invoked in the DTF using a specified allocation string. Do not use an equal sign before or quotes around the dynamic allocation string in the MODIFY DYN field on the MODIFY COMMAND screen.<br>**Note:** If you issue the MODIFY command in the Command Line Interface (CLI), you must use an equal sign before and quotes around the dynamic allocation string. |
| MODDIR.TRACE=YES | Requests a module trace. |

| Parameter | Description |
|---|---|
| NODE.TRACE.ON = (node name, debug bits) | Requests a trace run on one or more specified nodes. Up to 15 traces by nodes can run at one time. The node name is the 1–16 character name of the node on which the trace runs. The debug bits are the 8-character DEBUG bits setting. See the BITS.OFF = X'nnnnnnnn' parameter for a listing of debug bits.<br>The trace runs until turned off by the NODE.TRACE.OFF= parameter. |
| NODE.TRACE.OFF = (node name) | Turns off a trace set by the NODE.TRACE.ON parameter.<br>The node name is the 1–16 character name of the node on which the trace is running. |
| WHERE(SERVER=server name) | This parameter specifies which Connect:Direct/Plex member the MODIFY command applies to. The server name parameter is the 1–8 character name assigned to a Connect:Direct/Server by the CDPLEX.SERVER initialization parameter.<br>If this parameter is omitted, the MODIFY command applies to the Connect:Direct/Manager. |

### Issuing the MODIFY Command Through the Batch Interface

To use the MODIFY command from the batch interface:

1. Place the command in the DMBATCH job stream.
2. Submit the job while Connect:Direct is running.
3. Verify the results.

The following example turns on the short RPLOUT trace:

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

The following example turns off the short RPLOUT trace:

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

The following example invokes dynamic allocation in the DTF to the allocated DDNAME RPLERRCK:

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

The following job turns on the merged COPY routine and DMGCBSUB traces.

```
//TRACEON JOB (1111),'TRACES',NOTIFY=JSMITH,CLASS=O,
//            REGION=1024K,MSGCLASS=X
//STEP01   EXEC PGM=DMBATCH,PARM='YYSLYYY'
//STEPLIB   DD DSN=$CD.CD.LINKLIB,DISP=SHR
//DMPUBLIB  DD DISP=SHR,DSN=JSMITH.CNTL
//          DD DISP=SHR,DSN=JSMITH.PROCESS.LIB
//DMMSGFIL  DD DISP=SHR,DSN=JSMITH.CD1.MSG
//DMNETMAP  DD DISP=SHR,DSN=JSMITH.CD1.NETMAP
//DMPRINT   DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//NDMLOG    DD  SYSOUT=*
//SYSIN     DD  *
  SIGNON NETMAP=JSMITH.CD1.NETMAP USERID=(JSMITH)
     MODIFY BITS.ON=X'81000000'
SUBMIT PROC=ACCTSEPT
  SIGNOFF
```

The following job turns off the trace.

```
//NOTRACE JOB (1111),'TRACES',NOTIFY=JSMITH,CLASS=O,
//            REGION=1024K,MSGCLASS=X
//STEP01   EXEC PGM=DMBATCH,PARM='YYSLYYY'
//STEPLIB   DD DSN=$CD.CD.LINKLIB,DISP=SHR
//DMPUBLIB  DD DISP=SHR,DSN=JSMITH.CNTL
//          DD DISP=SHR,DSN=JSMITH.PROCESS.LIB
//DMMSGFIL  DD DISP=SHR,DSN=JSMITH.CD1.MSG
//DMNETMAP  DD DISP=SHR,DSN=JSMITH.CD1.NETMAP
//DMPRINT   DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//NDMLOG    DD  SYSOUT=*
//SYSIN     DD  *
  SIGNON NETMAP=JSMITH.CD1.NETMAP USERID=(JSMITH)
MODIFY BITS.OFF=X'81000000'
  SIGNOFF
```

See *Connect:Direct for z/OS User's Guide* for a description of DMBATCH, and the installation and administration guide for a description of the MODIFY command.

### Issuing the MODIFY Command through the IUI

To use the MODIFY command features through the IUI:

1. Request option **MD** from the Connect:Direct Administrative Options Menu to access the MODIFY COMMAND screen.

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

 Server ==> _________          00200000 (Current DEBUG Settings)
   MODIFY DEBUG           ==> _________     (nnnnnnnn)
   MODIFY BITS.ON         ==> _________     (nnnnnnnn)
   MODIFY BITS.OFF        ==> _________     (nnnnnnnn)
   MODIFY DDNAME          ==> ___________   (ddname,nn)
   MODIFY CLOSE           ==> _________     (ddname)
   MODIFY MODDIR.TRACE    ==> ___           (YES)
   MODIFY DYN ==> ____________________________________________________________
   MODIFY SESSIONS        ==> _ (Quiesce or Resume)   NODE ==> ________________
   MODIFY NODE.TRACE.ON   ==> ( ________________ ________ )
   MODIFY NODE.TRACE.OFF  ==>   ________________
   MODIFY INITPARMS       ==> ___           (YES)
NODE.TRACE.ON----DEBUG     NODE.TRACE.ON----DEBUG                              .
SC.DUB.USER01 8C0000AE     SC.DUB.USER02 QUIESCED
```

2. Type values in the appropriate fields. See the MODIFY command parameter descriptions in *Connect:Direct MODIFY Command* on page 368.

> **Note:** Do not type an equal sign or quotes in the MODIFY DYN field. However, if you issue the MODIFY command on the CLI, you must type an equal sign before and quotation marks around the dynamic allocation string.

3. When you have completed your entries, press **ENTER**.

### Examples of MODIFY Commands

This section shows examples of MODIFY commands for Connect:Direct for z/OS.

The following MODIFY commands set the bits to turn on a short send/receive trace and to dynamically allocate the ddname RPLOUT.

```
MODIFY BITS.ON=X‘00000040’
MODIFY DYN=‘DD=RPLOUT,DSN=A985467.PRINT,DISP=SHR,FREE=CLOSE’
```

After running the trace, the following MODIFY commands close the ddname RPLOUT and turn off the short send/receive trace.

```
MODIFY  CLOSE=RPLOUT
MODIFY  BITS.OFF=X‘00000040’
```

## Connect:Direct Automatic Traces

You can activate some internal Connect:Direct traces without using the DEBUG bits. Activate these traces during normal Connect:Direct operation. Others traces are useful mostly for problem diagnosis. You can activate each of these automatic Connect:Direct traces by having a DD statement in the Connect:Direct startup JCL or allocated in a TSO session, except for NDMCMDS, which is allocated in the JCL for DMBATCH or through the IUI. For more information, see *DD Statements in Startup JCL* on page 373.

**Note:** To avoid generating excessive output, review *Guidelines for Running Traces* on page 363.

Some of the most useful automatic traces are:

- DMVSOPEN contains information related to the allocation of the target data set.
- ESTAE captures information on I/O errors; VTAM connection errors; ABEND control blocks; open and close errors; TCQ/TCX errors on add, update, and so forth; and statistics file write errors. Provided in the basic CONNECT JCL member.
- CDESTAE contains various I/O errors from the statistics facility.
- RPLERRCK captures VTAM send and receive errors. Provided in the basic CONNECT JCL member.
- NDMCMDS shows all actual Connect:Direct commands issued to the DTF, including resolution of symbolics. It can be particularly helpful to debug Connect:Direct commands through the IUI or through DMBATCH if you are having signon problems, syntax errors, and so forth.
- NDMLOG lists all initialization parameters read from the INITPARM data set including obsolete parameters, which are indicated by SITA995I messages, and all modules, along with the last date on which they were modified, and related fix numbers. Provided in the basic CONNECT JCL member.
- CDLOG is a chronological log of Connect:Direct events listing all master, console, programmer, and operator messages, and information on failed tests.

## DD Statements in Startup JCL

Connect:Direct provides two members in the JCL file:

- CONNECT—Contains the minimum set of DDs to run Connect:Direct. You may want to use this set of DD statements as the base and add specific DD statements from the CONNECTX member to fit your tracing needs.
- CONNECTX—Contains all possible DDs to cover stand-alone servers, Connect:Direct/Plex systems, automatic traces, and various DEBUG output. Most of the DD statements in this JCL startup jobstream have been discussed earlier in this chapter along with the types of traces you can run to provide Sterling Commerce Customer Support when a problem occurs.

The DD statements in the CONNECTX JCL can be broken down into the following types:

- Minimal DD statements included in the CONNECT JCL
- DD statements for general operations
- DD statements for running automatic traces.

- DD statements for running detailed function traces in conjunction with the DEBUG initialization parameter in a stand-alone Connect:Direct system
- DD statements for running detailed function traces in conjunction with the DEBUG initialization parameter in a Connect:Direct/Plex system

| | |
|---|---|
| Caution: | The diagnostic DD statements associated with the DEBUG bits cause much more I/O and allocating them can consume significant CPU resources. To avoid degrading performance in your production environment, be sure to follow the guidelines in *Guidelines for Running Traces* on page 363. |

The following DD statements are included in both the CONNECT and CONNECTX JCL:

| DDNAME | Function |
|---|---|
| STEPLIB | Connect:Direct LINKLIB. |
| DMPUBLIB | Connect:Direct Process library |
| USRINFO | Standard display from User exits |
| NDMLOG | Automatic trace to list all initialization parameters read from the INITPARM data set including obsolete parameters, which are indicated by SITA995I messages, and all modules, along with the last date on which they were modified, and related fix numbers. |
| ESTAE | Automatic trace to capture I/O errors, VTAM connection errors, ABEND control blocks, open and close errors, TCQ/TCX errors on adds and updates, and statistics file write errors. |
| RPLERRCK | Automatic trace to capture VTAM/TCP send and receive errors. |

The following DD statements run general functions in all systems:

| DDNAME | Function |
|---|---|
| SYSTERM | Standard MVS |
| NSXOUT | SNA NSXEXIT |
| CDDUMPR | C:D DUMP command output [1] |
| ADRIOXLG | Sysprint for DSS exit |
| SECURITY | Security trace if Stage2 exit has TEST=YES[1] |
| CDSECURI | Stage1 Security Trace |
| DMPRINT | Submit command Output [1] |

1 Can generate excessive output. Use judiciously.

| DDNAME | Function |
|---|---|
| APITRACE | Stage2 API Trace, which consists primarly of IUI and DMBATCH activity[1] |
| AXUNIQ | Error Recovery |

1 Can generate excessive output. Use judiciously.

The following DD statements are used to run automatic traces and perform related functions.

| DDNAME | Function |
|---|---|
| DEVTRACE | Traces UCB open and close activity. |
| DMGEVENT | Traces Event Services activity. |
| DMVSOPEN | Formats the allocation block. |
| CDESTAE | Diagnostics on various I/O errors from the statistics facility. Allocate this name by including in the Connect:Direct startup JCL. |
| IGWTRACE | Traces PDSE program objects load to unload activity. |
| LOSTOUT | Lost term exit trace. |
| NDMAPI | Used for diagnostics on session errors with the API. |
| NDMCMDS | IUI/Application Program Interface (IUI/API) commands passed to Connect:Direct. Also use it for diagnostics on session errors with the API. |
| CDCMDS | IUI/Application Program Interface (IUI/API) commands passed to Connect:Direct. Also use it for diagnostics on session errors with the API. |
| CDLOG | A chronological log of Connect:Direct events listing all master, console, programmer, and operator messages, and information on failed tests.[1] |

1 Can generate excessive output. Use judiciously.

The following output DD statements are used to run detailed function traces and perform related functions in a stand-alone system.

| Output DD | Trace Type | DEBUG Setting |
|---|---|---|
| RADBDD01 | COPY Routine and RUN TASK trace | 80000000 |
| DMCBSUBM | Full TPCB/SYMBOLICS from DMCBSUBM | 10000000 |
| RADBDD05 | Session manager trace | 08000000 |
| Rnnnnnnn | Separate trace per task<br>(Example: “R0000005” to trace TASK 5)<br>**Note:** DDs are dynamically allocated by Connect:Direct as required. | 04000000 |

| Output DD | Trace Type | DEBUG Setting |
|---|---|---|
| RADBDD07 | API session trace | 02000000 |
| RADBDD08 | DMGCBSUB trace | 01000000 |
| DMCOPYRT | TCQSH from DMCOPYRT | 00400000 |
| N/A | Make each SVC dump unique | 00200000 |
| RADBDD16 | GETMAIN/FREEMAIN trace | 00040000 |
| RADBDD21 | I/O buffer trace | 00008000 |
| RADBDD22 | WTO all dynamic allocation parameters | 00004000 |
| RPLOUT | RPL trace - long<br>**Note:** To avoid generating excessive output when you use this trace with a large value for the V2.BUFSIZE initialization parameter, use the short RPL trace. | 00000080 |
| RPLOUT | RPL trace - short | 00000040 |
| RADBDD33 | Version 2 session trace | 00000020 |
| RADBDD35 | Logon exit trace | 00000008 |
| RADBDD36 | Logon Process or trace | 00000004 |
| RADBDD37 | SCIP exit trace | 00000002 |
| SCTRAPDD | SNMP trace | 00000001 |

The following output DD statements are used to run detailed function traces and perform related functions in a Connect:Direct/Plex environment:

| Output DD | Trace Type | DEBUG Setting |
|---|---|---|
| CDPLXACT | ACTION queue manager trace | 00002000 |
| CDPLXCKP | CKPT queue manager trace | |
| CDPLXTCQ | TCQ queue manager trace | |
| CDPLXSTA | STATS queue manager trace | |
| CDPLXREQ | First REQUEST queue manager trace | |
| CDPLXRnn | Second and subsequent REQUEST queue manager trace. For example, “CDPLXR03” traces the third queue manager. The number of queue manager traces is based on the maximum number of servers from the asset protection (APKEY) file. | |
| CDPLXJOI | JOIN queue manager trace | |
| CDPLXWLB | Workload Balancing trace | 00001000 |

# Appendix A

# Global Initialization Parameters

This appendix provides global initialization parameter descriptions and default values (indicated by underlined text). It contains the following sections:

- Global Connect:Direct Initialization Parameters
- Connect:Direct System File Initialization Parameters

The *Connect:Direct for z/OS Quick Reference* also contains a table of the initialization parameters and their default values.

## Global Connect:Direct Initialization Parameters

Initialization parameters supply values for various Connect:Direct functions. Connect:Direct processes these parameters during initialization.

Global initialization parameters apply to a Connect:Direct/Stand-alone Server or each member of a Connect:Direct/Plex environment.

In a Connect:Direct/Plex environment, the local initialization parameters of a member can override some global initialization parameters. See Appendix B, *Local Initialization Parameters* for more information about local initialization parameters.

**Note:** In a Connect:Direct/Plex environment, you can only override initialization parameters allowed in the *local* initialization parameters file using the PARM= keyword in the EXEC statement at system startup. In a Connect:Direct/Stand-alone Server environment, however, you can override *global* initialization parameters with the PARM= keyword in the EXEC statement.

### Handling Return Codes

If you receive a return code of **4** when you stop Connect:Direct, review the NDMLOG for information regarding obsolete and soon-to-be obsolete parameters. Once you remove these parameters, you avoid these warning messages at start-up. These messages are numbered: SITA995I, SITA988I, and SITA989I. For soon-to-be obsolete parameters, migrate to the suggested parameters.

If you combine parameters that cannot co-exist in either the global or local initialization parameter files, initialization terminates with an error message, and a return code of **16**. You can also review the JES log for error messages numbered between SITA674E and SITA689E. The message indicates which parameters cannot co-exist in the initialization parameter file. Decide which parameter to use, modify the file, and restart Connect:Direct.

## ABEND.CODES.NODUMP=(ABEND code list)

The ABEND code list parameter specifies up to 16 system or user ABENDs. This parameter does not apply to RUN TASK ABENDS unless you specify ABEND.CODES.NODUMP for the ABEND.RUNTASK initialization parameter.

When a task ABENDs, Connect:Direct searches this list. If the ABEND is found, the dump is suppressed. Use this parameter to prevent dumps on ABEND codes for which you do not need dump information.

Specify system ABEND codes in the list using an S followed by three characters such as SB37.

Specify user ABEND codes in the list using a U followed by four numeric digits such as U4030.

**Note:** The following ABEND codes are no longer suppressed by default: S9FC, S1ED, SXD9, U0728. If you want to suppress the dump for one of these codes, you must include it in the ABEND.CODES.NODUMP list.

It is not necessary to include the following ABEND codes in the ABEND.CODES.NODUMP list because they are suppressed by default: SX37, SX13, S00C (reason code 4 only), S047, S13E, S222, SA03, U1024, U1025, U1028.

You can indicate a masking character using X. Place the X in any position following either the S or U, such as SXB37 or UX4030.

Modifiable through MODIFY INITPARMS command: YES

## ABEND.RUNTASK=(DUMP | ABEND.CODES.NODUMP)

This parameter specifies whether to refer to ABEND.CODES.NODUMP for ABEND codes when suppressing dumps.

| Value | Description |
|---|---|
| DUMP | System dumps when a RUNTASK ABENDs. This value is the default. |
| ABEND.CODES.NODUMP | An ABEND dump is generated instead of a system dump. Searches the ABEND.CODES.NODUMP list for a match. If the ABEND code is found, the dump is suppressed. If the ABEND code is not found, the system dump continues. |

Modifiable through MODIFY INITPARMS command: YES

## ALLOC.CODES = (allocation errors)

This parameter specifies allocation errors for which Connect:Direct retries 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 generated by the dynamic allocation function (SVC99) of the operating system, such as z/OS. The following is an example of an IBM error reason code (0210) shown in the Connect:Direct SDAB005I message.

```
SDAB005I - ERR=0210,INFO=0000,REQUESTED DATA SET NOT AVAILABLE. ALLOCATED TO ANOTHER
JOB.
```

You can find common dynamic allocation error codes in the Connect:Direct MSG file by enclosing the code in the message SDE*xxxx*I skeleton, where *xxxx* is the error reason code. The variable SDAB005I message short text and description come from the *MVS Programming Authorized Assembler Services Guide* (general publication number SA22-7608-nn), which is accessible on IBM's website. For ERR codes not found in the Connect:Direct MSG file, see the *MVS Programming Authorized Assembler Services Guide.*

Connect:Direct has defined the following special error codes to enable retries for specific failures.

| Value | Description |
|---|---|
| PDSR | This is an internally generated Connect:Direct code, that unlike the others described in this section, is not an allocation code. When two or more Processes attempt to write to the same PDS at the same time, only one can write successfully to the PDS. The remaining Processes have error messages with this PDSR code and are retried in the same way as the other codes discussed in this section. |
| DSNR | This option is similar to the PDSR allocation code but works on nonPDS data sets. When you attempt to send or receive a nonPDS data set, but cannot because the data set 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. |
| GDGR | This option indicates that if the ENQ fails (as described for the GDGENQ parameter) for GDG data sets, the Process is retried. |
| TAPR | This option indicates that a tape Process is retried if the number of tape Processes specified by MAX.TAPE is reached. |
| ARCH | This option indicates that the data set is archived, and Connect:Direct retries based upon the value specified for the ALLOC.RETRIES initialization parameter. You must use the DMGALRCL allocation exit for this option to work.<br>**Note:** It is not necessary to use the Connect:Direct DMGALRCL exit to avoid a common allocation error that ties up the IBM task input/output table (TIOT) via the enqueue on SYSZIOT. See the discussion on the sample allocation exit, DMGALRCL, in *Chapter 10, Using Connect:Direct Exits*, for more detailed information on recall processing of migrated and archived data sets. |

Allocation retries are controlled by the PNODE and supported for z/OS, VM/ESA, and VSE/ESA 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 |

Modifiable through MODIFY INITPARMS command: YES

## ALLOC.MSG.LEVEL = INFO | WARN | SEVERE

This parameter specifies the severity level of dynamic allocation messages to be displayed. By setting the severity level, you can suppress unwanted messages. ALLOC.MSG.LEVEL values correspond to levels defined by Dynamic Allocation and are described in the IBM Authorized Assembler Services Guide (GC28-1467).

| Value | Description |
|---|---|
| INFO | This value indicates that all informational messages are displayed. |
| WARN | This value indicates that only warning messages are displayed. |
| SEVERE | This value indicates that only severe messages are displayed. |

Modifiable through MODIFY INITPARMS command: NO

## ALLOC.RETRIES = number of retries

This parameter specifies the number of Connect:Direct retries of an allocation failure. The default is 20.

Modifiable through MODIFY INITPARMS command: YES

## ALLOC.WAIT = hh:mm:ss

This parameter specifies the amount of time that Connect:Direct waits between retries of an allocation failure. The default is 00:03:00.

Modifiable through MODIFY INITPARMS command: YES

## ALLOCATION.EXIT = modname

This parameter specifies the name of the user-written interface that communicates with Connect:Direct for z/OS. 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.

DMGA3390 and DMGALOEX are the Connect:Direct sample exits. They are located in the $CD.SAMPLIB. distribution library.

The default is no allocation exit.

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX = NO | YES

This parameter indicates whether or not Connect:Direct initializes as a Connect:Direct/Plex operation.

| Value | Description |
|---|---|
| NO | Connect:Direct does not initialize as a Connect:Direct/Plex operation. |
| YES | Connect:Direct initializes as a Connect:Direct/Plex operation. |

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.TIMER = 5 | nn

This parameter specifies the time-out value for XCF communications in minutes. The valid range is 0, 5–99. Zero (0) indicates that no time-out is set for XCF communications.

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.WLM.GOAL = (NO | YES, exitname)

This parameter specifies whether the IBM Workload Manager (WLM) Goal Mode queries are used for balancing Connect:Direct/Plex Process workload in a sysplex.

| Value | Description |
|---|---|
| NO | No Goal Mode queries are made. Process workload is balanced without determining system capacity information. This is the default value. |
| YES | Goal mode queries are made. A warning message is issued when a WLM query is made that shows a server is running on a system that is NOT in Goal Mode. |

| Value | Description |
|---|---|
| exitname | If you want to implement your own method for using WLM query information, you can create a WLM user exit and specify the name of the exit. For more information, refer to *WLM Exit* on page 246. You can find a sample user exit in SAMPLIB called CDWLMEX. |

Modifiable through MODIFY INITPARMS command: yes

## CKPT = nK | nM

This parameter enables 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.) **K** means thousands of bytes; **M** means millions of bytes.

Valid values of **n** are:

- 1–2147483K
- 1–2147M

The default is no automatic checkpointing.

Connect:Direct uses the value specified, rounded to the nearest block boundary, to determine when a checkpoint is taken. The CKPT specification on the Connect:Direct Copy statement always overrides the CKPT initialization parameter value.

Modifiable through MODIFY INITPARMS command: YES

**Note:** For sequential files, do not specify a CKPT value less than:

BLKSIZE * NCP * 10 * # stripes

where NCP is the number of buffers for reading data from or writing data to a sequential data set using BSAM and # stripes refers to striped extended-format data sets. For more information, see *Problems Involving Checkpoints* on page 336. Also, see additional information on the CKPTand NCP parameters in the COPY statement on the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.

## CKPT.DAYS = number of days

This parameter 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. The default is 4.

Modifiable through MODIFY INITPARMS command: YES

## CKPT.MODE = (RECORD | BLOCK  BLOCK | RECORD  PDS | NOPDS NOPDS | PDS  VSAM | NOVSAM  VSAM | NOVSAM)

This parameter enables you to control checkpointing in both record and block level transfers.

**Note:** This parameter does not apply to TCP/IP or LU6.2 type connections. See the *Process Queuing and Recovery* chapter in Connect:Direct for z/OS *User's Guide* for an explanation of when checkpointing occurs.

| Subparameter | Description |
|---|---|
| RECORD\|BLOCK | Refers to transferring physical sequential (PS) files. It determines whether checkpointing is allowed when the CKPT parameter is specified on the COPY statement. If a record level transfer is taking place, BLOCK does not allow record level checkpointing, even if you have coded the request on the COPY statement. If you are performing a block level transfer, BLOCK enables checkpointing requests coded on the COPY statement. RECORD enables record level checkpointing. |
| BLOCK\|RECORD | Refers to transferring 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 you use the CKPT parameter, you do not have to request checkpointing on each COPY statement. 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. Use BLOCK to prevent automatic checkpointing on a record level transfer. |
| PDS\|NOPDS | Refers to transferring partition data sets (PDS). Because checkpointing information is sent with each member of a PDS, this subparameter specifies whether checkpointing is allowed with PDS transmission if a request is coded on the COPY statement. NOPDS prevents checkpointing on the PDS, even if you requested it on the Copy statement. PDS enables PDS transmissions to be checkpointed. |
| NOPDS\|PDS | Refers to transferring PDS. 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 do not have to request checkpointing on each COPY statement. If you specify PDS, all PDS transmissions are automatically checkpointed. NOPDS prevents automatic checkpointing of PDS transmission. |
| VSAM\|NOVSAM | Specifies whether checkpointing takes place for VSAM files when the checkpoint parameter is specified in the COPY statement. |
| VSAM\|NOVSAM | Specifies whether automatic checkpointing takes place for VSAM files. |

Modifiable through MODIFY INITPARMS command: YES

## CONFIRM.COLD.START = YES | NO

This parameter indicates whether an operator has to confirm a COLD start of the Transmission Control Queue (TCQ) and Statistics files.

| Value | Description |
|---|---|
| YES | Issues a Write to Operator with Reply (WTOR) prompt to force the operator to confirm the request for a COLD start before executing the COLD start. |
| NO | Performs a COLD start without requiring operator confirmation. |

Modifiable through MODIFY INITPARMS command: NO

## CRC = (OFF | ON, YES, No)

This parameter specifies whether cyclic redundancy checking (CRC) is performed for TCP/IP connections. The CRC parameter is a 32 bit checksum used by Connect:Direct to detect network data corruption which occasionally occurs in IP data networks. If network data corruption is detected, Connect:Direct will stop the Process execution and restart the Process from the last checkpoint record. This provides an extra layer of data integrity when transmitting data over an IP network. You cannot enable CRC checking when running Secure+ Option.

| Parameter | Description |
|---|---|
| OFF \| ON | This parameter specifies whether CRC is ON or OFF for the node. This first positional parameter establishes the default for the node. |
| YES \| No | This parameter specifies whether overrides are allowed. You can perform overrides in a Process statement or with a network map parameter. |

Modifiable through MODIFY INITPARMS command: YES

## CTCA = NO | YES

This parameter specifies whether the channel-to-channel adapter (CTCA) driver is loaded at Connect:Direct initialization.

| Value | Description |
|---|---|
| NO | The CTCA driver is not loaded at Connect:Direct initialization. This value is the default. |
| YES | The CTCA driver is loaded at Connect:Direct initialization. |

Modifiable through MODIFY INITPARMS command: NO

## CTCA.TIMER = number of seconds

This parameter specifies the number of seconds for a SEND/RECEIVE to wait before a timeout on the connection occurs. The valid range is from 60–300. The default is 180 seconds.

Modifiable through MODIFY INITPARMS command: NO

## DATEFORM = (MDY | DMY | YMD | YDM)

This parameter specifies how dates are displayed and input. Dates are displayed with a 4-digit year format unless a 2-digit year format is specified. You can use periods or back slashes (/) to separate the month, day, and year values.

This parameter applies only to Gregorian dates.

| Value | Description |
|---|---|
| MDY | This value indicates that dates are displayed or input in one of the following formats:<br>◆ MMDDYYYY<br>◆ MM/DD/YYYY<br>◆ MM.DD.YYYY<br>◆ MMDDYY<br>◆ MM/DD/YY<br>◆ MM.DD.YY |
| DMY | This value indicates that dates are displayed or input in one of the following formats.<br>◆ DDMMYYYY<br>◆ DD/MM/YYYY<br>◆ DD.MM.YYYY<br>◆ DDMMYY<br>◆ DD/MM/YY<br>◆ DD.MM.YY |
| YMD | This value indicates that dates are displayed or input in one of the following formats.<br>◆ YYYYMMDD<br>◆ YYYY/MM/DD<br>◆ YYYY.MM.DD<br>◆ YYMMDD<br>◆ YY/MM/DD<br>◆ YY.MM.DD |

| Value | Description |
|---|---|
| YDM | This value indicates that dates are displayed or input in one of the following formats.<br>◆ YYYYDDMM<br>◆ YYYY/DD/MM<br>◆ YYYY.DD.MM<br>◆ YYDDMM<br>◆ YY/DD/MM<br>◆ YY.DD.MM |

Modifiable through MODIFY INITPARMS command: NO

## DEBUG = nnnnnnnn

Turns on a specific trace option or any combination of options, where nnnnnnnn represents a debug setting in hexadecimal. By default, this initialization parameter is set to 00000000 meaning that the trace is not turned on. You can modify DEBUG= settings using the MODIFY command. See *Connect:Direct MODIFY Command* on page 368.

See *Debug Settings* on page 365 for a complete listing of the DEBUG settings, the trace types produced, and the ddnames used for output.

Modifiable through MODIFY INITPARMS command: NO

## DESC.CRIT = (descriptor code)

This parameter specifies the descriptor code used for critical write-to-operator (WTO) messages. Messages that go to the critical route code are, for example, disastrous session errors or critical ABENDs. You can specify as many as 16 codes. The default of DESC.CRIT = (2) specifies immediate action is required.

Modifiable through MODIFY INITPARMS command: YES

## DESC.NORM = (n,n,...)

This parameter specifies the descriptor code used for normal (WTO) messages. You can specify as many as 16 codes.

DESC.NORM = ( ) specifies that no descriptor code is assigned. The default is no descriptor code.

Modifiable through MODIFY INITPARMS command: YES

## DESC.TAPE = (n,n,...)

This parameter specifies the descriptor code for the tape pre-mount message used by Connect:Direct for z/OS. (For more information, see the *Connect:Direct for z/OS Facilities Guide*.) You can specify as many as 16 codes. The default is 2.

Modifiable through MODIFY INITPARMS command: YES

## DSNTYPE = YES | NO

This parameter indicates whether the DSNTYPE will be propagated from the source file (to be used as the default destination file DSNTYPE) or whether it must be coded within the Process. DSNTYPE=YES must be specified in the receiving node's initialization parameters for the receiver to perform the propagation of DSNTYPE from the source file.

| Value | Description |
|---|---|
| Yes | Indicates that the DSNTYPE of the source file (FROM DSN) will be used to create the new output file (TO DSN) if it cannot be obtained from any other COPY statement parameter. The DSNTYPE will be propagated only when the output data is allocated as DISP=NEW or does not exist. A COPY statement parameter that influences DSNTYPE is required when copying to different DSN types, for example, PDSE to PDS or PDS to PDSE. |
| No | Indicates that the DSNTYPE of the source file (FROM DSN) will not be propagated to the new output file (TO DSN). The DSNTYPE must be supplied by a COPY statement parameter, or else it will take the system default (PDS for partitioned organization or BASIC for sequential organization). |

Modifiable through MODIFY INITPARMS command: NO

## ECZ.COMPRESSION.LEVEL = 1 | n

This parameter determines the level of compression. The valid value range is 1–9. The default is 1, which usually provides sufficient compression. The data goes through the compression code the number of times indicated by the value specified for the parameter.

***Caution:*** Compression consumes significant CPU resources. To avoid degrading performance in your production environment by changing the global, default settings for the extended compression parameters, you should: (1) Review *Changing the Values of Extended Compression Initialization Parameters* on page 337 to view test results that describe how changing the global, default values affects performance, and (2) Review *Testing the Effects of Changing Values for Extended Compression Parameters* in *Connect:Direct for z/OS User's Guide* for information on using the CDSACOMP offline utility to perform tests to determine whether changing the default values of the extended compression parameters at the global level or by overriding them at the Process level will significantly improve your system performance.

Modifiable through MODIFY INITPARMS command: YES

## ECZ.MEMORY.LEVEL = 4 | n

This parameter identifies how much virtual memory is allocated to maintain the internal compression state. This memory is above the 16 megabyte line. The valid value range is 1–9. The default is 4. Level 1 requires the least memory (1 KB); level 9 requires the most memory (256 KB).

Compression consumes significant CPU resources. For more information, see the caution for the ECZ.COMPRESSION.LEVEL parameter on page 389.

Modifiable through MODIFY INITPARMS command: YES

## ECZ.WINDOWSIZE = 13 | nn

This parameter determines the size of the compression window or history buffer. This memory is above the 16 megabyte line. The valid values are 8–15. The default is 13. Size 8 uses 1 KB of memory, whereas Size 15 requires 128 KB of memory.

Compression consumes significant CPU resources. For more information, see the caution for the ECZ.COMPRESSION.LEVEL parameter on page 389.

Modifiable through MODIFY INITPARMS command: YES

## ESF.WAIT = hh:mm:ss

This parameter specifies the maximum amount of time that Connect:Direct waits before checking for ESF-submitted Processes. When that time expires, Connect:Direct retrieves any Processes submitted through the ESF.

The default is 00:03:00.

Modifiable through MODIFY INITPARMS command: YES

## ESTAE = YES | NO

This parameter specifies whether error recovery procedures are invoked for Connect:Direct for z/OS.

| Value | Description |
|---|---|
| YES | Establishes error recovery procedures for main tasks and subtasks. This is the default value. |
| NO | No error recovery procedures are established. |

***Caution:*** 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 goes into a wait state.

Modifiable through MODIFY INITPARMS command: NO

## EXPDT = (TT,DD,TD,DT) (if multiple values)
## EXPDT = TT | DD | TD | DT | ALL | NONE (if only one value)

This parameter 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. |
| 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 value is the default. |

If you specify multiple values, enclose them in parentheses and separate them by a comma. If you code a single value, you do not need to enclose them in parentheses. If you code ALL or NONE, you cannot code any other keyword.

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 uses 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 uses 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 obtains 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 uses the tape label. When Connect:Direct dynamically allocates the data set on the receiving side, 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 can reflect a different EXPDT than the tape label or DASD DSCB, depending upon the defaults on the receiving side.

Modifiable through MODIFY INITPARMS command: YES

## EXTENDED.RECOVERY = NO | YES

This parameter specifies whether Connect:Direct Extended Recovery is used.

| Value | Description |
|---|---|
| NO | Connect:Direct Extended Recovery is not used. This is the default value. |
| YES | Connect:Direct Extended Recovery is used. |

Extended recovery is supported in the Connect:Direct/Stand-alone Server and Connect:Direct/Plex environments.

Modifiable through MODIFY INITPARMS command: NO

## GDGALLOC = GENERATION | DSNAME

This parameter specifies whether Connect:Direct allocates GDG data set by generation or by data set.

| Value | Description |
|---|---|
| GENERATION | Connect:Direct allocates by generation, so the allocation is DATA.SET.NAME(xx), where (xx) is the relative generation such as (+1), (-3), (0), and so on. Use for SMS GDG files so that duplicate GDG generations are not created. GDGALLOC=GENERATION logic supports sites that use both SMS and non-SMS GDG data sets. The GDGENQ parameter is ignored and Connect:Direct handles the ENQ logic. Connect:Direct performs an exclusive SYSTEMS ENQ on QNAME NDMGDG and uses the base GDG name as the RNAME.<br>This exclusive ENQ causes a serialization within any Connect:Direct on the system. First, Connect:Direct issues a LOCATE to find the current absolute generation number. Next, Connect:Direct performs an exclusive SYSTEM ENQ on QNAME SYSDSN and uses the data set name returned from LOCATE as the RNAME. If successful, Connect:Direct then performs a shared SYSTEM ENQ on QNAME SYSDSN and uses the base GDG name as the RNAME. Dynamic allocation (SVC 99) is performed.<br>If the data set name allocated is not the same as what the LOCATE returned, the process ends with an SDEGDGRI error and the file is deleted. The two SYSDSN ENQs are not released until the end of the COPY step. The NDMGDG ENQ on the base is released (DEQueued) after the allocation is performed. |

| Value | Description |
|---|---|
| DSNAME | Connect:Direct allocates by data set name (i.e., DATA.SET.NAME.G0000V00). If GDGENQ=YES parameter is coded, Connect:Direct handles the ENQ logic. Connect:Direct performs an exclusive SYSTEMS ENQ on QNAME NDMGDG and uses the base GDG name as the RNAME.<br>This exclusive ENQ causes a serialization within any Connect:Direct on the system. First, Connect:Direct issues a LOCATE to find the current absolute generation number. Next, Connect:Direct performs an exclusive SYSTEM ENQ on QNAME SYSDSN and uses the data set name returned from LOCATE as the RNAME. If successful, Connect:Direct then performs dynamic allocation (SVC 99) on the data set name that LOCATE returned.<br>If the ENQ is not successful, the process ends with an SDEGDGRI error and allocation does not occur. The SYSDSN ENQ is not released until the end of the COPY step. The NDMGDG ENQ on the base is released (DEQueued) after the allocation is performed.<br>This value is the default. |

If you code GDGALLOC = GENERATION, then for new non-SMS managed files, you must use one of the IBM-approved methods of supplying DCB attributes. For example, you could use one of these methods:

- Code a model DSCB in the Process DCB=(model DSCB data set name).
- Use an existing data set with the attributes desired for the new GDS data set DCB=(cataloged data set name) in the Process.
- Have a model DSCB defined for the generation data group (GDG).
- Use the LIKE=(cataloged data set name) parameter in the Process.

If you fail to use an approved method when creating new GDG data sets by generation, you will receive an allocation error of 048C.

Modifiable through MODIFY INITPARMS command: YES

## GDGENQ = YES|NO

This parameter specifies whether or not Connect:Direct uses ENQ on the data set and on the base GDG before allocation to see if another address space or task has this data set or base GDG allocated. This condition applies to output GDG data sets only.

| Value | Description |
|---|---|
| YES | Connect:Direct performs an ENQ on the entire data set (with the G0000V00 appended to the base GDG name) if the Process is copying to an output GDG data set. The ENQ fails if any job in the system has this data set allocated (including Connect:Direct). Another ENQ is done for the GDG base. This ENQ fails if a different job (excluding Connect:Direct) has any generation of this GDG allocated. If either ENQ fails, the Process is retried according to the ALLOC.CODES, ALLOC.RETRIES, and ALLOC.WAIT initialization parameters. If GDGR is not specified in the ALLOC.CODES, then the Process is not queued for retry.<br>The ENQs are not done for data set names coded in the Process as data.set.GnnnnVnn, only for data sets that specify a relative generation, such as (+1) (0) or (-1). |

| Value | Description |
|---|---|
| NO | The GDGENQ parameter is disabled and no ENQ is done for GDG copies. This value is the default. |

Modifiable through MODIFY INITPARMS command: YES

## IMMEDIATE.SHUTDOWN = I | R | (I, nnn | 60) | (R, nnn | 60)

This parameter determines how an immediate shutdown issued through the STOP CD command is executed.

| Value | Description |
|---|---|
| I | The immediate shutdown waits for all Run Task programs to complete before shutting down Connect:Direct. This value is the default<br>nnn \| 60<br>This value specifies the number of seconds that Connect:Direct waits after starting the SHUTDOWN before it forces the DTF to terminate. If the wait time expires and Connect:Direct forces a shutdown, the following message is displayed to the operator:<br>SITB999I SHUTDOWN IMMEDIATE timed out. The default is 60.<br>nnn=0 indicates that Connect:Direct does not have a time limit within which it must shut down the DTF. |
| R | The immediate shutdown commands functions as a runtaskimm shutdown. It terminates all executing Run Task Processes and then shuts down Connect:Direct.<br>nnn \| 60<br>This value specifies the number of seconds that Connect:Direct waits after starting the SHUTDOWN before it forces the DTF to terminate. If the wait time expires and Connect:Direct forces a shutdown, the following message is displayed to the operator:<br>SITB999I SHUTDOWN IMMEDIATE timed out. The default is 60.<br>nnn=0 indicates that Connect:Direct does not have a time limit within which it must shut down the DTF. |

Modifiable through MODIFY INITPARMS command: NO

## INVOKE.ALLOC.EXIT = SEND|RECV|BOTH

This parameter determines whether to invoke the allocation exit upon sending a file, receiving a file, or both sending and receiving a file.

| Value | Description |
|---|---|
| SEND | Invokes the allocation exit upon sending a file. |

| Value | Description |
|---|---|
| RECV | Invokes the allocation exit upon receiving a file. This is the default value. |
| BOTH | Invokes the allocation exit upon both sending and receiving a file. |

Modifiable through MODIFY INITPARMS command: YES

## INVOKE.ALLOC.EXIT.ON.RESTART = NO|YES

This parameter indicates whether to invoke the allocation exit on restart of a previously failed Process.

| Value | Description |
|---|---|
| NO | This value indicates whether to invoke the allocation exit on restart of a previously failed Process. This is the default value. |
| YES | This value indicates whether to invoke the allocation exit on restart of a previously failed Process. |

Modifiable through MODIFY INITPARMS command: YES

## INVOKE.SPOE.ON.SNODEID = NO|YES

This parameter indicates whether to invoke Secure Point-of-Entry when a user codes SNODEID = parameter on the PROCESS.

| Value | Description |
|---|---|
| NO | Connect:Direct does not invoke Secure Point-of-Entry when a user codes SNODEID = parameter on the PROCESS. |
| YES | Connect:Direct invokes Secure Point-of-Entry when a user codes SNODEID = parameter on the PROCESS. |

Modifiable through MODIFY INITPARMS command: NO

## MAX.AGE = (nnn , * | *(nnn) | ALL | ALL(nnn) | status_type | status_type (nnn) , list)

This parameter specifies the number of calendar days to wait before purging a Process. You can also use this parameter to purge only Processes with a specific status. You can also define a different number of days to wait for each status type.

**Note:** Connect:Direct software does not automatically delete Processes when you specify **0**.

The following table explains the values.

| Value | Description |
|---|---|
| nnn | The number of days to wait before purging a Process. The maximum value is 32767. After the specified number of days as passed, the Process is deleted. If nnn=0, no MAX.AGE processing is performed. |
| * (asterisk) | Purge all Processes. |
| ALL | Purge all Processes with Hold queue error status types (HE, HO, HP, HS, RA, and RH). |
| status_type | Purge only the specified status types (HE, HO, HP, HS, RA, RH, HC, HI, HR, WC, WT, or WX). |
| list | Specify a list by separating entries with a comma. |

The following table describes the queue status values that you can select for automatic removal from the TCQ.

| Parameter | Description |
|---|---|
| ALL | This value indicates a request for all of the queue types for the MAX.AGE parameter. If you specify ALL and another queue type, ALL is ignored. |
| HE | This value indicates to automatically remove held for error (HE) status values. This parameter is the default. |
| HO | This value indicates to automatically remove held for operator (HO) status values. |
| HP | This value indicates to automatically remove held due to Process error (HP) status values. |
| HS | This value indicates to automatically remove held for suspension (HS) status values. |
| RA | This value indicates to automatically remove held for restart due to allocation error (RA) status values. |
| RH | This value indicates to automatically remove restart held (RH) status values. |

The order of precedence for MAX.AGE subparameters is:

1. Any specified status types take precedence over the ALL subparameter and the wildcard (*) subparameter.
2. The ALL subparameter (purge types HE, HO, HP, HS, RA, and RH) takes precedence over the wildcard (*) subparameter.
3. The wildcard (*) subparameter (purge all valid status types) takes the last precedence.

An example of the MAX.AGE parameter follows.

```
MAX.AGE=(10,*,ALL(4),HO(20),HI(0))
```

The following table explains the values in the example.

| Value | Description |
|---|---|
| 10 | The default number of days before a Process is removed from the Process queue. |
| * | All Process types are removed. Because no waiting period is specified for *, the default of 10 days is used. |
| ALL(4) | Processes with Hold queue error status types (HE, HO, HP, HS, RA, and RH) are removed after 4 days in the Hold queue. Note that the 4-day waiting period for Processes with these status types overrides the 10-day default waiting period. |
| HO(20) | Processes with an HO status type are purged after 20 days. As a result, they are not purged when ALL or wildcard (*) Processes are purged. |
| HI(0) | Processes with an HI status type are not eligible for purge because zero is specified as the waiting period. They are never automatically purged from the Hold queue. |

Modifiable through MODIFY INITPARMS command: YES

## MAX.AGE.TOD = time

This parameter specifies when to automatically purge a Process queue. If omitted, the queue is purged at midnight and at Connect:Direct initialization. You can use any valid Connect:Direct time format for the TIME parameter.

An example of the MAX.AGE.TOD parameter is.

```
MAX.AGE.TOD=14:30
```

In this example, the Process queue purge executes at 2:30 p.m.

Modifiable through MODIFY INITPARMS command: YES

## MAXBATCH = number of users

This parameter specifies the maximum number of batch users that can sign on to Connect:Direct at any one time. No other users are allowed to sign on when this limit is reached. The range is from 0–512. If you use 0, this parameter is not used and the signon limit is set by MAXUSERS. The default is the MAXUSERS value.

Modifiable through MODIFY INITPARMS command: NO

## MAXPRIMARY = number of PNODE sessions

A PNODE session is started when initiating a Process to one or more SNODEs. This parameter specifies the maximum number of PNODE sessions that can be active concurrently. The range is from 2 to 512. The default is 6.

**Note:** Although 512 is the maximum valid value for this parameter's range, the combined values of MAXPRIMARY and MAXSECONDARY (or MAXPROCESS alone) should equal 150 or less.

Modifiable through MODIFY INITPARMS command: NO

## MAXPROCESS = number of executing PNODE and SNODE Processes

This parameter specifies the maximum number of executing PNODE and SNODE Processes allowed at one time. The value allowed is between 2 and 1024, inclusive. The default is the value of MAXPRIMARY + MAXSECONDARY.

Modifiable through MODIFY INITPARMS command: NO

**Note:** Although 1024 is the maximum valid value for this parameter, MAXPROCESS should be set to 150 or less.

## MAXRETRIES = number of retries

This parameter specifies the maximum number of retries that is made to start a node-to-node session. If Connect:Direct cannot start the session, 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. The default is 7. For related information, see the WTRETRIES initialization parameter.

Modifiable through MODIFY INITPARMS command: YES

## MAXSECONDARY = number of SNODE sessions

A SNODE session is started when receiving a Process from a PNODE. This parameter specifies the maximum number of SNODE sessions that can be active concurrently. The range is from 2 to 512. The default is 6.

**Note:** Although 512 is the maximum valid value for this parameter's range, the combined values of MAXPRIMARY and MAXSECONDARY (or MAXPROCESS alone) should equal 150 or less.

Modifiable through MODIFY INITPARMS command: NO

## MAXSTGIO = maximum storage used for sequential data set transfers for system-determined NCP, maximum I/O storage for user-specified NCP)

This parameter specifies the maximum amount of storage used for BSAM sequential data set transfers. MAXSTGIO has two positional parameters that limit the total I/O buffer size in different circumstances. The first is used to limit it when the system determines the number of channel programs (NCP). The second is used to limit it when you specify the NCP in the COPY statement. (For a full description of all COPY statement parameters including NCP, refer to the Process Guide Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html. The larger the value, the better the I/O performance for sequential file transfers. However, the larger the value, the larger the REGION size that may be required for the DTF.) The valid value range for both parameters is 60000–8,388,608 (60K–8M). The default value is 1048576 (1M).

- Connect:Direct uses these parameters to limit the number of buffers/channel programs used for sequential I/O and calculates the NCP by dividing the MAXSTGIO value by the block size of the data set being transferred. The number of channel programs specified can range from 0 (to have the system determine the value) to 255. For more information on how Connect:Direct processes sequential data sets using BSAM, see *Improving BSAM Data Transfer Rates* on page 335.

For example, if you specify the default of 1 MB for MAXSTGIO, the following number of channel programs/buffers are allocated for data sets with the block sizes listed in the following table. Also listed is the amount of storage required for buffers for this transfer, which is a product of the block size and NCP.

| BLKSIZE | Number of Channel Programs/Buffers | Storage Used for Transfer |
|---|---|---|
| 80 | 255 | 20,400 |
| 4,080 | 192 | 783,360 |
| 6,400 | 128 | 819,000 |
| 27,998 | 32 | 895,936 |

- The data sets in the above table did not take advantage of striping or Large Block Interface (LBI) support, which affect BSAM sequential data set transfer rates. In addition, the method for determining NCP did not vary—the NCP values listed above were all system-determined. You can also specify the NCP by using the second positional parameter of the MAXSTGIO initialization parameter.
- If you specify a large value for MAXSTGIO, be sure to review 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.

Modifiable through MODIFY INITPARMS command: YES

## MAX.TAPE = number of tape Processes | NONE

This parameter specifies the maximum number of tape Processes that are allowed to start in a node.

| Value | Description |
|---|---|
| number of tape Processes | The numeric range is 0–32767. The default is 10.<br>When this limit is reached, one of two events can occur to any Processes that try to allocate a tape unit:<br>◆ The Processes end with a return code of 8 and a SDETAPRI message<br>◆ If the ALLOC.CODES initialization parameter includes the code TAPR, the Processes are placed on the timer retry queue. They are then retried the number of times specified in the ALLOC.RETRIES parameter, at the interval specified in the ALLOC.WAIT parameter. |
| NONE | This value indicates that the node does not perform tape Processing. |

If you specify NONE, any Process that tries to copy from or to a tape on this node ends with a return code of 8 and a SDETAPRI message. You can still copy from or to tapes on the SNODE.

Modifiable through MODIFY INITPARMS command: NO

## MAXUSERS = number of users

This parameter specifies the maximum number of interactive users and 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. The default is 6.

Modifiable through MODIFY INITPARMS command: NO

## MCS.CLIST = console operator CLIST library file name

This parameter specifies the file name of the CLIST library of the z/OS console operator. This parameter is required for use of the console operator interface. No default exists.

Modifiable through MODIFY INITPARMS command: NO

## MCS.SIGNON = (SIGNON USERID = (user ID,password) NETMAP = network map [optional parameters])

This parameter specifies the console Signon command of the operator for the Operator interface.

| Keyword | Subparameter | Description |
|---|---|---|
| USERID | user ID | The user ID of the console operator.<br>Password associated with the user ID of the console operator. |
| NETMAP | network map | The VSAM file that identifies all valid Connect:Direct nodes and applids in the network. |

| Keyword | Subparameter | Description |
|---|---|---|
| | optional parameters | Optional parameters associated with the SIGNON command. These parameters are described in the *Managing Sessions* chapter in the *Connect:Direct for z/OS User Guide*. |

You must specify the SIGNON USERID and NETMAP keywords.

This parameter is required for installations that use the console operator interface. You can specify all the parameters allowed on the SIGNON command here. There is no default value.

If a signon without a password occurs in a stage1 exit, the authority is inherited from the TSO user ID used for the signon. If a signon with a password occurs in a stage1 exit, the authority of the user ID in the signon command is used.

Modifiable through MODIFY INITPARMS command: NO

## MULTI.COPY.STAT.RCD=not set | CT | MC | M2

This parameter creates statistics records for files copied using the DMDSSIOX I/O exit and is particularly useful with the wildcard feature, which can produce large numbers of files. When this parameter is set, a message is sent to the Console each time the DMDSSIOX I/O exit copies a file, regardless of what other types of statistics records are being generated. For more information on the DMDSSIOX I/O exit, see the *Utility Programs* chapter in *Connect:Direct for z/OS User's Guide*.

By default, this initialization parameter is not set. Use the following table to determine the conditions under which to use each setting.

| Value | Conditions for Use | Description |
|---|---|---|
| Not set | ◆ DMDSSIOX I/O exit is not used.<br>◆ Information for individual files is needed. | No file completion statistics records are created and no console messages are produced. |
| CT | ◆ DMDSSIOX I/O exit is used.<br>◆ Pre-existing customer exits detect files using CT records. | Produces a Copy Termination type of statistics record. |
| MC | ◆ DMDSSIOX I/O exit is used.<br>◆ Control Center is installed and monitoring the Connect:Direct server. | Produces a PDS Member Copy type of statistics record, which replaces the member name with the file name provided by the ADRDSSU utility. |
| M2 | ◆ DMDSSIOX I/O exit is used.<br>◆ None of the other conditions apply. | Produces a "multiple copy" type of statistics record, which shows the file name, current return code, bytes processed as reported by the IBM ADRDSSU utility, and normal Step information. |

Modifiable through MODIFY INITPARMS command: YES

## NETMAP.CHECK = NO | (ALL | TCP, ALL | BOTH | NODENAME, FAIL | WARN | PASS)

This parameter defines the communication types that perform NETMAP checking, the verification to perform, and the action to take if the node does not exist. This parameter is ignored for CTCA connections.

| Value | Description |
|---|---|
| NO | This value 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 feature is convenient when another Connect:Direct node initiates contact the majority of the time. |
| ALL \| TCP | ALL enables NETMAP checking for all communication types except for TCP/IP.<br>TCP enables NETMAP checking for TCP/IP communication.<br>**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. For example: ADJACENT.NODE=((UNIX.DALLAS,5555,199.5.5.5,TCP) ENVIRONMENT = UNIX) |
| ALL \| BOTH \| NODENAME | ALL or BOTH (for SNA) enables verification on both the logical node name and APPLID/LUNAME.<br>ALL or BOTH (for TCP) enables verification on both the logical node name and IP address.<br>NODENAME enables verification on the logical node name or TCP Alias names. |
| FAIL \| WARN \| PASS | FAIL indicates that access to the system is denied.<br>WARN indicates that access is allowed, but a warning message is issued.<br>PASS indicates that access is allowed without any warning message being issued. |

You must define all three parameters to require that the Connect:Direct node establishing a session with this Connect:Direct node be defined in the network map of this node under certain conditions.

To enable NETMAP checking for all communication types, you must code the NETMAP.CHECK parameter for each. Following is an example.

```
NETMAP.CHECK=(ALL,ALL,FAIL)
NETMAP.CHECK=(TCP,NODENAME,WARN)
```

- The first entry for NETMAP.CHECK causes Connect:Direct to check all communication types, except for 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 permits access.

Modifiable through MODIFY INITPARMS command: NO

## NETMAP.CHECK.ON.CALL= YES | NO

This parameter indicates how Connect:Direct handles a HOLD=CALL Process at submit time.

| Value | Description |
|---|---|
| NO | This value specifies that the PNODE permits a submit of a Process with HOLD=CALL, even if the SNODE entry is not in the network map of the PNODE. However, the TCP.IP.DEFAULT entry must be in the network map of the PNODE. This value is the default. |
| YES | This value specifies that the PNODE does not allow a submit of a Process with HOLD=CALL, if the SNODE entry is not in the network map of the PNODE (even if a TCP.IP.DEFAULT entry exists in the network map). If you specify Yes, you must define the SNODE in the PNODE network map. |

If you specify Yes, you must define the SNODE in the PNODE network map.

Modifiable through MODIFY INITPARMS command: YES

## NON.SWAPABLE = YES | NO

This parameter specifies whether Connect:Direct is marked as non-swapable. When NON.SWAPABLE = YES, Connect:Direct is not paged out during periods of no activity.

**Note:** When Connect:Direct is running as a Connect:Direct/PLEX or when CTCA has been initialized, NON.SWAPABLE is forced to YES. Otherwise, this keyword controls the setting.

Modifiable through MODIFY INITPARMS command: NO

## PDSE.SHARING = YES | NO

This parameter indicates if the zOS PDSE sharing feature is supported by Connect:Direct. The keyword, PDSESHARING in the IGDSMSxx member in SYS1.PARMLIB, which defines the level of PDSE sharing across subsystems of a sysplex, has two possible values:

- NORMAL sharing allows users to share a PDSE only at a data set level.
- EXTENDED sharing allows users to share a PDSE at both a data set level and member level.

For more information on PDSE sharing, including requirements, and test scenarios and results for both the normal and extended mode, refer to the IBM Redbook Partitioned Data Set Extended (PDSE) Usage Guide, which you can find at www.redbooks.ibm.com/abstracts/sg246106.html?Open.

***Caution:*** Because PDSE sharing allows multiple users to open the same PDSE member for output, some operations may destroy directories and create data integrity problems. The last user to issue the STOW macro, which replaces an entry on the directory, gets their update permanently applied to the member. To ensure that updates are not lost, keep this consideration in mind so that users can take the appropriate steps.

When you initialize Connect:Direct, the IGWLSHR callable service is used to verify that the operating system can support PDSE sharing. If the operating system cannot support this PDSE sharing level, the SITA641W error message, *Level of PDSE.SHARING is not supported*, will display and initialization will continue as if the PDSE.SHARING parameter had been specified as NO.

| Value | Description |
|---|---|
| YES | Connect:Direct will support PDSESHARING as defined by the operating system.<br>**Note:** You must specify SHR as the disposition (DISP) keyword so that you can share a particular data set with other jobs. All other dispositions – OLD, NEW, RPL, and MOD – will continue to serialize the PDSE to ensure that multiple Connect:Direct processes cannot share the same PDSE. |
| NO | Connect:Direct will not support PDSE sharing. If multiple processes attempt to open the same PDSE for output at the same time, the processes terminate with one of the following error messages:<br>◆ SDEPDSRI – PDS already open for output by Connect:Direct.<br>◆ SDE0210I – Requested data set not available. Allocated to another job.<br>In addition, the processes are placed in the appropriate queue and retried according to any initialization parameters specified. |

Modifiable through MODIFY INITPARMS command: YES

## PDSENQ = YES | NO

This parameter specifies whether or not Connect:Direct serializes access of output PDSes for simultaneous directory updates from Connect:Direct and ISPF EDIT or the IBM linkage editor.

| Value | Description |
|---|---|
| YES | An ISPF/IEWL ENQUEUE (or RESERVE if the device is shared) is issued to serialize access for output PDS data sets opened with DISP = SHR. Use of PDSENQ does not prevent simultaneous directory updates from batch jobs or other sources that do not issue the same enqueues. If the ENQ or RESERVE fails, DATA SET IN USE allocation error is issued. The Process retries later if 0210 is specified in the ALLOC.CODES initialization parameter. Because the ISPF editor only enqueues a member of a PDS when an ISPF SAVE of the member is issued, a user can be in edit on the member from a PDS to which Connect:Direct is trying to copy. If you attempt to save a member being edited at the same time that Connect:Direct is copying to that PDS, the TSO/ISPF session hangs until the Connect:Direct COPY operation is complete. When the Connect:Direct COPY operation completes, the ISPF SAVE command executes, thereby overlaying the member that are recently copied by Connect:Direct. |
| NO | An ISPF/IEWL ENQUEUE is not issued. |

Modifiable through MODIFY INITPARMS command: YES

## PRTYDEF = Process priority

This parameter specifies the default priority for Processes submitted to Connect:Direct. If you do not specify priority on the Process statement, Connect:Direct uses the default priority when placing the Process on the TCQ. The priorities range from zero to 15, with 15 the highest priority. The default is 10. This parameter is not valid for LU6.2 and TCP/IP flows.

Modifiable through MODIFY INITPARMS command: YES

## QUIESCE = YES | NO

This parameter specifies whether or not Connect:Direct holds Processes from execution.

| Value | Description |
|---|---|
| YES | No DTF-to-DTF sessions are started, but you can establish interactive sessions. Any Process to be executed is placed in the WAIT queue. Refer to *Suspending and Resuming Processing on Individual Nodes* on page 47 for information on how to resume normal operations by setting SESSIONS to R (Resume) with the MODIFY command. |
| NO | Connect:Direct does not hold Processes from execution. |

In a Connect:Direct/Plex environment, this parameter applies to all Connect:Direct/Servers.

**Note:** When you initialize Connect:Direct for the first time with allocation to a TCQ created by CDTCQFIX, it is recommended that you specify YES for the QUIESCE parameter. After you delete any unwanted Processes from the TCQ, DTF activity can be resumed using the Modify command.

Modifiable through MODIFY INITPARMS command: NO

## QUIESCE.NODE = node name

This parameter indicates that all processing in the specified node is suspended and no new processing is permitted until a Resume is issued with the MODIFY command. You can specify this parameter up to 15 times, which suspends processing on up to 15 nodes.

The node name parameter is the 1–16 character local node name specified in the network map of the affected node. You can also specify a partial node name followed by an asterisk (*). For example, the following parameter suspends processing on all node names that begin with NODE.CHICAGO.

```
QUIESCE.NODE = NODE.CHICAGO*
```

If the parameter is issued on an SNODE to quiesce processing with a PNODE, the session with the PNODE is established. However, as soon as the PNODE node name is determined, the session is terminated. No processing of data occurs.

Use this parameter if you want to suspend processing on a node because of problems, but want other nodes to continue processing. You can also use it if you know that a node will be down for some time.

Refer to *Suspending and Resuming Processing on Individual Nodes* on page 47 for how to resume normal operations by setting SESSIONS to R (Resume) with the MODIFY command.

Modifiable through MODIFY INITPARMS command: NO

## REMOTE.DUMMY.PASSWORD=[ YES | INTERNAL ]

This parameter controls whether Signon and Process Start is authorized for remote nodes if a dummy password was specified during Signon.

| Value | Description |
|---|---|
| YES | Any remote can specify a dummy password to obtain Signon and Process Start authorization on the local node. |
| INTERNAL | Requires the remote node to have the Adjacent Node attribute of INTERNAL (see *Trusted Node Security* on page 80) for the authorization to occur. |

Modifiable through MODIFY INITPARMS command: NO

## REQUEUE = YES | NO

This parameter 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.

| Value | Description |
|---|---|
| YES | Places the Process in the hold queue if it did not end with any of the errors listed above but ABENDed with a return code greater than 4 and one of the following is true:<br>◆ The Process or SUBMIT command has REQUEUE = YES<br>◆ Neither the Process nor the SUBMIT command has REQUEUE specified, but REQUEUE = YES is specified in the initialization parameters<br>◆ The data set on the PNODE side is a tape data set |
| NO | Executes the remaining steps in a Process following a failed COPY STEP, but the failed COPY STEP is not requeued. If REQUEUE is specified on a PROCESS or SUB statement, it overrides the initialization REQUEUE specification. |

This parameter is only 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 places the Process in the hold queue if it did not end with any of the errors listed above but ABENDed with 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 and 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 is HE (hold/error). If the Process received a return code greater than 4, the status is RH (restart/held).

Modifiable through MODIFY INITPARMS command: YES

## RESET.ORIGIN.ON.SUBMIT = YES | NO

This parameter resets the originating node.

| Parameter | Description |
|---|---|
| YES | The originating node is set to the node where the submit is issued. This action applies to all connection types. |
| NO | The originating node is not set to the node where the submit is issued. |

This parameter only affects Processes submitted to the SNODE that use SNODEID or SUBMIT. If you use this parameter, both the sending and receiving nodes must use this parameter. Also test its impact, especially if you or your trading partner use Secure Point-of-Entry (SPOE). If you use SPOE and apply this parameter, you may need to update AUTHFILE entries for user ID/node combinations used by SPOE.

Modifiable through MODIFY INITPARMS command: YES

## REUSE.SESSIONS = YES | NO

Enables you to control the use of the sessions initiated by the local node. When you select a Process 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.

| Value | Description |
|---|---|
| YES | The previously allowed negotiation takes place as described. |
| NO | The remote DTF is not allowed to utilize 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.) |

Modifiable through MODIFY INITPARMS command: YES

## ROUTCDE.CRIT = (route code)

This parameter specifies the route code used for critical WTO messages. You can specify as many as 16 codes. Suppress these messages by typing 0 for the route code.

ROUTCDE.CRIT = (2,8) specifies master console information and teleprocessing control.

ROUTCDE.CRIT = (8,11) specifies teleprocessing control and programmer information. This value is the default.

Modifiable through MODIFY INITPARMS command: YES

## ROUTCDE.NORM = (route code)

This parameter specifies the route code used for normal WTO messages. You can specify up to 16 codes. You can suppress these messages by typing 0 for the value.

ROUTCDE.NORM = (2,11) specifies master console information and programmer information.

ROUTCDE.NORM = (11) specifies programmer information. This value is the default.

Modifiable through MODIFY INITPARMS command: YES

## ROUTCDE.TAPE = (route code)

This parameter specifies the route code used for the tape mount message issued by Connect:Direct for z/OS. You can specify as many as 16 codes. A specification of 0 suppresses the tape mount message.

ROUTCDE.TAPE = (3,5,11) specifies tape pool, tape library, and programmer information.

ROUTCDE.TAPE = (5,11) specifies tape library and programmer information. This value is the default.

**Note:** If ROUTCDE.TAPE = 0 is coded and TAPE.PREMOUNT = NO, then the z/OS system mount processing at allocation time holds an ENQ on SYSZTIOT until mount is satisfied.

Modifiable through MODIFY INITPARMS command: YES

## RUN.JOB.EXIT = modname

This parameter specifies the name of the Connect:Direct module responsible for user read/write control of job streams. The module name can be from 1–8 characters long. The first character must be alphabetic. No default exists for this parameter.

Specify one of the following interface programs to use with Connect:Direct for z/OS:

- RUN.JOB.EXIT = DMGACFRJ (CA-ACF2)
- RUN.JOB.EXIT = DMGRACRJ (RACF)
- RUN.JOB.EXIT = DMGRACRJ (CA-TOP SECRET)

Sterling Commerce ships these sample programs as part of the Connect:Direct for z/OS sample library. They may not meet the normal security requirements of an installation. Modify them accordingly.

You must define a user on all nodes involved in Process execution.

Modifiable through MODIFY INITPARMS command: NO

## RUNJOBID = USER | CD

This parameter specifies the type of security environment in force for Connect:Direct RUNJOB Processes.

| Value | Description |
|---|---|
| USER | This value specifies that the Process runs under the ID of the user. This value is the default. |
| CD | This value specifies that the Process runs under the Connect:Direct DTF ID. |

Modifiable through MODIFY INITPARMS command: NO

## RUN.TASK.EXIT = modname

This parameter 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 1–8 characters; the first character must be alphabetic. No default exists for this parameter.

Specify one of the following interface programs to use with Connect:Direct for z/OS:

- RUN.TASK.EXIT = DMGACFRT (CA-ACF2)
- RUN.TASK.EXIT = DMGRACRT (for RACF and CA-TOP SECRET)
- RUN.TASK.EXIT = DMGSAFRT (CA-ACF2 with SAF enabled)

Sterling Commerce ships these sample programs as part of the Connect:Direct for z/OS sample library. They may not meet the normal security requirements of an installation. Modify them accordingly.

You must define a user on all nodes involved in Process execution.

Modifiable through MODIFY INITPARMS command: NO

## RUNTASK.RESTART = YES | NO

This parameter determines whether a RUN TASK program executes at restart if Connect:Direct is unable to determine whether the program has run.

| Value | Description |
|---|---|
| YES | RUN TASK program executes at restart if Connect:Direct is unable to determine whether the program has run |
| NO | RUN TASK program does not executes 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.

Modifiable through MODIFY INITPARMS command: YES

## SECURE.DSN = filename

This parameter specifies the Secure + Option parameters file.

Modifiable through MODIFY INITPARMS command: NO

## SECURE.SSL.PATH.PREFIX = prefix

This parameter specifies the prefix location of the key database that contains the certificates for the SSL protocol. Use this parameter if you are using SSL security with the Connect:Direct Secure+ Option for z/OS and you are operating in a CD/Plex environment.

Modifiable through MODIFY INITPARMS command: NO

## SECURITY.EXIT = (module name, DATASET | ALL, PSTKT) | OFF
## SECURITY = (module name, DATASET | ALL, PSTKT) | OFF

This parameter specifies the name of the Connect:Direct exit which performs security checking. A sample security exit, DMGSAF, is provided in the SAMPLIB. You can modify this exit if it does not meet your security requirements.

| Value | Description |
|---|---|
| module name | A name 1–8 alphanumeric characters long, with the first character alphabetic. |
| 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. |

| Value | Description |
|---|---|
| PSTKT | Indicates that the local DTF RACF security is defined to accept RACF PassTicket passwords. For more information, see *Generating RACF PassTickets* on page 61. |
| OFF | Specifies that no security exists; all requests are valid. |

If you do not specify the SECURITY.EXIT parameter or it 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 z/OS, specify SECURITY.EXIT = OFF until a security exit is installed.

A user must be defined on all nodes involved in Process execution.

The default is the Connect:Direct Authorization Facility.

**Note:** You can also code this parameter as SECURITY=

Modifiable through MODIFY INITPARMS command: NO

## SECURITY.NOTIFY = YES | NO | HOLD

This parameter specifies whether Connect:Direct sends a message to users informing them of security failures on Processes they have submitted.

| Value | Description |
|---|---|
| YES | Specifies that Connect:Direct sends a message to users informing them of security failures on Processes they have submitted. If you set the SECURITY.NOTIFY initialization parameter to YES and you specify NOTIFY = %USER or NOTIFY = *user ID* on the Process statement, a security failure sends a TSO notification to the user specified in the NOTIFY parameter of a SUBMIT or PROCESS statement. |
| NO | Specifies that Connect:Direct does not send a message to users informing them of security failures on Processes they have submitted. |
| HOLD | Specifies that Connect:Direct places 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 = *user ID* specified. If a stage 2 security error occurs on the SNODE, the user ID is not notified. The user ID 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 = *user ID* specified. If a stage 2 security error occurs on the SNODE, the user ID is notified. The user ID is also notified of all other errors or normal completion. All messages and return codes are in the Statistics File.

Modifiable through MODIFY INITPARMS command: NO

## SNA = YES | NO

This parameter specifies if Connect:Direct initializes with SNA support. You must also specify a valid VTAM APPLID in the local node record of the NETMAP.

| Value | Description |
|---|---|
| YES | Connect:Direct tries to open the VTAM ACB. If it cannot open the ACB, Connect:Direct prompts the operator for the next action. |
| NO | Only Processes running under TCP/IP or CTCA run or connect to the DTF. |

If you change this parameter to SNA = NO after Connect:Direct initializes, you must restart Connect:Direct.

Modifiable through MODIFY INITPARMS command: NO

## SNMP = YES | NO

Initializes the SNMP trap agent environment.

| Value | Description |
|---|---|
| YES | Enables the SNMP trap agent environment. |
| NO | Disables the SNMP trap agent environment. |

Modifiable through MODIFY INITPARMS command: YES

## SNMP.DSN = data set name | data set name (member)

This parameter identifies the data set used by the SNMP task to initialize the default trap variables and user defined trap triggers. The data set contains the trap events that you want to disable and any trap triggers you define. All traps are enabled by default. A sample data set is installed in SAMPLIB called CDSNMP.

Modifiable through MODIFY INITPARMS command: YES

## SNMP.MANAGER.ADDR = hostname | IP address

This parameter determines the TCP/IP address or hostname of the host where the SNMP network manager is initialized. By default, this address is the same as the Connect:Direct TCP/IP address, or the local hostname. In a Connect:Direct/Plex environment, the default is the TCP/IP address for the Connect:Direct Manager. You may specify the IP address as an IPV4 or IPV6 address.

This parameter is required if the SNMP network manager resides on a different host or is required to use a different TCP/IP address.

Modifiable through MODIFY INITPARMS command: YES

## SNMP.MANAGER.PORTNUM = port-number

This parameter is the TCP/IP port that is defined for UDP traffic to the SNMP network manager. The default is port number 162. If the defined UDP port number is something other than 162, this parameter is required.

Modifiable through MODIFY INITPARMS command: YES

## STAT.ARCH.CONFIRM = YES | NO

This parameter indicates whether or not Connect:Direct is to have confirmation that the contents of a statistics file pair are archived before erasing them and reusing the file pair to record new information.

| Value | Description |
|---|---|
| YES | Specifies that Connect:Direct requires confirmation before reusing the file. The Connect:Direct for z/OS utilities DMSTARRT and DMSTARBT provide archive confirmation. You can invoke these utilities from an archive Process or an archive batch job, respectively.<br>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 halts until you type 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. |
| NO | Specifies that Connect:Direct erases the file contents at the time of a pair switch regardless of whether indication that the file was archived is received.<br>**Note:** If you code the STAT.ARCH.CONFIRM parameter as YES, then also specify the STAT.SWITCH.SUBMIT parameter. |

Modifiable through MODIFY INITPARMS command: 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

This parameter specifies 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. Generating these blocks 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.

**Note:** These buffers are allocated above the 16 megabyte line.

The defaults are:

- STAT.BUFFER.ESDSDATA = 6
- STAT.BUFFER.KSDSINDX = 6
- STAT.BUFFER.KSDSDATA = 6

Modifiable through MODIFY INITPARMS command: NO

### STAT.ERROR = ABEND | DISABLE

This parameter 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.

| Value | Description |
|---|---|
| ABEND | Specifies that the DTF ABENDs with U3400. This value is the default. |
| DISABLE | Specifies that the Statistics Facility is disabled but the DTF remains active. The DTF operates normally. However, no statistics records are written. |

When an ABEND occurs within the Statistics Facility, an SVC dump is written to a SYS1.DUMPxx data set 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.

Modifiable through MODIFY INITPARMS command: NO

### STAT.EXCLUDE = (record type list)

This parameter 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 Chapter 10, *Using Connect:Direct Exits* for a complete list of record type identifiers.

You can also selectively exclude using the Statistics exit. See Chapter 10, *Using Connect:Direct Exits* for information on the Statistics exit. You can also turn recording of specific record types on and off during DTF execution using the STATISTICS ON/OFF API command.

The following example excludes PDS member records from the statistics log.

```
STAT.EXCLUDE = (MC)
```

Statistics records are often useful or indispensable in debugging problems. Excluding records from the statistics log makes problem determination by the Sterling Commerce Customer Services staff difficult. Do not exclude the following record types:

| Record Type | Description |
|---|---|
| CT | Copy Termination |
| PS | Process Submit |
| PT | Process Termination |
| RJ | Run Job |
| RT | Run Task |
| SW | Submit within Process |
| WO | Write to Operator (WTO) |

No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: YES

## STAT.INIT = WARM | COLD

This parameter specifies whether or not to erase the contents of the statistics files for the DTF at initialization time.

| Value | Description |
|---|---|
| WARM | Specifies that the system does not erase the contents at DTF initialization. In this case, statistics from prior DTF executions are available in the new execution. This value is the default. |
| COLD | Specifies that the system erases all preexisting records. Only records generated during the current execution are available. |

Modifiable through MODIFY INITPARMS command: NO

## STAT.QUEUE.ELEMENTS = statistics record queue size

This parameter specifies the size of the queue that holds statistic records to be written.

When a Connect:Direct task writes 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 write a statistics record are held until a slot in the queue becomes available.

You should choose the size of the queue based on how busy Connect:Direct is expected to be during peak-use periods. A lightly loaded system (up to 5 Processes executing concurrently) can run with a small queue, so the default queue size of 100 elements is usually adequate. Busier systems require a larger queue (200-1500 elements) to avoid performance degradation caused by queue contention.

There are two size considerations for the statistics queue, the total size and the threshold size. The total size is initially specified by the value of the STAT.QUEUE.ELEMENTS initialization parameter, and can be adjusted if necessary. For a Connect:Direct/Plex environment, the value of STAT.QUEUE.ELEMENTS is multiplied by the number of servers permitted according to the License Management Key. Each stat queue element takes up 2 KB of space and is allocated above the 16 megabyte line. The threshold size is a value smaller than the total size, and represents a portion of the queue that is reserved for peak-use periods. This helps to improve queue availability under a high workload. The threshold size is calculated automatically, and is usually one-fourth the total size. If the threshold size is inadequate, the total queue size should be increased.

The range of the STAT.QUEUE.ELEMENTS initialization parameter is 1-9999 elements, with a default of 100. A queue that is too small can degrade system performance, whereas a queue that is too large can cause wasteful storage allocation above the 16 megabyte line. In a Connect:Direct/Plex environment, the Connect:Direct/Plex Manager calculates the number of elements to allocate by multiplying the value of the STAT.QUEUE.ELEMENTS by the number of servers that the Connect:Direct/Plex Manager can support. If the resultant calculation is less than 5000, 5000 is used; if it is greater than 10000, 10000 is used.

---

**Note:** To ensure that an adequate amount of virtual storage above the line is allocated for the queue holding the statistics records in a Connect:Direct/Plex environment, specify REGION=0M on the job card that starts Connect:Direct. For more information about storage requirements, see *Planning the Installation* in Connect:Direct for z/OS *Installation Guide*.

---

Modifiable through MODIFY INITPARMS command: NO

## STAT.SNODEID = (NO | YES,NO | YES)

This parameter specifies whether the submitter's ID should be placed in the statistics record. The first subparameter applies to version 1 flows (SNA-LU0) and the second subparameter applies to version 2 flows (SNA-LU6.2/TCP).

---

**Note:** STAT.SNODEID also affects the user ID (original submitter ID or SNODEID) which is used for the executing of a SUBMIT within a Process, instead of affecting just the submitter's ID placed in the statistics record.

---

A value of YES causes the SNODEID, if present, to be placed in the statistics record.

A value of NO causes the submitter's ID to be placed in the statistics record.

Modifiable through MODIFY INITPARMS command: NO

## STAT.SWITCH.SUBMIT = dsn [member]

This parameter enables 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 can submit a batch job which in turn archives 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 Statements Guide* for information on the SUBMIT statement.

If you code the STAT.ARCH.CONFIRM parameter as YES, then also specify the STAT.SWITCH.SUBMIT 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.

No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: NO

## STAT.SWITCH.TIME = (hh:mm:ss , ...)

This parameter specifies times of day to perform a statistics file switch. The STAT.SWITCH.TIME is in 24-hour clock format. You can 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.

Modifiable through MODIFY INITPARMS command: NO

## STAT.TPREC = (start_time, end_time, snaps_per_hour)

This parameter instructs Connect:Direct to create a statistics record that contains the number of Processes and the amount of data sent and received for a node. You can use the statistics record for load balancing and tracking trends.

| Value | Description |
|---|---|
| start_time | This value indicates the start time for the statistics records creation. The valid format is hh:mm:ss.<br>Valid values are 00:00:00 through 23:59:59. |

| Value | Description |
|---|---|
| end_time | This value indicates the time when the statistics record generation ends. The valid format is hh:mm:ss.<br>Valid values are 00:00:00 through 23:59:59. |
| snaps_per_hour | Valid values are 1–60. If you specify a value of **1**, Connect:Direct takes one snapshot per hour. If you specify a value of 60, Connect:Direct takes one snapshot per minute, or 60 per hour. No default exists for this parameter. |

Modifiable through MODIFY INITPARMS command: NO

## STAT.USER = (user ID, [password])

This parameter specifies the security ID under which the statistics log is written and any archive Process or batch job runs. 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 z/OS creates this task using the security user ID from the STAT.USER parameter. The system also propagates the user ID 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.

| Value | Description |
|---|---|
| user ID | Specifies the security ID that Connect:Direct passes to a security exit. It contains 1–8 characters.<br>**Note:** Certain Connect:Direct statistics records are written with the STAT.USER ID in their user ID field. For example, the S2 records that contain information about the statistics logging Process are written with this ID. Because user ID is one of the indexed statistics record fields, specifying a unique ID facilitates the rapid retrieval of these records through the SELECT STATISTICS command when the TYPE and USER selection criteria are specified. |
| password | Specifies the current security password. The security exit uses this parameter to validate the current security password. It contains 1–8 alphanumeric characters. |

If you do not specify this parameter, or if you do not implement the stage 2 security exit, the statistics logging task runs with the security ID of the DTF job, and with the user ID of NDM. In this case, the TP and S2 records are written with NDM in their user ID fields.

Modifiable through MODIFY INITPARMS command: NO

### STATISTICS.EXIT = modname | (modname[,MANAGER | SERVER | BOTH])

This parameter specifies the name of the Connect:Direct statistics exit module you can invoke to complement the Connect:Direct statistics gathering functions. Use this program to log Connect:Direct information, perform IBM system management facilities (SMF) functions, and log custom information. To use this feature, you must explicitly code this initialization parameter and supply a module name—the default is no security exit.

The Connect:Direct sample statistics exits DMGSMF, STATEXMC, and STATEXIT are located in the $CD.SAMPLIB.

In a Connect:Direct/Plex environment, you can specify whether the Manager, the server(s) or both launch the statistics exit. By specifying the MANAGER keyword, only the Manager region launches the statistics exit. By specifying SERVER, all servers launch the statistics exit. Specifying BOTH causes all regions—servers and the Manager—to launch the statistics exit and for the same records to be processed twice.

In a Connect:Direct/Plex environment, MANAGER is the default entity to launch the exit, but you must still specify the module name.

Modifiable through MODIFY INITPARMS command: NO

### STRNO.MSG = number | 5

This parameter specifies the number of strings allowed to message file processing. The acceptable range is 5–100. The default value is 5.

Modifiable through MODIFY INITPARMS command: NO

### SUBMIT.EXIT = modname

This parameter 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. No default exists for this parameter.

The Connect:Direct sample exit is named SUBMEXIT and located in the $CD.SAMPLIB.

Modifiable through MODIFY INITPARMS command: NO

### SYSOUT = class

This parameter specifies the JES output class for spool output generated during DTF execution. The class must be one character in length. No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: YES

## TAPE.PREMOUNT = YES | NO | LIST

This parameter specifies whether the Connect:Direct for z/OS tape premount message is displayed.

| Value | Description |
|---|---|
| YES | Tape premount message is displayed. |
| NO | Tape premount message is not displayed. |
| LIST | LIST instructs Connect:Direct to list up to 10 tape volume serial number on the SVST000I tape premount message. |

**Note:** If ROUTCDE.TAPE = 0 is coded and TAPE.PREMOUNT = NO, then the z/OS system mount processing at allocation time holds an ENQ on SYSZTIOT until mount is satisfied.

Modifiable through MODIFY INITPARMS command: YES

## TAPEIO = EXCP | BSAM

This parameter controls tape processing and can be particularly beneficial when using virtual tape systems and real tape devices.

| Value | Description |
|---|---|
| EXCP | All I/O is performed using the EXCP method that Connect:Direct has always used. |
| BSAM | All I/O is performed using the BSAM routines defined within Connect:Direct. This is a more efficient way of processing I/O to virtual and real tape devices. However, for TAPE to TAPE copy between two z/OS nodes when one node has specified TAPEIO=BSAM and the other specifies TAPEIO=EXCP, there is no noticeable improvement. |

**Note:** For a TAPE to TAPE copy between two z/OS nodes using ASCII tapes, use the same TAPEIO parameter setting for both nodes. If one node has TAPEIO=BSAM specified and the other TAPEIO=EXCP, then you must code OPTCD=Q in the Process for the node with TAPEIO=EXCP.

Modifiable through MODIFY INITPARMS command: NO

## TAPEMOUNT.EXIT = modname

This parameter specifies the name of interface to StorageTek Tape Silo software, which provides status information on the volumes to satisfy a tapemount request. You can invoke the exit prior to a tape VOLSER mount request to automatically cancel the request should any volume not be available for the Silo to process. For more information, see *Tapemount Exit* on page 248.

The module name can be from 1–8 alphanumeric characters long, with the first character alphabetic. No default exists for this parameter.

The Connect:Direct sample exit is named DMGTAPEX and located in the $CD.SAMPLIB.

Modifiable through MODIFY INITPARMS command: NO

## TCP = OES | NO

This parameter specifies whether the TCP/IP connection modules are loaded during initialization and if so, the type of modules.

| Value | Description |
|---|---|
| OES | Specifies the TCP/IP OpenEdition Sockets Interface support.<br>**Note:** You must install and run the IBM BPX facility, the series of IBM programs that comprise the OES functionality, before you can transfer files using the TCP = OES support of the Connect:Direct. In addition, RACF sites must install the OMVS security segment before using the OES interface to transfer files. |
| NO | Causes no modules for the TCP/IP connection to load during initialization. |

Modifiable through MODIFY INITPARMS command: NO

## TCP.ADDR=nnn.nnn.nnn.nnn | xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx | ANYADDR | ANYADDR6 | HOSTNAME

***Caution:*** TCP.ADDR cannot co-exist with either TCP.LISTEN or TCP.API.LISTEN in the initialization parameters file. Specifying these parameters together terminates the initialization, and generates an error.

TCP.ADDR is soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN.

If multiple addresses are required, use TCP.LISTEN or TCP.API.LISTEN.

This parameter specifies a single TCP/IP address of IPv4, IPv6, ANYADDR, or ANYADDR6 used for BIND in a stand-alone server environment.

| Value | Description |
|---|---|
| nnn.nnn.nnn.nnn | Specifies the IPv4 TCP/IP address used for BIND in a stand-alone server environment. |
| xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx | Specifies the IPv6 TCP/IP address used for BIND in a stand-alone server environment |

| Value | Description |
|---|---|
| ANYADDR | Causes Connect:Direct to listen on IP address 0.0.0.0. This value enables Connect:Direct to listen on multiple IP addresses associated with a TCP/IP stack, and accepts only an IPv4 address. Disables VIPA support because Connect:Direct does not bind to the dynamic VIPA address. This value is the default. |
| ANYADDR6 | When you specify ANYADDR6, 0::0 or (::), both IPv4 and IPV6 connection requests are accepted through this listen. |
| Hostname | Specifies a hostname (up to 256 bytes) that can be translated to an IP address using DNS lookup. |

The default is the home IP address of the primary TCP on the z/OS image on which Connect:Direct is initializing.

In a Connect:Direct/Plex environment, the CDPLEX.TCPIP= specification overrides the TCP.ADDR= specification.

Modifiable through MODIFY INITPARMS command: NO

## TCP.API.LISTEN = ((addr , port) , (addrn , portn))

***Caution:*** TCP.API.LISTEN cannot co-exist with any of the following parameters in the initialization parameters file: TCP.ADDR, TCP.NAME, TCP.PORTNUM, TCP.API.PORTNUM, CDPLEX.TCPIP or CDPLEX.TCPNAME. Specifying any of these parameters with TCP.API.LISTEN terminates the initialization, and generates an error.

TCP.ADDR, TCP.NAME, TCP.PORTNUM, TCP.API.PORTNUM, CDPLEX.TCPIP and CDPLEX.TCPNAME are soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN to define a single or multiple listening task.

Use this parameter to define up to eight different address and port combinations for each server for incoming connection requests.

The TCP.API.LISTEN parameter allows for a list of IP address and port number combinations to support multiple addresses, including IPv6.

For each server in a CD Plex environment, override the global initialization parameter by specifying the TCP.API.LISTEN parameter in that server's local initialization parameters. The first address defined in the parameter becomes the local or default address.

There is no default for TCP.API.LISTEN if the parameter is not specified. If the parameter is specified with an address only, the port can default. The default port for TCP.API.LISTEN is 1363.

The syntax for the TCP.API LISTEN parameter is similar to the following example:

```
TCP.API.LISTEN = ( (addr1 , port1) , (addrn , portn) )
```

In the example, addr1 through addrn is specified as either the word ANYADDR or ANYADDR6, a TCP hostname or a specific IP address. Also, port1 through portn is specified as a single port number. The following example demonstrates this specification:

```
TCP.API.LISTEN =(MVSA , 1363)                /* establish single API listen
```

| Value | Description |
|---|---|
| ANYADDR6 | When you specify ANYADDR6, 0::0 or (::), both IPv4 and IPv6 connection requests are accepted through this listen. To define a single listen task to accept IPv4 and IPv6 requests, specify the TCP.API.LISTEN parameter as follows.<br><br>TCP.API.LISTEN=(ANYADDR6,1364) |

Modifiable through MODIFY INITPARMS command: NO

## TCP.API.PORTNUM = nnnnn

***Caution:*** TCP.API.PORTNUM cannot co-exist with either TCP.LISTEN or TCP.API.LISTEN in the initialization parameters file. Specifying these parameters together terminates the initialization, and generates an error.

TCP.API.PORTNUM is soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN.

This parameter specifies the number for a single listen port for Connect:Direct API communications.

This parameter is required if you want to use the Connect:Direct TCP/IP API interface. It is available only if TCP=OES has been specified.

This parameter value must be different from the TCP.PORTNUM initialization parameter.

Modifiable through MODIFY INITPARMS command: NO

## TCP.FMH.TIMER=hh:mm:ss

The TCP.FMH.TIMER initialization parameter defines the length of time in hours, minutes, and seconds that the TCP session can be inactive waiting on a Connect:Direct FMH to be received from the remote node before the TCP session is terminated. The default value of 00:00:00 indicates that no timer is used. For a Connect:Direct Run Task step, the timer value does not apply.

The Connect:Direct Process is placed in the Timer Retry queue and retried after the amount of time specified by the WTRETRIES initialization parameter has elapsed. However, if the session times out a second time, the Process is placed in Held in Error status and remains in the Hold queue to give you an opportunity to analyze the problem to prevent this Process from retaining TCP resources.

Care should be taken in specifying these timer values in that Connect:Direct Processes could perform differently than expected due to the session being prematurely terminated.

Use this parameter only when necessary, and when you do, set it to an extremely high value. It is difficult to estimate how much time it takes, for example, for a file to be allocated by means of a manual tape mount. Through trial and error, you should be able to select an appropriate value for this parameter.

Modifiable through MODIFY INITPARMS command: NO

## TCP.LISTEN = ((addr , port) , (addrn , portn))

***Caution:*** TCP.LISTEN cannot co-exist with any of the following parameters in the initialization parameters file: TCP.ADDR, TCP.NAME, TCP.PORTNUM, TCP.API.PORTNUM, CDPLEX.TCPIP or CDPLEX.TCPNAME. Specifying any of these parameters with TCP.LISTEN terminates the initialization, and generates an error.

TCP.ADDR, TCP.NAME, TCP.PORTNUM, TCP.API.PORTNUM, CDPLEX.TCPIP and CDPLEX.TCPNAME are soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN to define a single or multiple listening task.

Use this parameter to define up to eight different address and port combinations for each server for incoming connection requests.

The TCP.LISTEN parameter allows for a list of IP address and port number combinations to support multiple addresses, including IPv6.

For each server in a CD Plex environment, override the global initialization parameter by specifying the TCP.LISTEN parameter in that server's local initialization parameters. The first address defined in the parameter becomes the local or default address.

There is no default for TCP.LISTEN if the parameter is not specified. If the parameter is specified with an address only, the port can default. The default port for TCP.LISTEN is 1364.

The syntax for the TCP.LISTEN parameter is similar to the following example:

```
TCP.LISTEN = ( (addr1 , port1) , (addrn , portn) )
```

In the example, addr1 through addrn is specified as either the word ANYADDR or ANYADDR6, a TCP hostname or a specific IP address. Also, port1 through portn is specified as a single port number. The following example demonstrates this specification:

```
TCP.LISTEN = ((MVSA , 1364),                    - /* using DNS for address & Default
              (10.20.129.3 , 4100),             - /* specific address, specific port
              (10.20.129.3 , 4101))
```

| Value | Description |
|---|---|
| ANYADDR6 | When you specify ANYADDR6, 0::0 or (::), both IPv4 and IPv6 connection requests are accepted through this listen. To define a single listen task to accept IPv4 and IPv6 requests, specify the TCP.LISTEN parameter as follows.<br><br>TCP.LISTEN=(ANYADDR6,1364) |

Modifiable through MODIFY INITPARMS command: NO

## TCP.NAME = TCP/IP started task name | NAME,FAIL

***Caution:*** TCP.NAME cannot co-exist with either TCP.LISTEN or TCP.API.LISTEN in the initialization parameters file. Specifying these parameters together terminates the initialization, and generates an error.

TCP.NAME is soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN.

This parameter specifies the name of the TCP/IP started task. The name can be from 1–8 alphanumeric characters long.

| Value | Description |
|---|---|
| name of TCP/IP started task | Specifies the name of a TCP/IP started task. The default is TCPIP. |
| NAME,FAIL | Causes Connect:Direct to listen on every TCP stack on the z/OS LPAR. This value provides inbound access to multiple TCP stacks. Outbound access will use the first found TCP stack. Only one stack will be used outbound. |

The TCP.NAME parameter was originally used only for Domain Name resolution and stack usage was defaulted to the configured or SYSTCPD identified stack. With multi-homed OES environments, this defaulting may no longer provide adequate control of which stack is used. This parameter has been changed to enforce the use of the TCP stack named by TCP.NAME. This is done by adding the optional FAIL parameter to TCP.NAME, that is, TCP.NAME=(NAME,FAIL). The default is not to fail initialization but to use the default stack as is currently done.

If you receive a return code of **8A** in the RPLERRCK trace file when you start Connect:Direct and you have only one TCP/IP stack defined, this is normal and no cause for concern. The specific error message reads:

```
STCO999E FUNCTION=FILESYS  RETURN CODE=0000008A  REASON CODE=11B3005A  TEXT=No such
device or address
```

However, if you are in an environment where multiple TCP/IP stacks are defined, you should investigate the situation and make sure the correct TCP stack is specified.

When NAME,FAIL is requested and initialization cannot match the named stack, as indicated by message STCO035I, no TCP/IP connections are established but stack initialization retries at one minute intervals, just as when a stack failure occurs. Initialization retries continue until the requested stack becomes available or Shutdown occurs. If stack failure occurs after initialization, the NAME,FAIL parameter is used to control which stack is used and can prevent a connection from being established until the requested stack is available again.

---

**Note:** TCP.NAME=(name,FAIL) may cause problems when using the extended recovery feature if the recovery job is directed to a different LPAR or system image. When using the FAIL option and extended recovery, be sure the correct TCP stack is specified in the recovery job's initialization parameters for the system where recovery will occur.

---

Modifiable through MODIFY INITPARMS command: NO

## TCP.PORTNUM = nnnnn

---

***Caution:*** TCP.PORTNUM cannot co-exist with either TCP.LISTEN or TCP.API.LISTEN in the initialization parameters file. Specifying these parameters together terminates the initialization, and generates an error.

TCP.PORTNUM is soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN.

---

This parameter specifies the TCP/IP server port number. The number ranges from 1–65535. The default is 1364.

Modifiable through MODIFY INITPARMS command: NO

## TCP.RUNTASK.TIMER =hh:mm:ss

The TCP.RUNTASK.TIMER initialization parameter defines the length of time in hours, minutes, and seconds that the PNODE will wait for the RUN TASK on the SNODE to complete before the TCP session is terminated. The default value of 00:00:00 indicates that no timer is used. This timer value only applies to a Connect:Direct Run Task function when the RUNTASK is being performed on the SNODE.

Modifiable through MODIFY INITPARMS command: NO

## TCP.SRC.PORTS = (ip.address,port-ranges),(ip.address2,port1,port2), - (ip.address3,port-ranges)
## TCP.SRC.PORTS = (ip.address/submask,port-ranges), …
## TCP.SRC.PORTS = (ip.address/0XFFFFFFFFFFFFFFFF,ports,ranges)

This parameter specifies a destination IP address (or multiple addresses) and the source ports associated with the destination addresses. The values in this parameter are loaded into a table that Connect:Direct uses to find a match when establishing a session. It is available only for z/OS nodes using TCP=OES.

Use a wildcard character [* or 0 (zero)] to define a destination IP address pattern. The wildcards must be in the least significant positions.

You can add an optional subnet mask for the destination IP address, followed by the source port number and/or range of port numbers for the destination IP address.

For IPv4, valid subnet mask values are:

- Dotted quad notation, such as 255.255.0.0
- Hexadecimal notation, such as 0xffffff00

For IPv4 address specification, the ip.address can be fully qualified such as 199.1.1.1, or a generic address such as 199.1.*

You cannot use a subnet mask if you use wildcards in the destination IP address pattern.

Specify the range of source ports from lowest port number to highest port number order. For example, 1025–2000 is valid, whereas 2000–1025 is invalid. The source port numbers must be between 1025–65535, inclusive.

**Note:** The number of source ports defined must be sufficient to handle the number of concurrent Connect:Direct sessions. If not, performance can be severely affected.

Following is an example.

```
TCP.SRC.PORTS =(199.2.4.*, 5000-5050),-
  (199.2.4.7, 1376),-
  (200.200.4.4/255.255.2.4, 2000-2100, 3000-3100),-
  (138.16.*.*, 2000-2050, 3000-3050, 4001, 4005)
```

For IPv6 specification, the ip.address is specified in IPv6 format with colons. It can be fully qualified such as 1:2:3:4:5:6:7:8 or a generic address 1:2:3:4:*. Generic specification of the IPv6 ip.address cannot use the shortcut specification. For example, 1111:0:0:0:0:6666:7777:8888 can be specified as 1111::6666:7777:8888 as a fully qualified name. However, 1111::6666:* is not allowed because there is no way to know how many zeros have been eliminated.

For IPv6, valid subnet mask values are:

- Hexadecimal notation, such as 0XFFFFFFFFFFFFFFFFFFFFFFFF00000000

**Note:** If the specification of an IPv6 address, mask and ports takes more than one line, split at the slash dividing the ip.address from the submask.

The following example demonstrates how to specify an IPv6 address, mask and ports over more than one line:

```
TCP.SOURCE.PORTS=(1111:2222:3333:4444:5555:6666:7777:8888/ -
    0XFFFFFFFFFFFFFFFFFFFFFFFF00000000,02000-03000,03500), -
  (199.1.1.2/255.255.0.0,04000-05000,05500)
```

Modifiable through MODIFY INITPARMS command: NO

## TCP.SRC.PORTS.LIST.ITERATIONS = number of scans

This parameter specifies the number of times that Connect:Direct scans the available ports list to attempt a connection before going into a retry state. Use any value between 0 and 255.

Modifiable through MODIFY INITPARMS command: NO

## TCP.TIMER = wait time

This parameter specifies the number of seconds that a Process waits on a TCP or UDT data read before the Process is cancelled and put in timer retry status. The number can range from 0–32767. The default value of 0 indicates that no timer is used.

Use a value of 60 or greater to keep Processes from waiting indefinitely because of a lost connection.

This parameter can have the following impacts on Processes:

1. Setting the parameter to 0 causes a Process to become stranded and requires you to manually requeue and restart the Process.
2. Setting this parameter to a high value (such as 1800 [a half-hour] or 3600 [an hour]) to allow long running tasks to complete prevents any shutdowns during that time. Also, long running tasks tie up communications links and associated resources for the time period. System resources are used more efficiently by breaking a task into smaller units, with a RUN JOB submitting the long running Process.

Modifiable through MODIFY INITPARMS command: NO

## TCQ = WARM | COLD

This parameter specifies how the TCQ is initialized.

| Value | Description |
|---|---|
| WARM | Connect:Direct uses the TCQ as it exists. |
| COLD | Connect:Direct reinitializes the TCQ and any Processes left on the TCQ are lost. |

Modifiable through MODIFY INITPARMS command: NO

## TCQ.THRESHOLD = NO | YES | nn

This parameter specifies how Connect:Direct issues warning messages as the TCQ reaches a defined capacity.

| Value | Description |
|---|---|
| No | Indicates that a warning message is not produced when control intervals (CIs) reach a certain level on the TCQ file. A message is issued when the TCQ is completely full. This value is the default. |
| Yes | Indicates that a warning message is issued when the TCQ becomes 90% full. The message is reissued when the percentage changes but remains above 90% full. |
| nn | Allows you to specify a 2-digit percentage of the number of VSAM file control intervals used on the TCQ file. When the TCQ reaches this percentage, Connect:Direct issues a message. The message is reissued when the percentage changes and remains above the percentage specified here. The range for percentage value is 0 through 99.<br>**Note:** A value of 0 is the equivalent of the default NO value, that is, no warning messages are produced; only when the TCQ is completely full is a message issued. |

Modifiable through MODIFY INITPARMS command: YES

## THIRD.DISP.DELETE = YES | NO

The PNODE setting for the THIRD.DISP.DELETE= parameter controls how Connect:Direct processes the value of the third subparameter of the (TO)DISP= parameter during a COPY operation when an ABEND occurs in the COPY step on the receiving node.

| Value | Description |
|---|---|
| YES | Enables the third disposition delete feature. |
| NO | Disables this feature. |

The following table describes the conditions under which THIRD.DISP.DELETE=YES is enforced and the result it produces.

| This processing occurs | When all of these conditions are present |
|---|---|
| The data set on the receiving node is deleted. | ABEND occurs in the COPY step on the receiving node (SNODE). |
| | THIRD.DISP.DELETE=YES. |
| | Third subparameter of the TO(DISP) parameter is not defined, or is defined as DELETE. |
| | Third subparameter of the TO(DISP) parameter is not defined as KEEP or CATLG. |

| This processing occurs | When all of these conditions are present |
|---|---|
| | The destination output file does not exist, that is, the COPY statement creates a file, which means that if RPL is specified, there is no file to replace. |
| | The DISP parameter of the TO side of the COPY statement is set to one of the following options:<br>◆ DISP=(NEW,CATLG)<br>◆ DISP=(NEW,CATLG,DELETE)<br>◆ DISP=(NEW,KEEP)<br>◆ DISP=(NEW,KEEP,DELETE)<br>◆ DISP=(RPL,CATLG)<br>◆ DISP=(RPL,CATLG,DELETE)<br>◆ DISP=(RPL,KEEP)<br>◆ DISP=(RPL,KEEP,DELETE) |
| | The Process is ineligible for REQUEUE because one of the following is true:<br>◆ Checkpointing is turned OFF.<br>◆ REQUEUE=NO is specified in the Process statement or as an initialization parameter. |

When THIRD.DISP.DELETE=NO is set, the following processing occurs:

- ✦ Connect:Direct does not apply the third subparameter value, even if it is set to DELETE in the Process itself.
- ✦ Connect:Direct uses the value set for the second subparameter of the TO(DISP) parameter as the value for the third subparameter. For example, if the second subparameter is set to CATLG, the third subparameter is treated as if CATLG were specified.

***Caution:*** When you create GDG data sets using relative numbering, for example, DATA.SET.NAME(+1), you must specify CATLG as the third subparameter of the TO(DISP) parameter. If you fail to do so, a RETRY failure occurs. For example, you can specify DISP=(NEW,CATLG,CATLG) or DISP=(,CATLG,CATLG).

Modifiable through MODIFY INITPARMS command: YES

## TRACE.BUFFER = nnn | 2

This parameter specifies how much space in 64-bit storage is allocated for tracing. By default, two megabytes are allocated for the in-storage wrap trace buffer. To turn off in-storage tracing, specify TRACE.BUFFER=0.

Modifiable through MODIFY INITPARMS command: NO

## TRANS.SUBPAS = YES|NO

This parameter specifies whether the submitter password is sent to the receiving node if the receiving node submits within a Process back to the submitting node.

| Value | Description |
|---|---|
| YES | The submitter password is sent to the receiving node |
| NO | The submitter password is not sent to the receiving node |

Modifiable through MODIFY INITPARMS command: YES

## UDP.SRC.PORTS = (ip.address,port-ranges),(ip.address2,port1,port2), - (ip.address3,port-ranges)

This parameter specifies a destination IP address (or multiple addresses) and the source ports associated with the destination addresses to use when making UDT connections. The values in this parameter are loaded into a table that Connect:Direct uses to find a match when establishing a session.

Use a wildcard character [* or 0 (zero)] to define a destination IP address pattern. The wildcards must be in the least significant positions.

Specify the range of source ports from lowest port number to highest port number order. For example, 1025–2000 is valid, whereas 2000–1025 is invalid. The source port numbers must be between 1025–65535, inclusive.

**Note:** The number of source ports defined must be sufficient to handle the number of concurrent Connect:Direct sessions. If not, performance can be severely affected.

Modifiable through MODIFY INITPARMS command: NO

## UDP.SRC.PORTS.LIST.ITERATIONS = number of scans

This parameter specifies the number of times that Connect:Direct scans the available ports list to attempt a UDT connection before going into a retry state. Use any value between 0 and 255.

Modifiable through MODIFY INITPARMS command: NO

## UDT = YES|NO

This parameter specifies whether UDT connection modules are loaded during initialization.

| Value | Description |
|---|---|
| YES | Loads modules for the UDT connections during initialization. |

| Value | Description |
|---|---|
| NO | Causes no modules for UDT connections to load during initialization. |

Modifiable through MODIFY INITPARMS command: NO

## UDT33.LISTEN = ((ipaddr,port)[,(ipaddr,port)…])

Use this parameter to define up to eight different address and port combinations for each server for incoming connection requests.

The UDT33.LISTEN parameter allows for a list of IP address and port number combinations to support multiple addresses, including IPv6.

For each server in a CD Plex environment, override the global initialization parameter by specifying the UDT33.LISTEN parameter in that server's local initialization parameters. The first address defined in the parameter becomes the local or default address.

The default port for UDT33.LISTEN is 1366.

| Value | Description |
|---|---|
| ANYADDR6 | When you specify ANYADDR6, 0::0 or (::), both IPv4 and IPv6 connection requests are accepted through this listen. To define a single listen task to accept IPv4 and IPv6 requests, specify the UDT33.LISTEN parameter as follows.<br><br>UDT33.LISTEN=(ANYADDR6,1364) |

Modifiable through MODIFY INITPARMS command: NO

## UPPER.CASE = YES|NO

This parameter specifies whether initialization messages sent to the console are displayed in uppercase letters.

| Value | Description |
|---|---|
| YES | Initialization messages sent to the console are displayed in uppercase letters |
| NO | Initialization messages sent to the console are displayed in uppercase letters |

**Note:** You must define this parameter on the execution parameter overrides to ensure that all initialization messages are displayed in uppercase letters.

Modifiable through MODIFY INITPARMS command: NO

## V2.BUFSIZE = (maximum buffer size for this transmission, TCP/IP send/receive buffer size)

The first positional parameter specifies the maximum buffer size that Connect:Direct uses for LU6.2 and TCP/IP data transmission. Valid values range from 3712–262144. The default is 4096.

The second positional parameter is used to alter the TCP/IP send and receive buffer sizes within TCP/IP. The maximum value is 256KB. Although the minimum value is set by TCPCONFIG in the TCP/IP PROFILE data set, the absolute minimum value is 16KB. If a value is specified that is lower than the TCPCONFIG value, the TCPCONFIG value is used. The default of the second parameter is double the first parameter, unless this is lower than the value set by TCPCONFIG in the TCP/IP stacks PROFILE data set. For example, TCPCONFIG TCPSENDBFRSIZE 64K TCPRCVBFRSIZE 64K in the TCP/IP PROFILE would set the default to 64KB. This default value of 64KB would be used unless the first parameter is greater than 32 KB, or the second parameter is greater than 64 KB.

Modifiable through MODIFY INITPARMS command: NO

## WTMESSAGE = NO | YES | (YES,nnn)

This parameter specifies whether a WTO message is generated when a Process is placed on the timer retry queue.

| Value | Description |
|---|---|
| YES | Specifies that message SVTM110I is written to the console each time a Process is placed on the timer queue. |
| NO | Specifies that message SVTM110I is not written to the console each time a Process is placed on the timer queue. |
| (YES,nnn) | Specifies the number of times the SVTM110I messages are written to the console. Valid value for nnn is 1–512. The default is 1, which specifies that the message is written every time. If you specify 2, every second message is displayed, and so on. |

Connect:Direct uses this parameter with the MAXRETRIES initialization parameter when attempting to establish a lost session.

Modifiable through MODIFY INITPARMS command: YES

## WTRETRIES = hh:mm:ss | 00:03:00

This parameter specifies the amount of time between attempts to reestablish a node-to-node session. The default is 00:03:00. Connect:Direct uses this parameter with the MAXRETRIES initialization parameter when trying to establish a lost session.

Modifiable through MODIFY INITPARMS command: YES

### XCF.NAME = XCF group name

This parameter specifies a unique name for a Connect:Direct/Plex environment, or for a Connect:Direct/Stand-alone Server (including the standby system) using extended recovery. This parameter is not needed in a Connect:Direct/Stand-alone Server that does not use extended recovery.

XCF group names cannot begin with the letters “A” through “J” or with the letters “SYS” because these are reserved by IBM. You cannot modify this parameter through the MODIFY INITPARMS command.

Modifiable through MODIFY INITPARMS command: NO

## Connect:Direct System File Initialization Parameters

This section lists initialization parameters for Connect:Direct system files. You must specify each initialization parameter.

These initialization parameters are considered global parameters in a Connect:Direct/Plex environment.

### APDSN = dsn

This parameter defines the APFILE data set that contains the asset protection key.

**Note:** If necessary, see the Connect:Direct for z/OS *Installation Guide* for instructions on how to obtain a temporary or permanent license management key (also known as the asset protection key). The APKey file is required to run Connect:Direct for z/OS.

Modifiable through MODIFY INITPARMS command: YES

### AUTHDSN = dsn

This parameter specifies the file name of the Connect:Direct VSAM Authorization file. No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: NO

### CKPTDSN = dsn

This parameter specifies the file name of the Connect:Direct VSAM Checkpoint/restart file. No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: NO

### MSGDSN = dsn

This parameter specifies the file name of the Connect:Direct VSAM Message file. No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: NO

## NETDSN = dsn

This parameter specifies the file name of the Connect:Direct VSAM network map file. No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: NO

## STAT.ARCH.DIR = archive directory file name

This parameter 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 is useful in locating the archive file containing records for a specific date/time. When you omit this parameter, the archive directory functions are unavailable. No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: NO

## STAT.DSN.BASE = dsname base
## STAT.FILE.PAIRS = number

The STAT.DSN.BASE parameter specifies the high-level qualifiers for the statistics files cluster names. Use any valid z/OS data set name qualifiers for this parameter. The high-level qualifier can 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  */
```

Modifiable through MODIFY INITPARMS command: NO

## TYPEDSN = dsn

This parameter specifies the file name of the Connect:Direct VSAM Type file. No default exists for this parameter.

Modifiable through MODIFY INITPARMS command: NO

# Appendix B

# Local Initialization Parameters

This appendix lists local initialization parameters, descriptions, and default values (indicated by underlined text).

Local initialization parameters are only used in a Connect:Direct/Plex environment. They only apply to a specific member of the Connect:Direct/Plex.

You can override any initialization parameter allowed in the local initialization parameters file using the PARM= keyword in the EXEC statement.

The *Connect:Direct for z/OS Quick Reference* also contains a table of initialization parameters and their default values.

## CDPLEX.MANAGER = NO | YES

This parameter indicates whether this Connect:Direct/Plex member is a Connect:Direct/Manager or a Connect:Direct/Server. Only one Connect:Direct/Manager is allowed in a Connect:Direct/Plex.

| Value | Description |
|---|---|
| YES | Indicates that this Connect:Direct/Plex member is a Connect:Direct/Manager or a Connect:Direct/Server. |
| NO | Indicates that this Connect:Direct/Plex member is not a Connect:Direct/Manager or a Connect:Direct/Server. This is the default value. |

You cannot modify this parameter through the MODIFY INITPARMS command.

## CDPLEX.MSGID = NONE | xx

This parameter specifies two characters that identify the Connect:Direct/Plex member. These characters display after the message number in messages sent to a console operator. They enable the operators to identify the message source.

Following is an example of the message display if the CDPLEX.MSGID value is set to S1. The two character identifier is highlighted in this example.

```
SVTM055I S1 SESSION (001) ESTABLISHED WITH       SNODE=SC.DUB.TPYLA2
SVTM055I S1 SESSION (001) ESTABLISHED WITH       PNODE=SC.DUB.TPYLA2
SVTM036I S1 PROCESS STARTED   MVSMVST3(       1) PNODE=SC.DUB.TPYLA2
SVTM036I S1 PROCESS STARTED   MVSMVST3(       1) SNODE=SC.DUB.TPYLA2
```

If your site uses automated operations monitoring and you use this parameter to identify Connect:Direct/Plex members, you may need to revise your monitoring program because of the change in message format. In this case, you may want to leave this parameter at NONE.

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.PLEXCLASSES = (*,plexclass,…,plexclass)

This parameter specifies which PLEXCLASSes are supported by the Connect:Direct/Server. You can restrict a Connect:Direct/Server to run only jobs in the specified PLEXCLASSes. An asterisk (*) indicates that the Connect:Direct/Server supports any Process that does not specify a PLEXCLASS or specifies a PLEXCLASS of '*'.

If you specify CDPLEX.PLEXCLASSES, you must explicitly specify '*' as its value to run any Processes with a Plexclass of '*' on that Connect:Direct/Server.

The PLEXCLASS name is 1–8 characters long and up to 8 classes may be specified.

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.REDIRECT = ((=((INT_IPv4,EXT_IPv4),(INT_IPv6,EXT_IPv6), (INT_UDT_IPv4,EXT_UDT_IPv4),(INT_UDT_IPv6,EXT_UDT_IPv6))

This parameter is used by the Connect:Direct Plex Manager in the Connect:Direct Plex environment to determine the redirection address that is presented to the remote node. It allows you to specify redirection addresses based on the security node type (internal or external) and session type (TCP or UDT33)of the adjacent node in the network map for the Connect:Direct Plex Servers within the local initialization parameters.

The CDPLEX.REDIRECT parameter is a local parameter for the servers only, and defaults to the first IPv4 or IPv6 address indicated for the server.

The following shows an example of the syntax for the CDPLEX.REDIRECT parameter followed by an example with actual TCP/IP and UDT addresses and port numbers.

```
CDPLEX.REDIRECT=((INT TCP IPV4 , PORT),(EXT TCP IPV4 , PORT ) -
                 (INT TCP IPV6 , PORT),(EXT TCP IPV6 , PORT ) -
                 (INT UDT IPV4 , PORT),(EXT UDT IPV4 , PORT ) -
                 (INT UDT IPV6 , PORT),(EXT UDT IPV6 , PORT ) )
```

```
CDPLEX.REDIRECT=((10.20.201.2,4199),(199.1.22.333,4199) -
                 (FD00::22CE:0:0:0:82,4299),(FD00::22CE:1:2:3:99,4299) -
                 (10.20.201.2,4399),(199.1.22.333,4399) -
                 (FD00::22CE:0:0:0:82,4499),(FD00::22CE:1:2:3:99,4499) )
```

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.SERVER = Connect:Direct/Server name

This parameter specifies the name of the Connect:Direct/Server. Use only for Connect:Direct/Plex members whose CDPLEX.MANAGER parameter value is NO. Each Connect:Direct/Server in the Connect:Direct/Plex must have a unique 1–8 character name. The Connect:Direct/Server name cannot be the same as the XCF.NAME name. (The XCF.NAME name is used as the Connect:Direct/Manager name).

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.SERVER.JOBDSN = data set name

This parameter specifies the data set name that contains the various jobs or started tasks that are submitted at initialization. This parameter is only valid for the Connect:Direct/Manager.

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.SERVER.JOBMEM = ((member name,server name),…)

This parameter specifies the CDPLEX.SERVER.JOBDSN member containing the JCL or start command that starts the specified Connect:Direct/Server. The member name can be up to 8 characters long.

This parameter is only valid for the Connect:Direct/Manager.

You can specify a maximum of 32 member/server name combinations.

If you want to start the Connect:Direct/Server as a started task in a JES2 environment, use one of the following to issue a start command:

- To issue a START command on the same z/OS image, include START= as the first statement in the member. The member is not submitted as JCL, but a START command is issued with the rest of the first statement.

  For example, if START=HOST4100,X is the first statement, then the command START HOST4100,X is issued to z/OS.

- To issue a START command on a different z/OS image, include /*$VS,‘command’ as the first statement in the member. The member is submitted to JES, but JES identifies the /*$VS and issues the appropriate command to z/OS.

  For example, if /*$VS,‘RO CSGB,S CDICOMB’, the member is submitted to JES and the RO CSGB,S CDICOMB command is issued by JES to z/OS rather than placed in the job queue.

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.SERVER.NODE = node name

This parameter specifies a unique 1–16 character name for a Connect:Direct/Server. This parameter is not used for a Connect:Direct/Manager.

In a Connect:Direct/Plex, all members appear to external Connect:Direct systems as a single local node. However, if you use network map checking and an external Connect:Direct system communicates with more than one Connect:Direct/Server, you can specify a name to identify each Connect:Direct/Server.

The adjacent node's USE.SERVER.NODE network map parameter (see page 137) must specify that the Connect:Direct/Server use this name in the initial communications.

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.TCPIP = TCP/IP address | Hostname

***Caution:*** CDPLEX.TCPIP cannot co-exist with either TCP.LISTEN or TCP.API.LISTEN in the initialization parameters file. Specifying these parameters together terminates the initialization, and generates an error.

CDPLEX.TCPIP is soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN.

If multiple addresses are required, use TCP.LISTEN or TCP.API.LISTEN.

This parameter specifies a single TCP/IP address of the Connect:Direct/Plex member. This parameter uses the standard IPv4 address format (199.1.1.1), IPv6 address format (1:2:3:4:5:6:7:8), ANYADDR, ANYADDR6, or you can specify a hostname.

| Value | Description |
|---|---|
| nnn.nnn.nnn.nnn | Specifies the IPv4 TCP/IP address used for BIND in a stand-alone server environment. |
| xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx | Specifies the IPv6 TCP/IP address used for BIND in a stand-alone server environment |
| ANYADDR | Causes Connect:Direct to listen on IP address 0.0.0.0. This value enables Connect:Direct to listen on multiple IP addresses associated with a TCP/IP stack, and accepts only an IPv4 address. Disables VIPA support because Connect:Direct does not bind to the dynamic VIPA address. This value is the default. |
| ANYADDR6 | When you specify ANYADDR6, 0::0 or (::), both IPv4 and IPv6 connection requests are accepted through this listen. |
| Hostname | Specifies a hostname (up to 256 bytes) that can be translated to an IP address using DNS lookup. |

Modifiable through MODIFY INITPARMS command: NO

## CDPLEX.TCPNAME = name of TCP/IP started task | NAME,FAIL

***Caution:*** CDPLEX.TCPNAME cannot co-exist with either TCP.LISTEN or TCP.API.LISTEN in the initialization parameters file. Specifying these parameters together terminates the initialization, and generates an error.

CDPLEX.TCPNAME is soon-to-be obsolete. Consider migrating to TCP.LISTEN or TCP.API.LISTEN.

This parameter specifies the name of the TCP/IP started task on a Connect:Direct server. The name can be from 1–8 alphanumeric characters long. The default is TCPIP.

The CDPLEX.TCPNAME parameter was originally used only for Domain Name resolution and stack usage was defaulted to the configured or SYSTCPD identified stack. With multi-homed OES environments, this defaulting may no longer provide adequate control of which stack is used. This parameter has been changed to enforce the use of the TCP stack named by CDPLEX.TCPNAME. This is done by adding the optional NAME,FAIL parameter to CDPLEX.TCPNAME, that is, CDPLEX.TCPNAME=(NAME,FAIL). The default is not to fail initialization but to use the default stack as is currently done.

**Note:** If you receive a return code of **8A** in the RPLERRCK trace file when you start Connect:Direct and you have only one TCP/IP stack defined, this is normal and no cause for concern. The specific error message reads:

```
STCO999E FUNCTION=FILESYS  RETURN CODE=0000008A  REASON CODE=11B3005A
TEXT=No such device or address
```

However, if you are in an environment where multiple TCP/IP stacks are defined, you should investigate the situation and make sure the correct TCP stack is specified.

When NAME,FAIL is requested and initialization cannot match the named stack, as indicated by message STCO035I, no TCP/IP connections are established but stack initialization retries at one minute intervals, just as when a stack failure occurs. Initialization retries continue until the requested stack becomes available or Shutdown occurs. If stack failure occurs after initialization, the NAME,FAIL parameter is used to control which stack is used and can prevent a connection from being established until the requested stack is available again.

Modifiable through MODIFY INITPARMS command: NO

**Note:** CDPLEX.TCPNAME=(name,FAIL) may cause problems when using the extended recovery feature if the recovery job is directed to a different LPAR or system image. When using the FAIL option and extended recovery, be sure the correct TCP stack is specified in the recovery job's initialization parameters for the system where recovery will occur.

## CDPLEX.TIMER = 5 | nn

This parameter specifies the time-out value in minutes for XCF communications. The range is 0, 5–99 (0 indicates no time-out).

Modifiable through MODIFY INITPARMS command: NO

### CDPLEX.VTAM = (VTAM-APPL,P/S-Node-APPL)

This parameter specifies the VTAM APPLIDs for a Connect:Direct/Server. This initialization parameter is required for a Connect:Direct/Server if SNA=YES is specified in the global initialization parameters.

Modifiable through MODIFY INITPARMS command: NO

## Local and Global Initialization Parameters

The following local initialization parameters are also global initialization parameters. See Appendix A, *Global Initialization Parameters* for a description of these parameters:

- CTCA = NO | YES
- CTCA.TIMER = number of seconds
- DEBUG = nnnnnnnn
- ESTAE = YES | NO
- MAXBATCH = number of users
- MAXPRIMARY = number of primary sessions
- MAXPROCESS = number of executing PNODE and SNODE Processes
- MAXSECONDARY = number of secondary sessions
- MAX.TAPE = number of tape Processes
- QUIESCE = YES | NO
- QUIESCE.NODE = node name
- SECURITY.EXIT = (modname,DATASET | ALL,PSTKT) | OFF
  SECURITY = (modname,DATASET | ALL,PSTKT) | OFF
- SNA = YES | NO
- STAT.INIT = WARM | COLD
- STATISTICS.EXIT = modname | (modname[,MANAGER] | [,SERVER] | [,BOTH])
- TCP = OES | NO
- TCP.API.LISTEN = ((addr , port) , (addrn , portn))
- TCP.LISTEN = ((addr , port) , (addrn , portn))
- TCP.NAME = name of TCP/IP started task | NAME,FAIL
- TCP.PORTNUM = port number
- TCP.TIMER = wait time
- TCQ = WARM | COLD
- TRACE.BUFFER = nnn | 2
- UDT = YES | NO
- UDT33.LISTEN = ((ipaddr,port)[,(ipaddr,port)…])
- UPPER.CASE = YES|NO
- V2.BUFSIZE = (maximum buffer size for this transmission, TCP/IP send/receive buffer size)

# Glossary

## A

### ABEND

A task that ends prematurely, or abnormally, due to an error that cannot be resolved by recovery facilities while the task is executing.

### ACB

See *Access Method Control Block (ACB)*.

### Access Method

A technique for moving data between main storage and input/output (I/O) devices.

### Access Method Control Block (ACB)

A control block that links an application program to VSAM or VTAM.

### Adapter

A hardware card that allows a device, such as a PC, to communicate with another device, such as a monitor, a printer, or other I/O device. See also Channel-to-Channel Adapter.

### ADJACENT.NODE

An 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 Pool

Identifies the APPLIDs to be used for API communication with the DTF.

### Application Program Interface (API)

The Connect:Direct component that 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 APPLIDs.

### Asynchronous Processes

Processes that occur without a regular or scheduled time relationship. Unexpected or unpredictable with respect to the instructions of the program or to time. Contrast with synchronous.

### Attributes

Characteristics or properties that can be controlled, usually to obtain a required appearance; for example, the color of a line.

### Authorization File

Connect:Direct file used to control access to Connect:Direct and identify commands that can be executed by user ID. This file can also be used in conjunction with security exit interfaces to support the secured point-of-entry feature.

## B

### Batch Interface

An interface where non-interactive programs are executed. The environment schedules their execution independently of their submitter. Connect:Direct users issue batch commands using DMBATCH, a Connect:Direct-supplied program.

### Buffer

1. A portion of storage used to hold input or output data temporarily.
2. A routine or storage used to compensate for a difference in data rate or time of occurrence of events, when transferring data from one device to another.

### Buffer Pool

A set of buffers that contains buffers of the same length.

# C

## Central Processing Unit (CPU)

The part of a computer that includes the circuits that control the interpretation and execution of instructions.

## CF

See *Coupling Facility (CF)*.

## Channel

1. A functional unit, controlled by a z/OS server that handles the transfer of data between processor storage and local peripheral equipment.
2. A path along which signals can be sent.
3. The portion of a storage medium that is accessible to a given reading or writing station.
4. In broadband transmission, a designation of a frequency band 6 MHz wide.

## Channel-to-Channel (CTC)

Refers to the communication (transfer of data) between programs on opposite sides of a channel-to-channel adapter (CTCA). The CTCA for Connect:Direct for z/OS can be an ESCON CTC.

## Channel-to-Channel Adapter (CTCA)

A hardware device that can be used to connect two channels on the same computing system or on different systems. The CTCA for Connect:Direct for z/OS can be an ESCON CTC.

## Checkpoint/Restart

Eliminates the need to retransmit an entire file in the event of a transmission failure. A value in the COPY statement or CKPT initialization parameter specifies the checkpoint interval. If a copy procedure is interrupted, Connect:Direct will restart that copy at the last checkpoint.

## CICS

See *Customer Information Control System (CICS)*.

## Command Line Interface

Connect:Direct interface that allows users to submit Connect:Direct Processes and commands from their native command line environment.

## Commands

An instruction that directs a control unit or device to perform an operation or a set of operations. Connect:Direct users issue commands to initiate and monitor activity within the Connect:Direct system. Connect:Direct commands can be issued from the IUI, the operator console, a batch job, or a user application program.

## Component

1. Hardware or software that is part of a functional unit.
2. A functional part of an operating system; for example, the scheduler or the Hold queue.

## Compression

Storing data in a format that requires less space than usual. Data compression is particularly useful in communications because it enables devices to transmit the same amount of data in fewer bits. See also Variable Extended Compression.

## Configuration

The arrangement of a computer system or network as defined by the nature, number, and main characteristics of its functional units. More specifically, the term configuration may refer to a hardware or software configuration. See also System Configuration.

## Connect:Direct/Manager

The component of a Connect:Direct/Plex environment that handles the following functions:

- ✦ Interface connections
- ✦ Statistics file updates
- ✦ CKPT and TCQ/TCX file access
- ✦ TYPE file, AUTH file, NETMAP file, SECURE+ parameter file, and SECURE+ Digital Signature updates
- ✦ Workload balancing

## Connect:Direct/Plex

A Connect:Direct system consisting of a Connect:Direct/Manager and one or more Connect:Direct/Servers in a system complex or parallel system complex.

## Connect:Direct/Server

A Connect:Direct/Plex component that executes the Processes.

## Connect:Direct/Stand-alone Server

A Connect:Direct system that is not part of a Connect:Direct/Plex.

## Connectivity

A term used to describe the physical interconnections of multiple devices, computers, or networks employing similar or different technology and/or architecture together to accomplish effective communication between and among connected members. It involves data exchange and/or resource sharing.

## Console

A logical device that is used for communication between the user and the system.

## Coupling Facility (CF)

A special logical partition (LP) that provides high-speed caching, list processing, and locking functions in Parallel Sysplex.

## CPU

See *Central Processing Unit (CPU)*.

## Cross-System Coupling Facility (XCF)

A z/OS facility that allows multiple instances of the same application to communicate and share information with each other.

## CTC

See *Channel-to-Channel (CTC)*.

## CTCA

See *Channel-to-Channel Adapter (CTCA)*.

## Customer Application

An application that does customer-specific processing.

## Customer Information Control System (CICS)

An IBM-licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

# D

## Daemon

A background task, process, or thread that intermittently awakens to perform some task or function and then returns to an idle state.

## DASD

See *Direct Access Storage Device (DASD)*.

## Database

1. A set of data, or a part or the whole of another set of data, that consists of at least one file and is sufficient for a given purpose or for a given data-processing system.
2. A collection of data fundamental to a system. See also Database Control (DBCTL), data entry database (DEDB), data sharing, and data sharing group.

## Data Set

The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

## Data Sharing

In a Parallel Sysplex, the ability of concurrent subsystems or application programs to directly access and change the same data while maintaining data integrity.

## Data Transmission Facility (DTF)

The nucleus component of Connect:Direct. The DTF controls information distribution to other Connect:Direct nodes in the network. Startup parameters that govern the overall activity of the DTF are defined within the initialization parameters.

In a Connect:Direct/Plex, the DTF consists of a Connect:Direct Manager and one or more Connect:Direct Servers.

## Decipher

To convert enciphered data into clear data.

## Decrypt

To convert encrypted data into clear data.

### Default

Pertaining to an attribute, value, or option that is assumed when none is explicitly specified.

### Direct Access Storage Device (DASD)

A physical device, such as an IBM 3390, in which data can be permanently stored and subsequently retrieved using licensed products like IMS and DB2, or using IBM supported access methods like VSAM in operating system environments like z/OS.

### Directory

A list of files that are stored on a disk or diskette. A directory also contains information about the file, such as size and date of last change.

### DTF

See *Data Transmission Facility (DTF)*.

### Dynamic

Pertaining to an operation that occurs at the time it is needed rather than at a predetermined or fixed time.

## E

### Execution

The process by which a computer carries out the instruction or instructions of a computer program.

### Extended Submit Facility (ESF)

The facility that 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.

## F

### File System

The collection of files and file management structures on a physical or logical mass storage device such as a disk.

### Format

1. A specified arrangement of things, such as characters, fields, and lines, usually used for displays, printouts, or files.
2. To arrange things such as characters, fields, and lines.

### Function Management Header (FMH)

One or more headers, optionally present in the leading request units (RUs) of an RU chain, that allow one LU to (a) select a transaction program or device at the session partner and control the way in which the end-user data it sends is handled at the destination, (b) change the destination or the characteristics of the data during the session, and (c) transmit between session partners status or user information about the destination (for example, a program or device). Function management headers can be used with LU type 1, 4, and 6.2 protocols.

## H

### Hardware

The physical equipment as opposed to programs, procedures, rules, and associated documentation. Contrast with software.

### Host (computer)

1. In a computer network, a computer that provides end users with services such as computation and databases and that usually performs network control functions.
2. The primary or controlling computer in a multiple-computer installation.

## I

### ICO

See *InterConnect Option (ICO)*.

### Initialization

The preparation of a system, device, or program for operation. Connect:Direct initialization parameters specify alternate values for various parameters used during Connect:Direct start up.

## Input/Output (I/O)

1. Pertaining to a device whose parts can perform an input process and an output process at the same time.
2. Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process.

## Interactive

Pertaining to a program or system that alternately accepts input and then responds. An interactive system is conversational; that is, a continuous dialog exists between user and system. Contrast with batch.

## Interactive User Interface (IUI)

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.

## InterConnect Option (ICO)

A feature of Connect:Enterprise. This option provides an automatic, secure way to route application-produced distribution files from a Connect:Direct supported node to a Connect:Enterprise node for distribution, automatically distribute Connect:Enterprise batches to a Connect:Direct node upon arrival, and provide success or failure notification at each process step.

## Interface

A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

## Interrupt

1. A suspension of a process, such as execution of a computer program caused by an external event, and performed in such a way that the process can be resumed.
2. In data communication, to take an action at a receiving station that causes the sending station to end a transmission.
3. To temporarily stop a process.

## I/O

See *Input/Output (I/O)*.

### I/O Service Units

A measure of individual data set I/O activity and JES spool reads and writes for all data sets associated with an address space.

### IUI

See *Interactive User Interface (IUI)*.

## J

### Job Entry Subsystem (JES)

A system facility for spooling, job queuing, and managing job-related data.

## L

### LAN

See *Local Area Network (LAN)*.

### Link

The combination of physical media, protocols, and programming that connects devices.

### Load Module

A computer program in a form suitable for loading into storage for execution.

### Local Area Network (LAN)

A data network located on the user's premises in which serial transmission is used for direct data communication among data stations. It services a facility without the use of common carrier facilities.

### Local Cache

A buffer in local system storage that may contain copies of data entries in a CF cache structure.

### LOCAL.NODE

An 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.

### Logical Connection

In a network, devices that can communicate or work with one another because they share the same protocol.

### Logical Unit (LU)

In VTAM, the source and recipient of data transmissions. Data is transmitted from one logical unit (LU) to another LU. For example, a terminal can be an LU, or a CICS system can be an LU.

### Logically Partitioned (LPAR) Mode

A CPC power-on reset mode that enables use of the PR/SM (Processor Resource/Systems Manager) feature and allows an operator to allocate CPC hardware resources (including CPs, central storage, expanded storage, and channel paths) among logical partitions.

### LU

See *Logical Unit (LU)*.

## M

### Main Storage

A logical entity that represents the program addressable portion of central storage. All user programs are executed in main storage.

### Mainframe (z/OS server)

A large computer, in particular one to which other computers can be connected so that they can share facilities the z/OS server provides, for example, a z/OS computing system to which personal computers are attached so that they can upload and download programs and data.

### Memory

The program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing.

### Migration

Installing a new version or release of a program when an earlier version or release is already in place. See file migration.

### Modal Statements

Statements (IF THEN, EIF, ELSE, EXIT, and GOTO) in Connect:Direct that allow you to alter the sequence of Connect:Direct Process execution based on completion of a previous Process step.

### Module

A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or output from an assembler, compiler, linkage editor, or executive routine.

### Multiprocessing

The simultaneous execution of two or more computer programs or sequences of instructions. See also Parallel Processing.

## N

### NCP

See *Network Control Program (NCP)*.

### Netmap

See *Network Map*.

### Network

A configuration of data processing devices and software connected for information interchange.

### Network Control Program (NCP)

A program residing in a communication controller (for example, the IBM 3745 Communication Controller) that controls the operation of the communication controller.

### Network Map

The VSAM file that identifies 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

1. Any site in a network from which information distribution can be initiated.
2. In SNA, an endpoint of a link or junction common to two or more links in a network. Nodes can be distributed to Z/OS server processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities.

# O

## Offline

Not controlled directly by, or not communicating with, a computer. Contrast with online.

## Online

Pertaining to equipment, devices, or data under the direct control of the processor. Contrast with offline.

## Online Messages

The completion and error messages that are displayed online.

## Operating System (OS)

The software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible. Examples are z/OS, VSE/ESA, and VM/ESA.

## Operator Interface

Allows Connect:Direct commands to be issued from the operator console. This interface also allows tailoring of Connect:Direct commands through a command list (CLIST) facility.

## OS

See *Operating System (OS)*.

# P

## Parallel

1. Pertaining to a process in which all events occur within the same interval of time, each handled by a separate but similar functional unit; for example, the parallel transmission of the bits of a computer word along the lines of an internal bus.
2. Pertaining to the concurrent or simultaneous operation of two or more devices or to the concurrent performance of two or more activities in a single device.
3. Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels.
4. Pertaining to the simultaneity of two or more processes.

5. Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts.

## Parallel Processing

The simultaneous processing of units of work by many servers. The units of work can be either transactions or subdivisions of large units of work (batch).

## Parallel Sessions

The 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.

## Parallel Sysplex

A sysplex with one or more coupling facilities.

## Partitioned Data Set (PDS)

A data set in DASD storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

## Partitioned Data Set Extended (PDSE)

A data set organization that stores many different but related members. A PDSE contains an indexed directory and members that are similar to the directory and members of a partitioned data set (PDS) but offers architectural advantages that a PDS cannot. For example, space can be dynamically allocated and reclaimed using PDSEs, exploiting space more efficiently.

## Primary Node (PNODE)

The 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

The statements that 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).

### Processing Unit

The part of the system that does the processing, and contains processor storage.

### Processor

A processing unit, capable of executing instructions when combined with main storage and channels.

### Protocol

A specification of the format and relative timing of information exchanged between peer entities within a layer.

## R

### Record

A set of data treated as a unit.

### Recovery

To maintain or regain system operation after a failure occurs. Generally, to recover from a failure is to identify the failed hardware, to de-configure the failed hardware, and to continue or restart processing.

### Remote Node

The Connect:Direct node that interacts with the local node during Process execution. The remote node is also referred to as the participating, receiving, target, destination, or secondary node (SNODE).

### Resource Access Control Facility (RACF)

The facility that provides access control by identifying and verifying users to the system. RACF authorizes access to resources, logs unauthorized access attempts, and logs accesses to protected data sets.

### Response Time

The amount of time it takes after a user presses the enter key at the terminal until the reply appears at the terminal.

### Retry Interval

An installation parameter that specifies the interval, in minutes, that the retries mentioned in the Max Retries parameter will be performed.

### Routing

The assignment of a path by which a transfer reaches its destination.

## S

### Secondary Node (SNODE)

The Connect:Direct node that interacts with the primary node (PNODE) during process execution. The secondary node (SNODE) can also be referred to as the participating, target, or destination node. Every Process has one PNODE and one SNODE.

### Session

1. The entity through which a Connect:Direct PNODE initiates and executes one or more Processes to one or more SNODEs.
2. The entity through which a Connect:Direct SNODE receives one or more Processes.
3. In SNA, a logical connection between two network addressable units that can be activated, tailored to provide various protocols, and deactivated as requested.
4. The data transport connection resulting from a call or link between two devices.
5. The period of time during which a user of a node can communicate with an interactive system; usually it is the elapsed time between logon and logoff.
6. In network architecture, an association of facilities necessary for establishing, maintaining, and releasing connections for communication between stations.

### Session Classes

The installation parameter that specifies the Process class groupings, priorities and number of Processes that can be concurrently executed on this Connect:Direct node.

### Shared

Pertaining to the availability of a resource to more than one use at the same time.

### SNA

See *Systems Network Architecture (SNA)*.

### SNODE

See *Secondary Node (SNODE)*.

### Standby Connect:Direct Manager

In an extended recovery environment, the backup Connect:Direct Manager that takes over work from the active Connect:Direct Manager when the active Connect:Direct Manager fails.

### Standby Connect:Direct Server

In an extended recovery environment, the backup Connect:Direct Server that takes over work from the active Connect:Direct/Server when the active Connect:Direct Server fails.

### Standby Connect:Direct System

In an extended recovery environment, the backup Connect:Direct system that takes over work from the active Connect:Direct system when the active system fails.

### Statistics Facility

The Connect:Direct facility that records all Connect:Direct activities.

### Statistics Files

A pair of VSAM data sets that hold Connect:Direct statistics records to document the history of a Connect:Direct Process.

### Storage

A unit into which recorded data can be entered, in which it can be retained and processed, and from which it can be retrieved.

### Subsystem

A secondary or subordinate system, or programming support, that is usually capable of operating independently of or asynchronously with a controlling system.

### SYMBOL Statement

The Connect:Direct Process statement that allows you to build symbolic substitution values.

## Symbolics

The parameters that allow 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.

## Synchronous

1. Pertaining to two or more processes that depend on the occurrences of a specific event such as common timing signal.
2. Occurring with a regular or predictable timing relationship.

## Sysplex

A set of z/OS systems communicating with each other through certain multisystem hardware components and software services to process workloads.

## Sysplex Data Sharing

The ability of multiple z/OS subsystems to share data across multiple system images. Sysplex data sharing differs from two-way data sharing in that the latter allows sharing across only two system images.

## System

In data processing, a collection of people, machines, and methods organized to accomplish a set of specific functions.

## System Configuration

A process that specifies the devices and programs that form a particular data processing system.

## Systems Network Architecture (SNA)

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.

The SNA network consists of network addressable units (NAUs), boundary function components, and the path control network.

# T

## TCQ

See *Transmission Control Queue (TCQ)*.

## TCP/IP

See *Transmission Control Protocol/Internet Protocol (TCP/IP)*.

## TDQ

See *Transient Data Queue (TDQ)*.

## Terminal

A device that is capable of sending and receiving information over a link; it is usually equipped with a keyboard and some kind of display, such as a screen or a printer.

## Throughput

1. A measure of the amount of work performed by a computer system over a given period of time, for example, number of jobs per day.
2. A measure of the amount of information transmitted over a network in a given period of time.

## Transient Data Queue (TDQ)

A CICS temporary storage queue in which event data is stored so that a client application can retrieve the information.

## Transmission Control Protocol/Internet Protocol (TCP/IP)

A set of public domain networking protocol standards that specify the details of how computers communicate, as well as a set of conventions for interconnecting networks and routing traffic.

## Transmission Control Queue (TCQ)

A VSAM relative record data set (RRDS) used to hold all Processes that have been submitted to Connect:Direct.

# U

## UDT

See *UDP (User Datagram Protocol) Data Transfer.*

## UDP (User Datagram Protocol) Data Transfer

An alternative transport layer protocol to TCP designed for a high-latency, high-bandwidth connection to achieve higher data transfer rates than TCP over this type of connection.

# V

## Version 1/Version 2 Flows

Version 1 and Version 2 Flows pertain to the internal Connect:Direct architecture and the way the Connect:Direct Function Management Header (FMH) protocol is performed between the source and destination nodes. V1 flows were the original architectural design of Connect:Direct when it was used for VTAM SNA LU0 mainframe-to-mainframe data transfers. As the Connect:Direct product has matured and TCP/IP and LU6.2 protocols were adopted, continued usage of the original V1 FMH protocols was no longer practical.

The Version 2 FMH protocol introduced in the early 1990's changed the structure of the FMH protocol from DSECT-mapped to keyword (XDR) fields. In addition, Version 2 Buffer Headers and a new more efficient method of Checkpointing were included. The flexibility of the Version 2 architecture has enabled several other Connect:Direct product enhancements to occur, such as support for ZLIB compression, Secure+ Encryption, and CRC checking.

SNA LU0 and SNUF LU0 (for the OS/400) still use Version 1 FMH protocols, whereas TCP/IP, UDT, SNA LU6.2, and CTCA use Version 2 FMH protocols.

## Virtual Storage (VS)

The storage space regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. Virtual storage size is limited by the address scheme of the computer system and the amount of auxiliary storage available, rather than the actual number of main storage locations.

## Virtual Storage Access Method (VSAM)

An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

### Virtual Telecommunication Access Method (VTAM)

The program that provides for workstation and network control. It is the basis of a System Network Architecture (SNA) network. It supports SNA and certain non-SNA terminals. VTAM supports the concurrent execution of multiple telecommunications applications and controls communication among devices in both single processor and multiple processor networks.

## W

### Wide Area Network (WAN)

A network that provides communication services to a geographic area larger than that served by a local area network.

## X

### XCF

See *Cross-System Coupling Facility (XCF)*.

# Index

## A



## B

# C

## D

# E



# F

# G



# I

## J



## L



## M

# N



# O



# P

# Q



# R



# S

## T

# U

# V



# W



# X