Connect:Direct®
HP NonStop

Management Programming Guide

Version 3.5

Sterling Commerce
An IBM Company

Connect:Direct HP NonStop Management Programming Guide
Version 3.5
First Edition

Sterling Commerce, Inc.

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

Contents

Preface

The *Connect:Direct HP NonStop Management Programming Guide* is for programmers writing applications to interface with Connect:Direct HP NonStop.

Read the first four chapters to learn how to write and run an API to use with Connect:Direct HP NonStop. These chapters introduce Connect:Direct HP NonStop and the components of the API environment. Chapter 5, *Sample Code*, provides a sample API written in C. If you plan to retrieve event messages, refer to Chapter 6, *Using DSM/EMS Event Reporting*, to integrate Connect:Direct HP NonStop in a Distributed Systems Management (DSM) environment.

This guide assumes knowledge of the Connect:Direct HP NonStop operating system, its applications, network, and environment. If you are not familiar with the Connect:Direct HP NonStop operating system, refer to the HP NonStop library of manuals.

Chapter Overview

The organization of the *Connect:Direct HP NonStop Management Programming Guide* follows:

- Chapter 1, *About Connect:Direct HP NonStop*, provides general information about the product and describes how Connect:Direct HP NonStop works.
- Chapter 2, *Using Application Programming Interfaces*, provides an overview of an Application Programming Interface (API) and describes the internal data structures used by Connect:Direct HP NonStop
- Chapter 3, *Programming the API*, provides the following information you need to program and run an API:
- Chapter 4, *Interface for User-Written Programs*, describes the two types of I/O exits and provides information about determining which I/O exit to define.
- Chapter 5, *Sample Code*, provides an example to illustrate concepts and considerations that are useful when you write an API.
- Chapter 6, *Using DSM/EMS Event Reporting*, describes how to use Event Management Service (EMS) in the HP NonStop Distributed Systems Management (DSM) environment.
- *Glossary*, defines Connect:Direct HP NonStop terms used in this manual.

Chapter 1

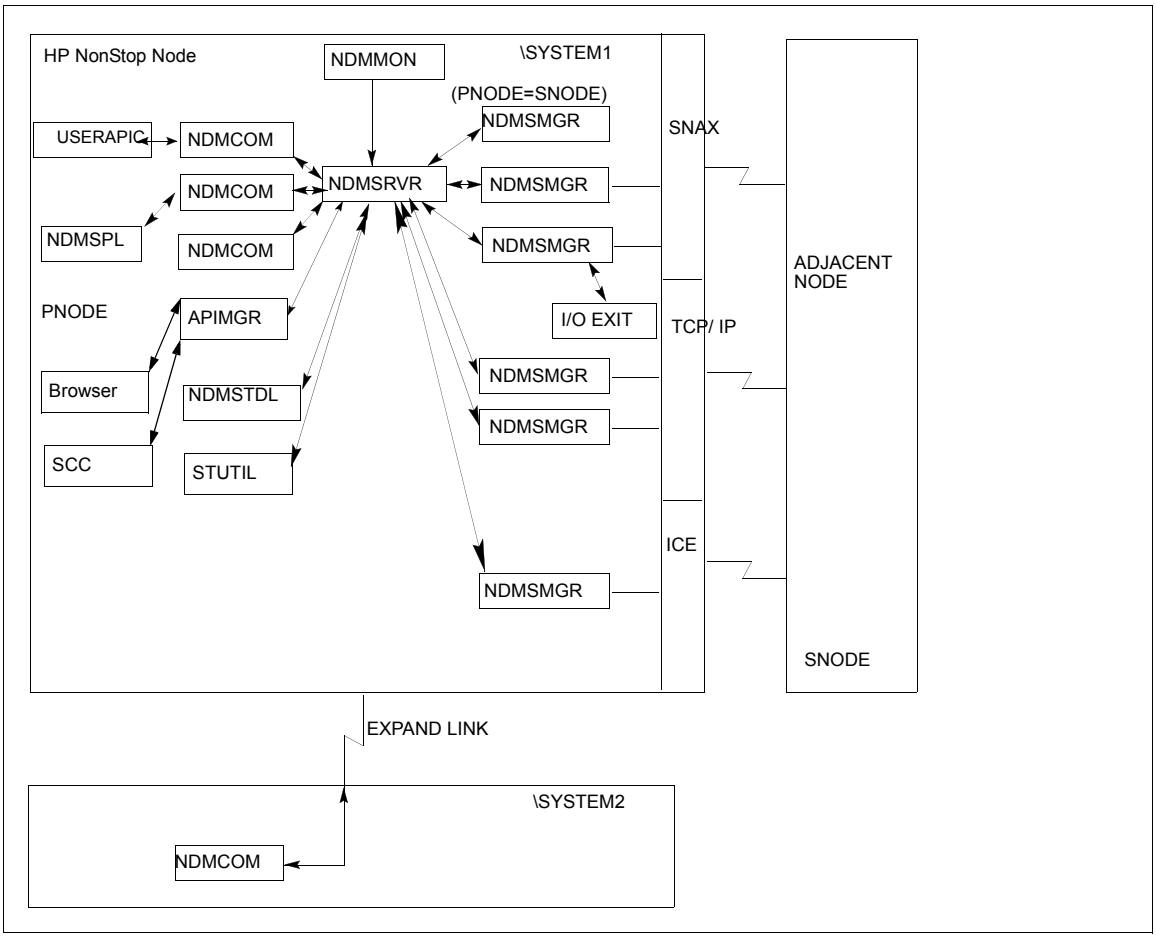
About Connect:Direct HP NonStop

Connect:Direct HP NonStop links technologies and moves all types of information between networked systems and computers. It manages high-performance transfers by providing features such as automation, reliability, efficient use of resources, application integration, and ease of use. Connect:Direct HP NonStop software offers choices in communications protocols, hardware platforms, and operating systems. It provides the flexibility to move information among mainframes, midrange systems, desktop systems, and LAN-based workstations.

Connect:Direct HP NonStop Components

Connect:Direct HP NonStop runs as an application on the Guardian operating system. The product components interact to execute the Process statements and commands submitted through the user interface.

The following figure illustrates the basic components of Connect:Direct HP NonStop: Monitor (NDMMON), Server (NDMSRVR), User Interface (NDMCOM), Session Manager (NDMSMGR), API Manager (APIMGR), I/O Exits (I/O EXIT), Statistics Deletion Program (NDMSTDL), Statistics Utility Program (STUTIL), Application Programming Interface (USERAPIC), and Connect:Direct HP NonStop Spooler Option (NDMSPL). Brief descriptions of each component follow the sample network configuration.



Component	Description
Monitor	The monitor (NDMMON) is a nonstop process that creates and monitors the Connect:Direct HP NonStop server (NDMSRVR) process. For NDMMON startup instructions, refer to the *Connect:Direct HP NonStop Installation Guide*.
Server	The Connect:Direct HP NonStop server (NDMSRVR) process manages: - Command requests - Communication with the session manager - Session establishment requests for TCP/IP **Note**: If the NDMSRVR process ends abnormally or the CPU executing the NDMSRVR process fails, NDMMON creates a new NDMSRVR process with the original NDMSRVR process name and parameters.
User Interface	NDMCOM is the user interface with NDMSRVR. Use NDMCOM, the command-line interface, to issue Connect:Direct HP NonStop commands and to change, configure, and display the Connect:Direct HP NonStop environment.

Component	Description
Session Manager	The Connect:Direct HP NonStop session manager (NDMSMGR) module establishes sessions and transfers data between the local and adjacent nodes. The application can be configured to start session managers at initialization, or you can start them manually using the MODIFY command. If you define dynamic LUs for TCP/IP connectivity, NDMSRVR starts session managers as needed. You cannot issue the MODIFY command to start dynamic session managers. The figure on the previous page shows six session managers, two of which are communicating across SNAX sessions, two across TCP/IP, and one across ICE. One session manager is using the PNODE=SNODE facility.
API Manager	The Connect:Direct HP NonStop API Manager (APIMGR) module provides an interface for browser(s) and Sterling Control Center. To define an API Manager, see the instructions on the INSERT NETMAP AMGR command in the *Connect:Direct HP NonStop Administration Guide*.
I/O Exits	I/O exit support enables the user-written programs to serve as application interfaces for Connect:Direct HP NonStop data transfers. I/O exits permit manipulation of data formats and database architectures not currently supported by Connect:Direct HP NonStop. For transfers (COPY), Connect:Direct HP NonStop supports direct access only to Enscribe and Spool files. I/O exit support enables user-written programs to access non-supported databases, such as SQL, and manipulate data during a COPY step.
Statistics Deletion Program	The statistics deletion program (NDMSTDL) ensures that sufficient space is available to write statistics records in the statistics files. NDMSTDL deletes records from the Connect:Direct HP NonStop statistics file based on user-specified deletion criteria and maximum percentage of file capacity. For instructions on using NDMSTDL, refer to *Optimizing Performance* in the *Connect:Direct HP NonStop Administration Guide*.
Statistics Utility Program	The statistics utility program (STUTIL) analyzes the statistics files to determine how much space is available. Connect:Direct HP NonStop returns this information to the server for determination on when to run NDMSTDL.
Application Program Interface	An Application Program Interface (API) is a user-written application that communicates with NDMCOM. Refer to the *Connect:Direct HP NonStop Management Programming Guide* for details on creating and using an API. **NOTE**: The sample program, USERAPIC, is provided as a template for writing a customized programmatic application.
Connect:Direct HP NonStop Spooler Option	The Connect:Direct HP NonStop Spooler option permits an installation to transfer spooler jobs automatically from an HP NonStop node to a file on an adjacent node. For the information you need to install, configure, and run the Connect:Direct HP NonStop Spooler option, refer to Connect:Direct HP NonStop *Spooler Option* in the *Connect:Direct HP NonStop Administration Guide*.

Automated Installation and Management System

The Automated Installation and Management System (AIMS) is a full-screen, block-mode interface for installing, configuring, and starting Connect:Direct HP NonStop.

Note: You must have a 6530 terminal or a 6530 emulation program to run AIMS.

AIMS is a menu-driven system that collects information about your node and the nodes you are communicating with and guides you through the installation. Performing the menu options in the displayed numerical order expedites installation. Each user-input screen has a Help feature, which describes the entry fields for the screen. Throughout the AIMS procedure, messages displayed on the bottom line of the screen inform you of the status of the procedure and indicate errors. For more information on AIMS, refer to *Connect:Direct HP NonStop Installation Guide*.

Connect:Direct HP NonStop Concepts

This section introduces certain concepts and definitions important to understanding user operations.

Processes

The Process language provides instructions for transferring files, running programs, submitting jobs on the adjacent node, and altering the sequence of Process step execution. You can include one or more steps in a Process.

A Process consists of a Process definition statement (PROCESS statement) and one or more additional statements. Parameters further qualify Process instructions. For more information, including sample Processes, see the Processes web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.

Transmission Control Queue

The Transmission Control Queue (TCQ) controls Process execution as Connect:Direct HP NonStop operates. Connect:Direct HP NonStop stores submitted Processes in the TCQ which is divided into logical queues.

Connect:Direct HP NonStop places the Process in the appropriate queue based on Process statement parameters that affect scheduling. Examples of such parameters are the HOLD, RETAIN, and STARTT parameters.

Connect:Direct HP NonStop selects Processes in a first-in first-out manner for execution in Process class and priority as sessions are available. You can access the queues and manipulate the Processes through Connect:Direct HP NonStop commands.

Refer to *Queuing Processes* in the *Connect:Direct HP NonStop User Guide and Reference* for a discussion of the following topics:

- Understanding the Transmission Control Queue
- Managing Processes in the TCQ
- Scheduling Connect:Direct HP NonStop Activity

Network Map

The network map file defines the nodes with which Connect:Direct HP NonStop can communicate. The network map includes a local node record and one or more adjacent nodes, logical units (LUs), API managers (AMGRs), and logmode records.

The local node is the logical name for the node on which you installed Connect:Direct HP NonStop. An adjacent node is the node definition for a remote site. LUs provide communication between the HP NonStop

system (local node) and adjacent nodes. Logmode records define the session protocol for an SNA HP NonStop LU, and are only used when the local LU is configured as the primary LU (PLU).

In addition to creating explicit adjacent node records for the individual nodes with which you communicate, you can also define domain node adjacent node records for communications with large networks of Connect:Direct nodes, including Connect:Direct/Plex systems, in a TCP domain. These special-purpose adjacent node records simplify your network map and increase efficiency.

Defining Domain Nodes to Manage Inbound TCP/IP Connections

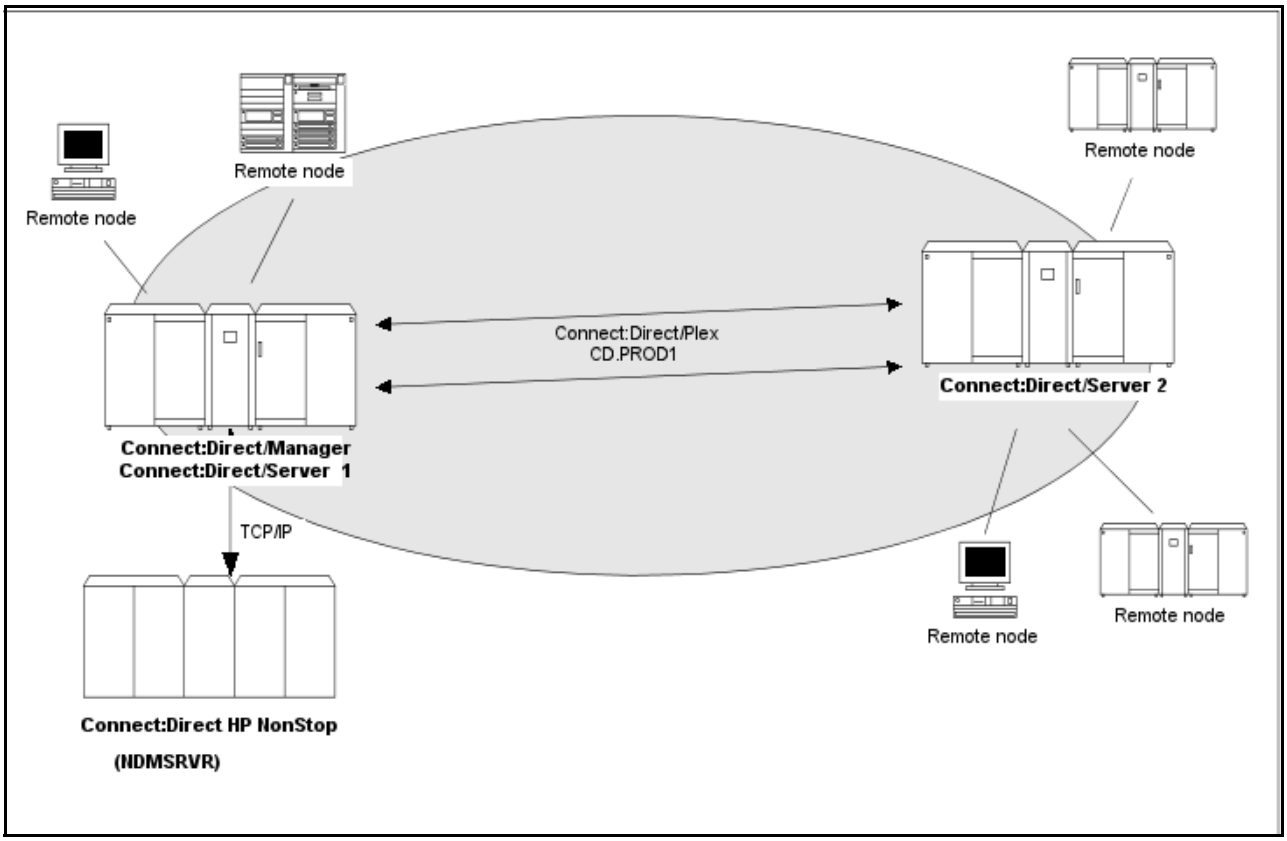
The domain node feature enables you to manage inbound connection requests to the Connect:Direct HP NonStop node from IP addresses that are not explicitly configured in the network map, for example from multiple Connect:Direct/Server processes under the direction of the Connect:Direct/Plex Manager. Using the domain node feature, you can create an adjacent node entry of the type NDM.DOMAIN for any TCP/IP domain containing one or more Connect:Direct nodes and define a range of IP addresses instead of defining an adjacent node record for each remote connection. When the Connect:Direct HP NonStop server receives a connection request, it first attempts to match the originating IP address with a specific adjacent node entry in the network map. If this search fails, the server searches for adjacent nodes of the type NDM.DOMAIN and then uses the IPMASK parameter as a template to identify a node that best fits the mask's pattern. Without a domain node record, each Connect:Direct/Plex Server or remote node must have an adjacent node record in the Connect:Direct HP NonStop network map to initiate connections to the local Connect:Direct HP NonStop node.

The DOMAINSERVER and the NETMAPCHECK initialization parameters are associated with this feature.You must set the DOMAINSERVER global initialization parameter to Yes before you can define a domain node.

You can use the NETMAPCHECK initialization parameter and Connect:Direct Secure+ Option to secure the TCP/IP sessions. See *Connect:Direct Secure+ Option* in this chapter for more information about Connect:Direct Secure+ Option and *Planning the Installation* in the *Connect:Direct HP NonStop Installation Guide* for a discussion about how the security options function in your environment.

Using Session Redirection for Outbound TCP/IP Connections

Connect:Direct HP NonStop supports session redirection for outbound connections to a Connect:Direct/Plex system. As illustrated in the following figure, a Connect:Direct/Plex system is a Connect:Direct for OS/390 (zOS) system consisting of a Connect:Direct/Plex Manager and one or more Connect:Direct/Servers in a TCP/IP environment. Connection requests from the Connect:Direct HP NonStop node to the Connect:Direct/Plex system are routed to the Connect:Direct/Plex Manager, which redirects the connection request to the appropriate, available Connect:Direct/Plex Server process. Redirecting communications sessions across multiple Connect:Direct Server processes simplifies the network map, facilitates load-balancing, and ensures continuous, efficient use of resources.



You can create adjacent node records either through AIMS or with individual network map commands. Use the following table as a guide to the tools and the parameters used to create adjacent node records:

Task	Reference
Planning your network map to use domain nodes and session redirection	*Defining Adjacent Node Records for TCP/IP Connections* in *Planning the Installation* in the *Connect:Direct HP NonStop Installation Guide*
Setting the DOMAINSERVER and NETMAPCHECK initialization parameters	*Setting Initialization Parameters* in *Installing and Configuring Connect:Direct HP NonStop* in the *Connect:Direct HP NonStop Installation Guide*
Assessing your security options	*Defining Adjacent Node Records for TCP/IP Connections* in *Planning the Installation* in the *Connect:Direct HP NonStop Installation Guide*
Creating the worksheets for your adjacent node records in the network map	*Preparing to Define the Network Map through AIMS* in *Planning the Installation* in the *Connect:Direct HP NonStop Installation Guide*
Defining the network map through AIMS	*Configuring the Network Map* in *Installing and Configuring Connect:Direct HP NonStop* in the *Connect:Direct HP NonStop Installation Guide*
Using individual commands, syntax, and parameters to define and maintain the network map	*Defining and Maintaining the Network Map* in the *Connect:Direct HP NonStop Administration Guide*

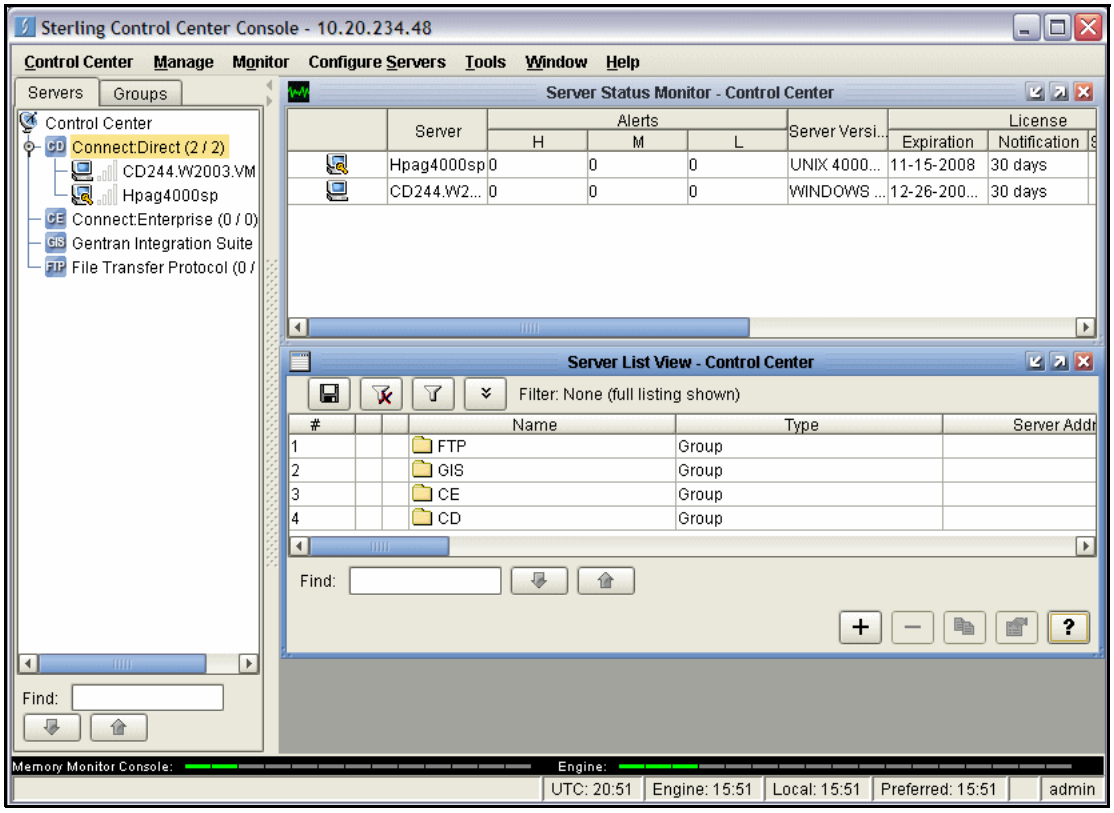
Connect:Direct Secure+ Option

The certificate authentication and multiple cipher suites offered by Connect:Direct Secure+ Option provide the confidence that your organization can use public networks knowing that data is being reliably transferred from a known source and can only be read by the intended recipient. To use Connect:Direct Secure+ Option for communications with remote nodes, you must have node records in the Connect:Direct Secure+ Option parameters file (SPNODES) that duplicate the adjacent node records in the Connect:Direct HP NonStop network map.You can populate the Connect:Direct Secure+ Option parameters file from entries defined in an existing network map using the Sync with NetMap function. For more information about populating the Connect:Direct Secure+ Option parameters file (SPNODES) and configuring nodes for Connect:Direct Secure+ Option, refer to the *Connect:Direct Secure+ Option HP NonStop Implementation Guide*. For information about using Connect:Direct Secure+ Option with domain nodes, see *Planning the Installation* in the *Connect:Direct HP NonStop Installation Guide.*

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 e-mail

❖ 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.

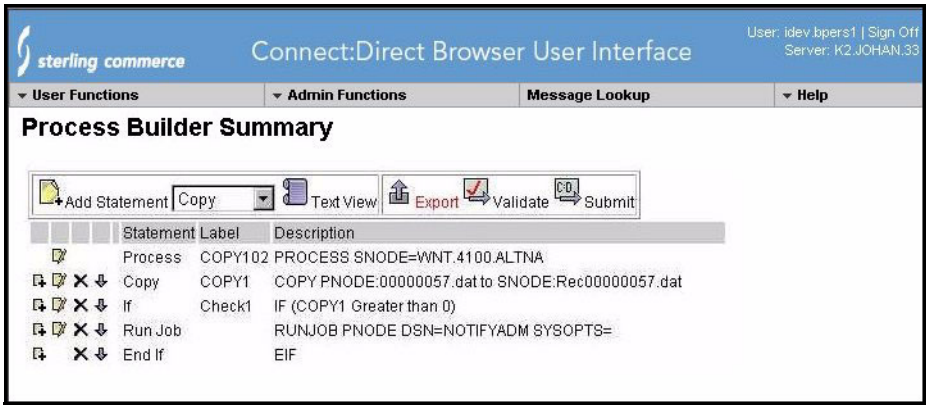
Connect:Direct Browser User Interface

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 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.

Interfacing with Sterling Control Center or Browser User Interface

Connect:Direct HP NonStop can interface with Sterling Control Center and Connect:Direct Browser User Interface. The TCP/IP API enables users of these other Sterling Commerce applications to configure, control, and operate Connect:Direct HP NonStop from any host on a TCP/IP network. To set up a connection between Connect:Direct HP NonStop and Sterling Control Center or Browser User Interface, you need to define two entities in the network map:

- An adjacent node with the TYPE parameter defined as NDM.API and the IPADDR parameter defined as the address of the external application client from which connection requests may be received.
- An API manager (AMGR) to handle communications sessions with the external application. The AMGR record is used to define the local TCP process and port number on which a LISTEN is to be posted to accept incoming connection requests.

After you have defined these components, you must identify the AMGRs you want to use to communicate with an adjacent node by using the RELATE NETMAP command. For more information on both the INSERT and RELATE NETMAP commands, refer to *Defining and Maintaining the Network Map* in the *Connect:Direct HP NonStop Administration Guide*. You can also perform these functions using the Automated Installation & Management System (AIMS) to set up the network map. For more information, refer to *Installing and Configuring* Connect:Direct HP NonStop in the *Connect:Direct HP NonStop Installation Guide*.

Commands

You use commands to submit Connect:Direct HP NonStop Processes to the TCQ and to manipulate Processes in the queue by flushing, deleting, or suspending them.

The following command submits the Process called ONESTEP to the TCQ with a HOLD status of Yes:

```
SUBMIT FILE ONESTEP HOLD=YES
```

Other commands allow you to select and display statistics or perform administrative functions, such as maintain network maps, user authorities, and default types.

The command language consists of the following types of commands:

- User
- Administrator

- Environment
- Message

User and Administrator Commands

Issue user and administrator commands to perform the following tasks:

- Submit Connect:Direct HP NonStop Processes
- Monitor and control Process execution
- Perform administrative functions
- Examine Connect:Direct HP NonStop node definitions
- Display and update initialization parameters
- Stop Connect:Direct HP NonStop

Refer to the *Connect:Direct HP NonStop User Guide and Reference* for command syntax and parameter descriptions for user commands. Command syntax and parameter descriptions for administrator commands are in the *Connect:Direct HP NonStop Administration Guide*.

The following table lists the user and administrator commands and their functions:

Command	Function
CHANGE PROCESS	Modifies a Process in the TCQ.
DELETE PROCESS	Removes a nonexecuting Process from the TCQ.
DELETE NETMAP	Removes a node, LOGMODE, LU, AMGR, or relation entry from the network map.
DELETE SECURITY†	Removes a user record from the security file.
DELETE TYPE†	Removes a type record from the type file.
DELETE USER†	Removes a user record from the authorization file.
DISPLAY LICENSE	Displays current license key.
DISPLAY LOGGING	Displays or prints the settings for EMS and STATS, and the name of the collector process.
DISPLAY PARMS	Displays or prints current settings of the initialization parameters (from the NDMINIT file).
DISPLAY SESSIONS	Displays active and licensed session counts.
FLUSH PROCESS	Removes an executing Process from the TCQ.
INSERT NETMAP†	Adds a local node, an adjacent node, an LU, a LOGMODE, or an AMGR record to the network map.
INSERT NETMAP AMGR†	Adds an API manager, AMGR, to the network map, which enables connections to Sterling Control Center and the Browser User Inteface.
INSERT SECURITY†	Adds a security record to the security file for Secure Point of Entry.
INSERT TYPE†	Adds a type record to the type file.
INSERT USER†	Adds a user record to the authorization file.
LASTPNUMBER	Determines the number of the last Process submitted in the current NDMCOM session.
MODIFY†	Starts the trace facility and/or modifies other operational functions.

† Administrative commands

Command	Function
RELATE NETMAP†	Assigns specific LUs or AMGRs to an adjacent node record.
SELECT NETMAP	Displays or prints definitions of API Manager, node, LOGMODE, and LU entries in the network map file.
SELECT PROCESS	Displays or prints information about a Process in the TCQ.
SELECT SECURITY	Displays or prints records in the security file.
SELECT STATISTICS	Displays or prints statistics, messages, license information, and /or commands in the statistics log.
SELECT TYPE	Displays or prints type records.
SELECT USER	Displays or prints user records in the authorization file.
STATUS	Displays the status of nodes and LUs, Processes in queues, and TCP listen ports.
STOP ALL†	Stops Connect:Direct HP NonStop operation.
SUBMIT	Submits a Process for execution.
SUSPEND PROCESS	Suspends an executing Process.
UPDATE LICENSE	Validates the license key in the LICENSE file and updates the active license.
UPDATE LOGGING†	Modifies settings for EMS, STATS, and COLLECTOR.
UPDATE NETMAP†	Modifies settings for an API manager, node, LOGMODE, or LU record in the network map.
UPDATE PARM†	Alters operating parameters from the NDMINIT file. For more information on this command, see Connect:Direct HP NonStop *Installation Guide*.
UPDATE SECURITY†	Changes a security record in the security file.
UPDATE STATISTICS†	Dynamically changes the percentage setting, deletion criteria, and midnight housekeeping flag in the statistics facility (NDMSTDL).
UPDATE TYPE†	Changes a type record in the type file.
UPDATE USER†	Changes a user record in the authorization file.

† Administrative commands

Environment Commands

Use environment commands to change and define the Connect:Direct HP NonStop environment or to facilitate the use of NDMCOM. The following table lists the environment commands and their functions:

Command†	Function
!	Reexecutes a previous command line, without modifications.
DISPLAY STATINFO	Displays percentage setting, deletion criteria, midnight flag setting, last execution of NDMSTDL, and file information for the statistics files (STATFILE, STATSRCH, STATSRC0).
EDIT	Invokes the HP NonStop TEDIT editor.
ENVIRONMENT	Displays the current Connect:Direct HP NonStop environment, including defaults.

† Refer to the *Controlling the Environment* chapter in the *Connect:Direct HP NonStop User Guide and Reference* for command syntax and parameter descriptions for environment commands.

Command†	Function
EXIT	Exits NDMCOM.
FC	Changes and/or reissues previously typed commands.
HELP	Accesses the interactive Connect:Direct HP NonStop Help facility.
HISTORY	Displays up to the last 100 commands issued.
LIST	Displays the contents of an edit file.
LOGON	Logs on to NDMCOM.
OBEY	Executes a series of HP NonStop and Connect:Direct HP NonStop commands, except FC, contained in an edit file.
OBEYVOLUME	Defines the default volume used for expansion of the obey file name.
OPEN	Opens the NDMSRVR process.
OUT	Changes the default output file.
PRINTER	Defines the print file name.
PROCVOLUME	Defines the default volume used for expansion of the Process file name.
RUN	Executes any user-written or system programs without exiting NDMCOM.
SYMBOL	Builds, deletes, and displays symbolic substitution values for use in Connect:Direct HP NonStop.
TIME	Retrieves the current day, date, and time.
VERSION	Displays or prints the version, release, and maintenance level for the NDMCOM currently running.
VOLUME	Defines the current default volume.

† Refer to the *Controlling the Environment* chapter in the *Connect:Direct HP NonStop User Guide and Reference* for command syntax and parameter descriptions for environment commands.

Message Commands

Use Connect:Direct HP NonStop message commands to insert, delete, display, modify, and print messages. Refer to *Using* Connect:Direct HP NonStop in the *Connect:Direct HP NonStop User Guide and Reference* for syntax and parameter descriptions for displaying and printing messages. Refer to *Modifying the Message File* in the *Connect:Direct HP NonStop Administration Guide* for syntax and parameter descriptions for modifying messages.

Flow of Connect:Direct HP NonStop Operations

The following shows the processing flow for a SUBMIT command.

Note: In this discussion, the Browser User Interface can be substituted wherever you see the command line interface, NDMCOM.

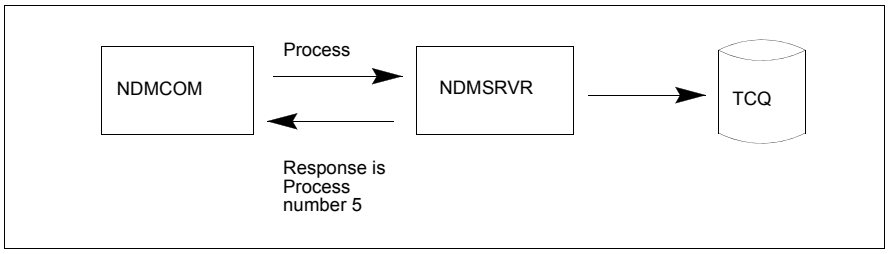
- The SUBMIT command is issued through NDMCOM.

```
CD.49.>SUBMIT FILE $VOL.SEND.FILE
```

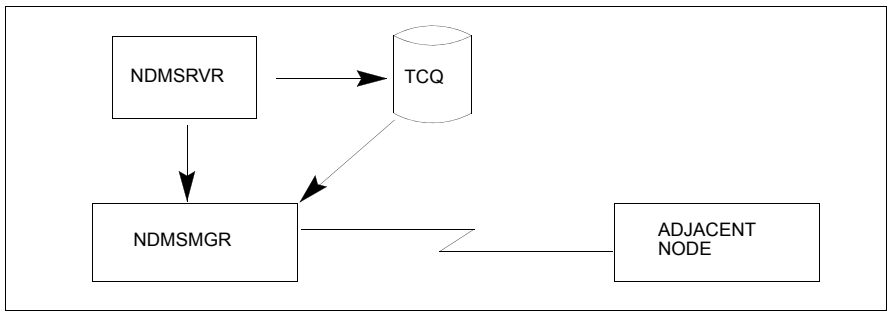
- The command submits the file, $VOL.SEND.FILE. The file contains Process statements.

```
SEND     PROCESS  SNODE=MVS.NODE
STEP01   COPY     FROM (DSN=$SYS.TAN.TXT)-
                  TO   (DSN=MVS.FILE SNODE)
```

- The Process is sent to the server. The server then places the Process on the TCQ, responds to NDMCOM with the Process number (PNUMBER), and routes the Process to an available session manager. In the following figure, the server returns a PNUMBER of 5 to NDMCOM.



- The session manager reads the Process from the TCQ and executes it.



- While the Process is queued, or during execution, you can display Process status by issuing the SELECT PROCESS command.

```
CD.50.>SELECT PROCESS PNUMBER=5
```

Refer to *Managing Processes* in the *Connect:Direct HP NonStop User Guide and Reference* for sample output from the SELECT PROCESS command.

- After Process execution, you can display the results of the operation by issuing the SELECT STATISTICS command. Refer to *Viewing System Files* in the *Connect:Direct HP NonStop User Guide and Reference* for sample output from the SELECT STATISTICS command.

```
CD.51.>SELECT STATISTICS PNUMBER=5
```

Chapter 2

Using Application Programming Interfaces

This chapter provides an overview of an Application Programming Interface (API) and describes the internal data structures used by Connect:Direct HP NonStop.

API Overview

An API is a user-written application that interfaces with Connect:Direct HP NonStop interface, NDMCOM.

Initially, you start the API as a process. The API is then responsible for the following tasks:

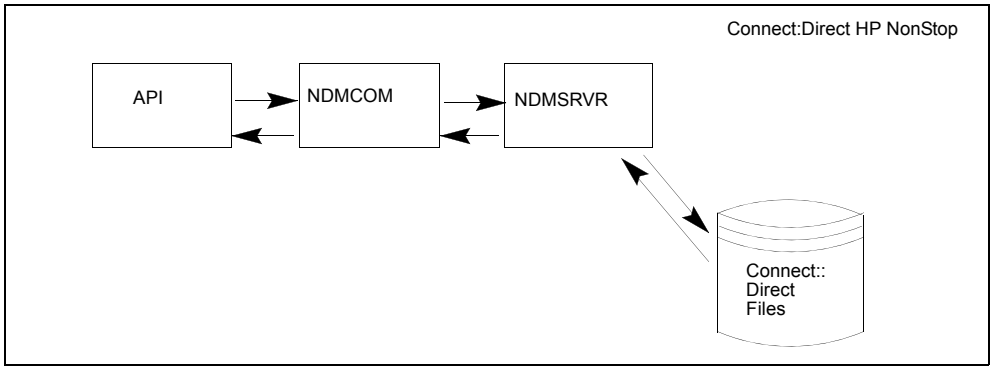
- Building the parameters and startup messages for NDMCOM
- Creating and opening NDMCOM
- Passing the parameters and startup messages to NDMCOM

NDMCOM then opens the API, sends the version, issues the OPEN command to the server (NDMSRVR), and validates the user ID. If NDMSRVR opens and you are a valid user and logged on to NDMCOM, then you can issue commands through the API.

The following steps occur when you issue Connect:Direct HP NonStop commands through an API:

1. The API writes to the NDMCOM process and gets responses by reading $RECEIVE. Connect:Direct HP NonStop uses this method of communication because NDMCOM can respond to a single API request with multiple messages, rather than a one-to-one correlation in the traditional requester/server relationship.
2. NDMCOM parses the command. If the command is valid, NDMCOM sends the command to NDMSRVR.
3. NDMSRVR processes the request and formats the results into a C-string control structure for return to NDMCOM. Refer to *Elements of C-String Control Structures* on page 24 for a description of C-string format.
4. NDMCOM then sends the output to one of the following locations:
   - If the command includes the FILE parameter, such as SEL PROC FILE, the response is sent to the API as a C-string control block.
   - If the command includes either the PRINT or OUT option, such as SEL PROC PRINT or SEL PROC OUT, the response is sent to the specified output location.
   - If you do not specify FILE, PRINT, or OUT as a command parameter, the response is displayed to the standard output file (stdout) specified in the startup of the NDMCOM process.

The following figure shows the flow of a request through Connect:Direct HP NonStop.

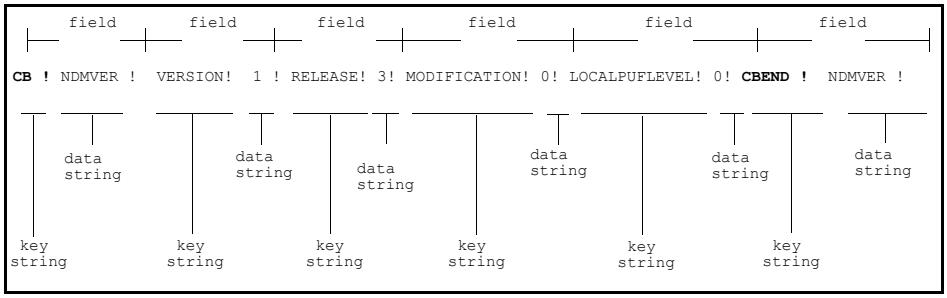


Elements of C-String Control Structures

A C-string control structure is a group of one or more related C-string control blocks. A C-string control block (CB) contains two or more fields. Each field has two strings, with a null (binary zero) character marking the end of each string. Throughout this document, the null character is represented by ! (exclamation point).

The first string in a field is the key (token); the second string in a field is the associated value (data) of the key. A key is always unique in a C-string control block. Two fields present in every control block have the keys CB and CBEND. CB is the field that begins the control block. CBEND is the field that ends the control block. The values for CB and CBEND indicate the name of the control block. The actual command output is positioned between these two fields in the control block.

For example, the following control block displays the output from successful Connect:Direct HP NonStop VERSION command. The fields in bold mark the required fields of every control block. The name of the control block is NDMVER, as shown by the values for the CB and CBEND keys. The output of the VERSION command consists of the version number (VERSION key and value), release number (RELEASE key and value), modification level (MODIFICATION key and value), and the maintenance level (LOCALPUFLEVEL key and value).



Refer to Chapter 3, *Programming the API*, for the description of the CB ! NDMVER ! control structure.

Error Control Structure

An error control structure (ERRCS) is a type of C-string control structure designed to store the messages that are generated from the execution of Connect:Direct HP NonStop Processes and commands. Each error control structure provides information detailing the number of messages captured for a particular command, message IDs, return codes, feedback codes, and optional data.

An ERRCS has one or more contiguous C-string control blocks. The first control block (an ERR control block) in an ERRCS is always present, with the beginning and ending fields as CB ! ERR ! and CBEND ! ERR !. The other two fields in the ERR control block are as follows:

Field	Description
N (number)	Specifies the number of messages in the ERRCS.
T (top message)	Specifies the number of the most important message.

The remaining control blocks in an ERRCS are message control blocks, which are sequenced as they occur. The fields in a message control block are as follows:

Field	Description
CB ! En !	Numbers the messages in an ERRCS. For example, E2 indicates that it is the second error message control block in an ERRCS.
FDBK ! fb !	Specifies the feedback code.
RC ! rc !	Specifies a completion code (RC) returned by Connect:Direct HP NonStop. A zero (0) value indicates successful operation.
MSGID ! msgid !	Specifies the message ID.
OK ! od ! (optional keywords ! optional data !)	Enables runtime information generated by Connect:Direct HP NonStop to return to the end user.
CBEND!En!	Indicates the end of this message control block

The following figure shows an example of an ERRCS generated by the successful execution of the SUBMIT command. An optional keyword, NEWPNUM, is returned for this command. Refer to Chapter 3, *Programming the API*, for additional commands that return optional keywords.

```
CB ! ERR ! N ! 1 ! T ! 0 ! CBEND ! ERR !
CB ! E1 ! NEWPNUM ! 36 ! FDBK ! 0 ! RC ! 0 !
MSGID ! SSRV101I ! CBEND ! E1 !
```

In the previous example, the ERRCS output is divided into control blocks and not shown in a continuous stream of data.

Every Connect:Direct HP NonStop command that is executed returns a CB ! ERR ! control block. If no errors occur during command execution, the CB ! ERR ! control block returns with zero (0) messages in the *N* field. For example:

```
CB ! ERR ! N ! 0 ! T ! 0 ! CBEND ! ERR !
```

The end of command execution is indicated by a CB ! ERR ! control block followed by a null-terminated NDMREADY.

Chapter 3

Programming the API

This chapter provides the following information you need to program and run an API:

- API basics
- Command control structure keywords
- ERRCS optional keywords
- CB function prototypes
- Message file structure

API Basics

The following API files are in the NDMAPI subvolume on the distribution tape:

File	Description
NDMAPI	The object file contains all C control block functions. You must bind the API written in C with this file.
NDMAPIB	This sample file contains binder commands to rebind NDMAPIC after it is compiled.
NDMAPIC	Contains the source code for all the CB functions written in C. Compile this module to produce NDMAPI in order to change or add CB functions.
NDMAPIH	Specifies a header file for use with NDMAPIC, USERAPIC, or other user-written programs.
NDMAPICH	Contains the #define statements for all control block keywords and the function declarations for C.
NDMAPITH	Contains the DEFINE statements for all control block keywords and the function declarations for TAL.
USERAPIC	Contains the source code for the sample C API. To compile an API written in C, code #INCLUDE NDMAPICH. You must also bind the API with NDMAPI.

Setting Parameters

Before running an API, set parameters for NDMCOM and the API, either through TACL as PARAM commands or by including the parameters in the API. Following are the NDMCOM parameters:

Parameter	Description
NDMAPI YES	Sets the API flag. This parameter is sent to NDMCOM at process creation and notifies NDMCOM that it is running in API mode. This parameter is hardcoded in the USERAPIC sample module.
NDMSRVR $<process name> (optionally used by the TAL API only)	Specifies the NDMSRVR process name, which is sent to NDMCOM at process creation time. The parameter notifies NDMCOM to issue an OPEN command on the specified process name in order for NDMCOM to communicate with NDMSRVR. The default is $NDMS. **Note:** The NDMSRVR process must be running before you start NDMCOM.

Following are the API parameters:

Parameter	Description
NDMCOM $<volume>.<subvolume>.<program name>	Specifies the location of the NDMCOM object file. The default is the current volume and subvolume.
NDMCOMNAME $<process name> (used by the C API only)	Specifies the name to be used for NDMCOM. The default value is $NCOM.
MSGFILE $<volume>.<subvolume>.<msgfile> (used by the TAL API only)	Specifies the location of the MSGFILE on your system.
NDMINFILE $<volume>.<subvolume>.<file> (optionally used by the TAL API only)	Specifies the location of a Process to submit through NDMCOM.

Running the API

After setting parameters, start the API as a process to allow communication between the API and NDMCOM. You can start the API as a named process using the optional parameter, name. The syntax for running an API is:

```
TACL>RUN <program name>/name/
```

Following is a description of the NDMCOM parameters:

Parameter or Command	Description
RUN	Executes an API.
program name	Name of the API.
name	Starts the API as a named process.

Understanding the API

The API manages the following activities:

- Building the parameters and startup messages for NDMCOM
- Creating and opening NDMCOM
- Passing the parameters and startup messages to NDMCOM

NDMCOM then automatically opens the API, sends the version, issues the OPEN command to NDMSRVR, and validates the user ID. If NDMSRVR opens, and you are a valid user and logged on to NDMCOM, then you can issue commands through the API.

Refer to the appropriate HP NonStop manuals for details on building parameters and startup messages. The program, USERAPIC, provides sample source code showing the preliminary steps. The sample program may need to be modified to ensure that file references are properly qualified, and the compiler environment includes all necessary subvolumes.

NDMCOM processes the data in its usual manner. If no output parameter is identified, the response is sent to the home terminal of the NDMCOM process. If the OUT or the PRINT option is defined, such as SEL PROC PRINT or SEL PROC OUT, the output is sent to the printer or a terminal. If the FILE parameter is used, such as SEL PROC FILE, the output is sent to the API as a C-string control block. Data sent to the API can then manipulated through Connect:Direct HP NonStop functions. Connect:Direct HP NonStop functions are provided in the NDMAPI object file for C programs or the TAL API program.

Note: After passing a command to NDMCOM, the API should always check for MSGID SSUB531I, *Invalid command.*

Issue the EXIT command from your API to close NDMCOM. Connect:Direct HP NonStop passes an ERRCS followed by NDMREADY to the API indicating successful completion of the command.

Refer to Chapter 5, *Sample Code*, for sample code that submits a Connect:Direct HP NonStop Process, monitors Process execution, and performs message handling functions.

Using the FILE Parameter

Use the FILE parameter to issue SELECT or DISPLAY commands, such as SELECT PROCESS, SELECT STATISTICS, SELECT USER, DISPLAY STATINFO and DISPLAY LOGGING. The FILE parameter specifies that the output from these commands bypasses the reportwriter and returns to the API in C-string format.

The following example illustrates a command with the FILE parameter:

```
SELECT PROCESS DETAIL FILE PNUMBER=1
```

Command Control Structure Keywords

The following tables describe the fields in various C-string control structures. Each table describes a particular command. Each command listing describes header information, the keywords as they should appear in control block functions, the keywords as they should appear in control blocks, and a description of the fields.

The content of C-string control structure is dependent on the command issued. Refer to *CB Function Prototypes* on page 43 for a description of how functions affect the control blocks they send and receive.

Note: Some commands only return an ERRCS; however, the select commands return a C-string control structure and an ERRCS.

About the DISPLAY STATINFO Control Structure

The following table describes the DISPLAY STATINFO control structure. The header is:

```
CB ! STATINFO !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_STATCRITERIA	STATCRITERIA	Deletion criteria for records in the statistics files.
CBKEY_STATMIDNITE	STATMIDNITE	Midnight flag for execution of NDMSTDL.
CBKEY_STATPERCENT	STATPERCENT	Maximum allowable percentage statistics files can be used before NDMSTDL is created.
“STDL_TIMESTAMP”	STDL_TIMESTAMP	Date and time NDMSTDL last executed.
“FILENAME0"	FILENAME0	STATFILE file name.
“EXTSIZE0"	EXTSIZE0	Primary extent size for STATFILE.
“SECEXTSIZE0"	SECEXTSIZE0	Secondary extent size for STATFILE.
“MAXEXT0"	MAXEXT0	Maximum number of extents for STATFILE.
“EXTALLOC0"	EXTALLOC0	Extents allocated for STATFILE.
“EOF0"	EOF0	End-of-file flag for STATFILE.
“FILEPERCENT0"	FILEPERCENT0	File percentage for STATFILE.
“MAXFILESIZE0"	MAXFILESIZE0	Maximum file size for STATFILE.
“FILENAME1"	FILENAME1	STATSRCH file name.
“EXTSIZE1"	EXTSIZE1	Primary extent size for STATSRCH.
“SECEXTSIZE1"	SECEXTSIZE1	Secondary extent size for STATSRCH.
“MAXEXT1"	MAXEXT1	Maximum number of extents for STATSRCH.

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
“EXTALLOC1"	EXTALLOC1	Extents allocated for STATSRCH.
“EOF1"	EOF1	End-of-file flag.
“FILEPERCENT1"	FILEPERCENT1	File percentage for STATSRCH.
“MAXFILESIZE1"	MAXFILESIZE1	Maximum file size for STATSRCH.
“FILENAME2"	FILENAME2	STATSRC0 file name.
“EXTSIZE2"	EXTSIZE2	Primary extent size for STATSRC0.
“SECEXTSIZE2"	SECEXTSIZE2	Secondary extent size for STATSRC0.
“MAXEXT2"	MAXEXT2	Maximum number of extents for STATSRC0.
“EXTALLOC2"	EXTALLOC2	Extents allocated for STATSRC0.
“EOF2"	EOF2	End-of-file flag for STATSRC0.
“FILEPERCENT2"	FILEPERCENT2	File percentage for STATSRC0.
“MAXFILESIZE2"	MAXFILESIZE2	Maximum file size for STATSRC0.

About the ENVIRONMENT Control Structure

The following table describes the ENVIRONMENT control structure. The header is:

```
CB ! ENV!.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_CDLOBJ	CDLOBJ	CDL object
CBKEY_OBEYVOL	OBEYVOL	Obeyvolume
CBKEY_PRINTER	PRINTER	Printer
CBKEY_PROCVOL	PROCVOL	Process volume
CBKEY_SRVR	SRVR	Server object
CBKEY_USER	USER	User
CBKEY_VOLUME	VOLUME	Volume (current volume)
“SAVEVOL”	SAVEVOL	Save volume
“SYSTEM”	SYSTEM	System

About the NETMAP Control Structure

The following table describes the NETMAP control structure. The headers are:

```
CB ! NODE ! | CB ! SNODE ! | CB ! LU! | CB ! LOGMODE!.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_ALLOC_RETRY_ADJ	ALLOC.RETRY.ADJ	Message ID. Indicates file allocation failure on an adjacent node.
CBKEY_ALLOC_RETRY_ADJ	“ALLOC.RETRY.ADJ”	Allocation errors to retry on adjacent node
CBKEY_APPLID	APPLID	Application ID
CBKEY_CPU	CPU	CPU
CBKEY_HIPIN	"HIGHPIN"	HIGHPIN indicator (Y/N)
CBKEY_IPADDR	"IPADDR"	Lists up to three IP addresses
CBKEY_IPDYNAMIC	IPDYNAMIC	Resolves symbolic address on connection request
CBKEY_LNODE	LNODE	Local node
CBKEY_LOCAL_CPNAME	"LOCAL.CPNAME"	Name of the local HP NonStop node control point
CBKEY_LOGMODE	LOGMODE	Name of the logmode record associated with a specific LU
CBKEY_LOGMODE_COMPROT	LOGMODE_COMPROT	Common LU protocols for the logmode
CBKEY_LOGMODE_FMPROT	LOGMODE_FMPROT	Function management profile for the logmode
CBKEY_LOGMODE_NAME	LOGMODE_NAME	Name of the logmode record
CBKEY_LOGMODE_PRIPROT	LOGMODE_PRIPROT	Primary LU protocol for the logmode
CBKEY_LOGMODE_PSERVIC	LOGMODE_PSERVIC	LU presentation services profile and usage field for the logmode
CBKEY_LOGMODE_PSNDPAC	LOGMODE_PSNDPAC	Primary sending pacing count
CBKEY_LOGMODE_RUSIZE	LOGMODE_RUSIZE	Transmission buffer size
CBKEY_LOGMODE_SECPROT	LOGMODE_SECPROT	Secondary LU protocol for the logmode
CBKEY_LOGMODE_SRCVPAC	LOGMODE_SRCVPAC	Secondary receive pacing count
CBKEY_LOGMODE_SSNDPAC	LOGMODE_SSNDPAC	Secondary send pacing count
CBKEY_LU	LU	LU
CBKEY_LU_MAXRETRY	LUMAXRETRY	Number of connection attempts to a session with the LU

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_LU_TIMEOUT	LUTIMEOUT	Time allowed for LU I/O
CBKEY_LU_USE	"LUUSE"	LU usage: A (any, default), P (PNODE), S (SNODE)
CBKEY_NAME	NAME	Name
CBKEY_PARSESS	PARSESS	Parallel sessions
CBKEY_PLEXCLASS	PLEXCLASS	Plex server ID
CBKEY_PORTNUM	"PORTNUM"	TCP/IP port number
CBKEY_SESSION_TYPE	SESSION.TYPE	Type of session
CBKEY_SNODE	SNODE	Secondary node
CBKEY_SNODE_MAXRETRY	SNODEMAXRETRY	Number of connection attempts to a session with the adjacent node
CBKEY_TCP_PROCESS_NAME	TCPNAME	List of TCP process names for outbound connections
CBKEY_TERM	TERM	Home terminal for session manager process or API manager process
CBKEY_TYPE	TYPE	Type
CBKEY_VOLUME	VOLUME	Default Volume/directory for data files

About the PROCESS Control Structure

The following table describes the PROCESS control structure. The header is:

```
CB ! SEL_PROC !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_CLASS	CLASS	Class (submitter class)
CBKEY_CLASS_EXEC	CLASSEX	Execution class
CBKEY_FDSN	FDSN	From data set name
CBKEY_FILE	FILE	Process file
CBKEY_FUNC	FUNC	Function executing
CBKEY_FUNC_COPY	FUNCCOPY	Function copy
CBKEY_INBYTE	INBYTE	Number of incoming bytes
CBKEY_LU	LU	LU (logical unit)

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_OTBYTE	OTBYTE	Number of received bytes
CBKEY_PNAME	PNAM	Process name
CBKEY_PNUM	PNUM	Process number (in SELECT commands)
CBKEY_PRI	PRI	Priority (submitter priority)
CBKEY_Q	Q	Queue state
CBKEY_RBYTES	RBYTES	Number of bytes received
CBKEY_REC_READ	RECRD	Number of records read
CBKEY_REC_WRITTEN	RECWR	Number of records written
CBKEY_RETAIN	RETAIN	Process retention
CBKEY_RU_RCVD	RURCVD	Number of RUs received
CBKEY_RU_SENT	RUSENT	Number of RUs sent
CBKEY_SBYTES	SBYTES	Number of bytes sent
CBKEY_SNODE	SNODE	Secondary node
CBKEY_START_DATE	STDATE	Start date (scheduled execution)
CBKEY_START_TIME	STTIME	Start time (scheduled execution)
CBKEY_STATE	STATE	State (execution state)
CBKEY_STEPNAME	STEPNAME	Step name (in the Process)
CBKEY_TDSN	TDSN	To data set name
CBKEY_UID	UID	User ID (submitter)
CBKEY_UNODE	UNODE	User node (submitter)

About the SECURITY Control Structure

The following table describes the SECURITY control structure. The header is:

```
CB ! SEC !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_SEC_OTHER_NODE	SEC_OTHER_NODE	Adjacent node
CBKEY_SEC_OTHER_USERID	SEC_OTHER_USERID	Adjacent node User ID
CBKEY_SEC_LOCAL_USERID	SEC_LOCAL_USERID	Local node User ID

About the STATISTICS Control Structure

The following table describes the STATISTICS control structure. Headers are:

```
CB ! SUBMIT ! | CB ! PROCSTART ! | CB ! STEPSTART ! | CB ! STEPEND ! | CB ! MSG ! | CB
! PROCEND ! | CB ! CMD ! |CB ! SESS !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_ATOE	ATOE	ASCII to EBCDIC (translation)
CBKEY_CB	CB	Executed step
CBKEY_CDATE	CDTE	End date
CBKEY_CLASS	CLASS	Class
CBKEY_CMD_NAME	CMDNM	Command name
CBKEY_COMPRESS	COMP	Compress (Y/N)
CBKEY_CTIME	CTME	End time
CBKEY_DATA_RATE	CDR	Number of bytes per second
CBKEY_DATE	DATE	Systems date
CBKEY_ETOA	ETOA	EBCDIC to ASCII (translation)
CBKEY_FDBK	FDBK	Feedback code
CBKEY_FDSN	FDSN	From data set name
CBKEY_FILE	FILE	File name
CBKEY_FROM_COMP_PER	FCMPPER	Compression percentage (sending side)
CBKEY_FROM_PNODE	FPNODE	From primary node
CBKEY_TO_PNODE	TPNODE	To primary node
CBKEY_FUNC	FUNC	Function
CBKEY_INBYTE	INBYTE	Number of incoming bytes
CBKEY_INRECN	INRECN	Number of incoming records
CBKEY_IPADDR	IPADDR	IP address of the adjacent node
CBKEY_LNKFL	LNKFL	Link stat
CBKEY_LU	LU	LU name
CBKEY_MSGID	MSGID	Message ID
CBKEY_MSG_TEXT	MSGTXT	Message text
CBKEY_NODE	NODE	Node type
CBKEY_OTBYTE	OTBYTE	Number of outgoing bytes

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_OTBLK	OTBLK	Number of outgoing blocks
CBKEY_OTRECN	OTRECN	Number of outgoing records
CBKEY_PC	PC	Process control
CBKEY_PGM	PGM	Program name
CBKEY_PNAME	PNAM	Process name
CBKEY_PNUM	PNUM	Process number
CBKEY_PRI	PRI	Priority
CBKEY_RBYTES	RBYTES	Number of received bytes
CBKEY_RC	RC	Return code
CBKEY_RSTRT	RSTRT	Step restart (Y|N)
CBKEY_RUSZ	RUSZ	RU size
CBKEY_RU_RCVD	RURCVD	Number of RUs received
CBKEY_RU_SENT	RUSENT	Number of RUs sent
CBKEY_SBYTES	SBYTES	Number of bytes sent
CBKEY_SDATE	SDATE	Start date
CBKEY_SECURECIPHER	SECURECIPHER	Secure+ cipher used
CBKEY_SECURECNAME	SECURECNAME	Common name of remote node's certificate
CBKEY_SECUREPROTOCOL	SECUREPROTOCOL	Secure+ protocol used SSLV3 or TLSv1
CBKEY_SNODE	SNODE	Secondary node
CBKEY_STEP	STEP	Step name in a STEPEND control block
CBKEY_STEPNAME	STEPNAME	Step name in Statistics control blocks other than STEPEND control block
CBKEY_STIME	STIME	Start time
CBKEY_SUB_ERR1	ERR1	Submit error message
CBKEY_TCP_PROCESS_NAME	TCP_PROCESS_NAME	TCP process used for the session
CBKEY_TDSN	TDSN	To data set name
CBKEY_TIME	TIME	System time
CBKEY_TO_COMP_PER	TCMPPER	Compression percentage (receiving side)
CBKEY_UID	UID	User ID (submitter)
CBKEY_UNODE	UNODE	User node (submitter)
CBKEY_XNODE	XNODE	Transmit node

About the TIME Control Structure

The following table describes the TIME control structure. The header is:

```
CB ! NDMTIME !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
“DAYOFWEEK”	DAYOFWEEK	Day of the week (text)
“MONTH”	MONTH	Month
“DAY”	DAY	Day of the month (number)
“YEAR”	YEAR	Year
“TIME”	TIME	Time in hours, minutes, seconds, and hundredths of seconds (HH:MM:SS:DDD)

About the TYPE Control Structure

The following table shows the TYPE control structure. The header is:

```
CB ! TYPE !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_ALTFILE	ALTF	Alternate file
CBKEY_ALTKEY	ALTK	Alternate key
CBKEY_AUDIT	AUD	Audit
CBKEY_AUDITCOMPRESS	AUDCOMP	Audit compress
CBKEY_BLOCK	BLK	Data block length
CBKEY_BUFFERED	BUFD	Buffered. Default is NOBUFFERED
CBKEY_BUFFERSIZE	BUFSIZE	Buffer size
CBKEY_COMPRESS	COMP	Compress. Default is NOCOMPRESS
CBKEY_DCOMPRESS	DCOMP	No Decompress. Default is NODCOMPRESS
CBKEY_FAST_LOAD	FASTLOAD	Fastload option (Y|N)

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_FAST_LOAD_CPU	FASTLOADCPU	Fastload option with CPU specified to run FUP (0–15)
CBKEY_FAST_LOAD_PRI	FASTLOADPRI	Fastload option with priority specified to run FUP (1–199)
CBKEY_FAST_LOAD_SORTED	FASTLOADSORTED	Fastload option with sorted data (Y|N)
CBKEY_FILE_CODE	FCODE	File code
CBKEY_FILE_TYPE	FTYPE	File type
CBKEY_FORMAT	FORMAT	Describes the preferred Enscribe file format (0, 1,2). 0 is the default value.
CBKEY_ICOMPRESS	ICOMP	Compress in index blocks. Default is NOICOMPRESS.
CBKEY_KEYLEN	KYL	Key length
CBKEY_KEYOFF	KYOFF	Key offset
CBKEY_LIKE_FILE	LFILE	Like file name
CBKEY_LIKE_TYPE	LTYPE	Like file type
CBKEY_MAXEXTENTS	MAXEXT	Maximum extents
CBKEY_NOALTCREATE	NOALTCR	No alternate create. Default is ALTCREATE.
CBKEY_NOBLOCKIO	NOBLOCKIO	No block I/O. Default is NOBLOCKIO.
CBKEY_NOLARGEIO	NOLARGEIO	No large I/O. Default is NOLARGEIO.
CBKEY_ODDUNSTR	ODDUS	No upward rounding of odd-numbered unstructured files.
CBKEY_PART	PART	Partition specifications for partitioned files
CBKEY_PARTONLY	PARTONLY	Subsequent file creation in partitions. Default is NOPARTONLY.
CBKEY_PRI_EXT	PRIEXT	Primary extents
CBKEY_RECSIZE	RECSIZE	Record size
CBKEY_REFRESH	REFRESH	Update volume label. Default is NOREFRESH.
CBKEY_SEC_EXT	SECEXT	Secondary extents
CBKEY_SERIALWRITES	SERWR	Serial writes to disk. Default is NOSERIALWRITES.
CBKEY_TYPEKEY	TYPE	Typekey name
CBKEY_VERIFIEDWRITES	VERIFIEDWRITES	Verify write. Default is NOVERIFYWRITES.
CBKEY_XLATE	XLATE	Translate data (ON|OFF)

About the USER Control Structure

The following table describes the USER control structure. The header is:

```
CB ! USER !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_ALIAS_CONVERT	ALIAS.CONVERT	Converts all aliases of this owner to the owner ID at logon (Y/N)
CBKEY_DEFAULT_AUTHORITY	DEFAUTH	Default authority (A|G|O|N)
CBKEY_MODIFY	MODIFY	Modify (Y|N)
CBKEY_MYID	MYID	MYID option (Y|N)
CBKEY_NAME	NAME	Name
CBKEY_LOGGING	LOGGING_UPD	Logging update command (Y|N)
CBKEY_NETMAP_DEL	NET_DEL	Network map delete (Y|N)
CBKEY_NETMAP_INS	NET_INS	Network map insert (Y|N)
CBKEY_NETMAP_REL	NET_REL	Network map relate (Y|N)
CBKEY_NETMAP_SEL	NET_SEL	Network map select (Y|N)
CBKEY_NETMAP_UPD	NET_UPD	Network map update (Y|N)
CBKEY_OBEYVOL	OBEYVOL	Obeyvolume
CBKEY_PHONE	PHONE	Phone
CBKEY_PROCVOL	PROCVOL	Process volume
CBKEY_PROC_CH	PROCALT	Process change (alter) (A|G|O|N)
CBKEY_PROC_DEL	PROCDEL	Process delete (A|G|O|N)
CBKEY_PROC_FLUSH	PROCSTOP	Process flush (stop) (A|G|O|N)
CBKEY_PROC_SEL	PROCSEL	Process select (A|G|O|N)
CBKEY_PROC_SUSPEND	PROCSUSP	Process suspend (A|G|O|N)
CBKEY_SEC_DEL	SECDELETE	Security delete (Y|N)
CBKEY_SEC_INS	SECADD	Security insert (Y|N)
CBKEY_SEC_SEL	SECLIST	Security select (list) (Y|N)
CBKEY_SEC_UPD	SECALT	Security update (alter) (Y|N)
CBKEY_STATISTICS	STATS	Select statistics (A|O|N)
CBKEY_STATISTICS_UPD	STATS_UPD	Update statistics command (Y|N)
CBKEY_STOPNDM	STOPNDM	Stop Connect:Direct HP NonStop operation (Y|N)

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
CBKEY_SUBMIT	SUBMIT	Submit command (Y|N)
CBKEY_TYPE_DEL	TYPEDEL	Typekey delete (Y|N)
CBKEY_TYPE_INS	TYPEADD	Typekey insert (Y|N)
CBKEY_TYPE_SEL	TYPESEL	Typekey select (Y|N)
CBKEY_TYPE_UPD	TYPEALT	Typekey update (alter) (Y|N)
CBKEY_USER	USER	User ID
CBKEY_USER_DEL	USERDEL	User delete (A|G|N)
CBKEY_USER_INS	USERINS	User insert (A|G|N)
CBKEY_USER_SEL	USERSEL	User select (A|G|O|N)
CBKEY_USER_UPD	USERUPD	User update (A|G|O|N)
CBKEY_VOLUME	VOLUME	Default volume for this user

About the VERSION Control Structure

The following tables show the VERSION control structure. The header is:

```
CB ! NDMVER !.
```

Following are the keywords as they should appear in control block functions, in control blocks, and a description of the fields:

Keyword in the Control Block Function	Keyword in the Control Block	Description of Field
“VERSION”	VERSION	Version
“RELEASE”	RELEASE	Release
“MODIFICATION”	MODIFICATION	Modification
“LOCALPUFLEVEL”	LOCALPUFLEVEL	Local maintenance level

ERRCS Optional Keywords

Upon successful execution, certain commands only return status messages in the form of ERRCS. The following table describes the number of errors/messages returned by each command, the optional keywords in control block functions, optional keywords in control blocks, and a field description.

Refer to *CB Function Prototypes* on page 43 for a description of how the various functions affect the control blocks they send and receive.

Command	# of Errors Returned	Optional Keyword Used in the Control Block Functions	Optional Keyword Used in the Control Block	Field Description
DELETE NETMAP LOCAL.NODE	1	CBKEY_NAME	NAME	Name of local node
INSERT NETMAP LOCAL.NODE	2	CBKEY_NAME	NAME	Name of local node
DELETE NETMAP LOGMODE	2	CBKEY_LOGMODE_NAME	LOGMODE_NAME	Name of the logmode
INSERT NETMAP LOGMODE	1	CBKEY_LOGMODE_NAME	LOGMODE_NAME	Name of the logmode
UPDATE NETMAP LOGMODE	1	CBKEY_LOGMODE_NAME	LOGMODE_NAME	Name of the logmode
DELETE NETMAP AMGR	4	CBKEY_APIMGR	AMGR	Name of the API manager
		CBKEY_SNODE	NODE	Name of the adjacent node
INSERT NETMAP AMGR	1	CBKEY_APIMGR	AMGR	Name of the API manager
UPDATE NETMAP AMGR	1	CBKEY_APIMGR	AMGR	Name of the API manager
DELETE NETMAP LU	4	CBKEY_NAME	NAME	Name of the local node
		CBKEY_LU	LU	Name of the LU
		CBKEY_NODE	NODE	Name of the adjacent node
INSERT NETMAP LU	1	CBKEY_LU	LU	Name of the LU
UPDATE NETMAP LU	1	CBKEY_LU	LU	Name of the LU
DELETE NETMAP ADJACENT.NODE	2	CBKEY_NAME	NAME	Name of the adjacent node
INSERT NETMAP ADJACENT.NODE	2	CBKEY_NAME	NAME	Name of the adjacent node
RELATE NETMAP ADJACENT.NODE	2	CBKEY_NAME	NAME	Name of the adjacent node
		CBKEY_LU	LU	Name of the LU

Command	# of Errors Returned	Optional Keyword Used in the Control Block Functions	Optional Keyword Used in the Control Block	Field Description
UPDATE NETMAP ADJACENT.NODE	1	CBKEY_NAME	NAME	Name of the adjacent node
DELETE USER	2	CBKEY_USER	USER	User name
INSERT USER	2	CBKEY_USER	USER	User name
UPDATE USER	2	CBKEY_USER	USER	User name
DELETE TYPE	2	CBKEY_TYPE	TYPE	Type name
INSERT TYPE	2	CBKEY_TYPE	TYPE	Type name
UPDATE TYPE	2	CBKEY_TYPE	TYPE	Type name
DELETE SECURITY	2	SECNODE	SECNODE	Adjacent node
		SECUSER	SECUSER	User ID at adjacent node
INSERT SECURITY	2	SECNODE	SECNODE	Adjacent node
		SECUSER	SECUSER	User ID at adjacent node
UPDATE SECURITY	2	SECNODE	SECNODE	Adjacent node
		SECUSER	SECUSER	User ID at adjacent node
VOLUME	1	CBKEY_VOL	VOL	Volume name
OBEYVOLUME	1	CBKEY_VOL	VOL	Obey volume name
PROCVOLUME	1	CBKEY_VOL	VOL	Process volume name
SUBMIT FILE	1	NEWPNUM	NEWPNUM	Process number assigned by Connect:Direct HP NonStop
CHANGE PROCESS	2	CBKEY_PNUM	PNUM	Process number
DELETE PROCESS	2	CBKEY_PNUM	PNUM	Process number
FLUSH PROCESS	2			
MODIFY SESSION	1			
STOP NDM I	1			
EXIT	1			
UPDATE STATISTICS CRITERIA	1			
UPDATE STATISTICS MIDNITE	1			
UPDATE STATISTICS PERCENT	1			

CB Function Prototypes

A CB function contains statements that perform specific tasks and often return a value to the statement that calls them. Use the following CB function prototypes in your API to manipulate control blocks. Use the prototypes for C applications except where noted. TAL prototypes for several commonly used routines can be found in NDMAPITH.

Note: Parameters in the functions are positional.

Prototype	Format	Description
CB_DATA	char *CB_DATA (char *)	Returns a pointer to the data string of the associated C-string field. Returns: *char *Example:* char *cb_field_ptr; char *data; data=CB_DATA (cb_field_ptr);
CB_NEXT_AVAILABLE	char *CB_NEXT_AVAILABLE (char *) Parameters: 1. (input) character pointer to the beginning of a C-string control structure. 2. (output) pointer to next available location for a C-string control structure	Receives a pointer to beginning of a C-string control block. Returns a pointer to the next location where a C-string control block should be built. *Example:* char *csptr; char *cbptr; csptr=CB^NEXT^AVAILABLE (cbptr)
CB_NEXT_CB	char *CB_NEXT_CB (char *)	Returns a pointer to the next control block in a C-string control structure. Returns a NULL value if the passed pointer is the last control block in the structure. Returns: *char *Example:* char *cbptr; char *next_cb_ptr; next_cb_ptr=CB_NEXT_CB (cbptr);
CB_FIND_CB	char *CB_FIND_CB (char *, char *) Parameters: 1. (input) character pointer to the beginning of a C-string control structure. 2. (input) pointer to a string containing the name of the requested control block.	Returns a pointer to the requested control block in a C-string control structure. Returns a NULL value if the passed pointer is not found. Returns: *char *Example:* char *cbptr; char *req_cb_ptr; req_cb_ptr=CB_FIND_CB (cbptr, “COPY”);

Prototype	Format	Description

CB_FIND_FIELD	char *CB_FIND_FIELD (char *, char *, char **)
	
	Parameters:
	1. (input) character pointer to the beginning of a target C-string control block.
	
	2. (input) pointer to a string containing the name of a requested field.
	
	3. (output) pointer to a pointer to char. Returns with a pointer to a pointer to the data portion of a target field.
		Returns a pointer to the field having the same name as the passed name argument.
		Returns: *char
		Example:
		char *cbptr;
		char *data;
		
		if ((CB_FIND_FIELD (cbptr, “FILE1", &data))==NULL) /* FILE1 was NOT found*/;
		else
		/* the data portion of the field points to the data associated with FILE1*/;

CB_POINT_MSG	char *CB_POINT_MSG (char *, char **, char **, char **) Parameters: 1. (input) pointer to the ERRCS. 2. (output) pointer to a pointer to the message ID returned. 3. (output) pointer to a pointer to the return code returned. 4. (output) pointer to a pointer to the feedback code returned.	Sets pointers to the most recent error in an ERRCS. Returns: char * pointer to message CB. The value is null if a message is not found. *Example:* char *errptr; char *msg; char *rc; char *fdbk; char *cb; cb=CB_POINT_MSG (errptr, &msg, &rc, &fdbk);
CB_TOP_MSG	char *CB_TOP_MSG (char *) Parameters: 1. (input) char * pointer to an ERRCS.	Sets pointers to the most important error in an ERRCS. Returns: char * pointer to message CB. The value is NULL if a message is not found. *Example:* char *errptr; char *msg; msg=CB_TOP_MSG (errptr);
CB_MSG_COUNT	short CB_MSG_COUNT (char *) Parameters: 1. (input) char * pointer to an ERRCS.	Returns the number of messages in an ERRCS. Returns: (short) number of messages in the ERRCS. *Example:* char *errptr; short count; count=CB_MSG_COUNT (errptr);
MSG_SHORT	char *MSG_SHORT (short , char *) Parameters: 1. (input) short message file number. 2. (input) char * pointer to message ID.	Reads the message file and returns a pointer to the short text of the message. If an error occurs during the read of the message file, a pointer to a null string is returned. Returns: char * pointer to short text of message. *Example:* char *mess; char *msgid; short file_num; mess=MSG_SHORT (file_num, msgid);

Prototype	Format	Description
MSG_OPEN	short MSG_OPEN (short *, short , char *, short *) Parameters: 1. short * file number (by reference). 2. short open mode (0x400=read only mode, 0x200=write only mode, 0=read/write mode). 3. char * file name. For TAL programs, the filename must be fully qualified; for example, $DATA.NDMFILES.MSGFILE. 4. short * rc return code. The value for the string is the return code.	Opens the message file. Returns: TRUE/FALSE *Example:* short flag; short error; short file_num; flag=MSG_OPEN (&file_num, 0x400, “MSGFILE”, &error);
MSG_DISPLAY	void MSG_DISPLAY (FILE * , short , char *) Parameters: 1. (input) FILE * fp out file pointer (opened with fopen call). 2. (input) short message file number (opened with MSG_OPEN). 3. (input) char * pointer to message ID.	Reads the message record from the message file and displays it on the OUT file. Only the message is displayed. No symbolic substitution occurs. Returns: void *Example:* char *id; short msgfnum; MSG_DISPLAY (stdout, msgfnum, ID);
ERRCS_DISPLAY	short ERRCS_DISPLAY (FILE * short, char *) Parameters: 1. (input) FILE * fp out file pointer (opened with fopen call). 2. (input) short message file number (opened with MSG_OPEN). 3. (input) char * pointer to ERRCS.	Displays messages from an ERRCS. If a top message is set, the top message is displayed first. Remaining messages are displayed in the order they occurred. Returns: void *Example:* char *errptr; short msg_fnum; ERRCS_DISPLAY (stdout, msg_fnum, errptr);

Message File Structure

The record format for the message file is composed of a C-string control block and one key, which is the message ID. The message ID is a maximum of 8 characters, plus one null byte (!).

Note: The entire message record, which includes short text, symbol indicator, and program name, can be no larger than 4096 bytes.

The following keywords define the structure of a message record:

Keyword	Definition
CB ! !	Required.
PGM ! <program name> !	Contains a maximum of 29 characters.
COMMON|IBM !	Indicates whether the message is *common* to both the host and Connect:Direct HP NonStop message files, or if the message is only a part of the IBM message file.
SYM ! <on|off> !	Indicates whether symbols are used in the message.
S ! <short text> !	Contains a maximum of 72 characters.
L ! <text> <\n> <text> <\n> <text><\n> !	Can consist of approximately 20 lines of text, with a maximum of 72 characters per line.
CBEND ! !	Required.

Example

The following example illustrates records in a message file:

```
The message ID (key) is :  SAPI101I
The program name is :  com_sel_command
It is a COMMON message.
No Symbols are used
The short text is : The select command was successfully executed.
The long text is : You can check your output file
Long text line 2 is : and check for completion codes.
```

The following example illustrates a sample control block describing the values. The percent sign (%) denotes a carriage return.

```
SAPI101I!!CB!PGM!com_sel_command!COMMON!!SYM!OFF!S!The select
command was successfully executed.!L!You can check your output file%
and check for completion codes.!CBEND!!
```

Chapter 4

Interface for User-Written Programs

Connect:Direct provides an interface to user-written programs that allows manipulation of data formats and database architectures not currently supported by Connect:Direct HP NonStop, as well as an interface to DataLoader MP. You determine whether to use a standard I/O exit, generic Inter-Processor Communications (IPC) processing, or DataLoader MP. This chapter details the types of I/O exits and provides information about determining which type of exit to define.

Determining the Type of Exit to Define

You determine when to use generic IPC processing or a standard I/O exit. Advantages of using an IPC I/O exit are:

- Generic IPC uses less overhead because only data is exchanged. No additional request buffers are sent.
- Generic IPC is independent of Connect:Direct request and response record formats. The IPC exit is only required to recognize the standard HP NonStop open and close messages, and the data.

Disadvantages of using an IPC exit are:

- Generic IPC I/O exit feature can only be used to write data received from a remote node. Data sent from HP NonStop to a remote node cannot use generic IPC.
- Connect:Direct does not receive error messages from the IPC exit. You must define error handling within the IPC exit program to ensure that messages are properly logged.

Advantages of using a standard I/O exit are:

- The exchange of standardized action requests means that Connect:Direct and the I/O exit may be synchronized at several stages so that each is proceeding at the same stage of the copy.
- Connect:Direct and a standard I/O exit may exchange error messages, allowing results to be recorded in the statistics log.

Disadvantages of using a standard I/O exit are:

- The exchange must adhere to the formats outlined in this chapter for specific request and response types.
- The extra exchange of requests and responses adds overhead and potentially slows the copy.

Specifying a Standard I/O Exit

Specify a standard I/O exit on the HP NonStop node in the FROM and TO clauses of the COPY statement. You can specify a different user-written I/O exit in each clause.

Note: Do not define DSN and FILE parameters in the same clause as the IOEXIT parameter.

The following example shows a Process that is submitted from a HP NonStop node and invokes an I/O exit on the HP NonStop node:

```
PROC1PROCESSSNODE=TAN.NODE1 SNODEID(USER01, &PW)
STEP1COPY TO  (PNODE,IOEXIT=(EIGHTCHR-
                '/CPU 1, NAME $ZXIT,-
                VOL \K2.$SYSTEM.NDMST, HIGHPIN OFF/'))-
                COMPRESS PRIME X'20'-
         FROM(SNODE DSN=&FR DISP=SHR)
```

You can specify the I/O exit in the COPY statement in two different ways:

- Specify the program name and startup parameters. The syntax for specifying an I/O exit with startup parameters follows:

```
IOEXIT=(ioexit-pgm  'startup-parameters')
```

- Specify the name of a Process with which Connect:Direct HP NonStop should communicate. Connect:Direct HP NonStop assumes that the Process is already running when the COPY operation is underway. The syntax for specifying an I/O exit where the process is already running follows:

```
IOEXIT=($process-name)
```

Note: The receiving Process must be started and be in read mode when Connect:Direct HP NonStop attempts to open the user Process.

The I/O exit ignores DCB information specified in a Process. Refer to the *Connect:Direct Process Statements Guide* for syntax and parameter descriptions for the IOEXIT parameter.

Invoking an I/O Exit on an OS/390 Node

Exits on the OS/390 node pass parameters in a different manner from those on a HP NonStop node. The Process in the following example is submitted from a HP NonStop node and invokes an I/O exit on the OS/390 node:

```
PROC1PROCESSSNODE=390.NODE1
    COPY FROM(IOEXIT=(IOEXIT,-
                "C'SYSTEM.BACKUP.FILE',-
                C'($DATA.FILES.CODE)',-
                X'05',        -
                C'NO',        -
                C'YES',       -
                C'OFF',       -
                C'ON',        -
                XL5'7F'    ")  -
              SNODE)          -
        TO(DSN=$DATA.FILES.CODE, DISP=RPL PNODE-
             SYSOPTS="SET XLATE ON")
```

Implementing an I/O Exit

The I/O exit function contains routines that start the specified I/O exit program, send the startup message and the I/O exit requests, and handle interprocess communications. When you include an I/O exit statement in the COPY statement, Connect:Direct HP NonStop starts the specified I/O exit program and sends it the startup message. The I/O exit program gets the startup message and the I/O exit requests through $RECEIVE. The input file parameter in the startup message is always $RECEIVE, because all the requests are sent through $RECEIVE. If you specify the input file in the I/O exit statement, Connect:Direct HP NonStop ignores it.

If Connect:Direct HP NonStop cannot start the I/O exit program or communicate with the I/O exit process, it cancels the Process and terminates the session.

After the I/O exit program successfully starts or opens, Connect:Direct HP NonStop initiates a BEGIN request. Other requests depend on whether Connect:Direct HP NonStop is the receiver or sender.

Sending Request Sequence

Connect:Direct HP NonStop uses this request sequence to send requests to the I/O exit on the sending side:

1. BEGIN request.
2. OPEN request.
3. INFO request.
4. GET request.
5. Connect:Direct HP NonStop passes data to the HP NonStop communications software.
6. CLOSE request.
7. END request.

Note: Steps 4 and 5 repeat until the I/O exit returns an end of data message.

Receiving Request Sequence

Connect:Direct HP NonStop uses this request sequence to send requests to the I/O exit on the receiving side:

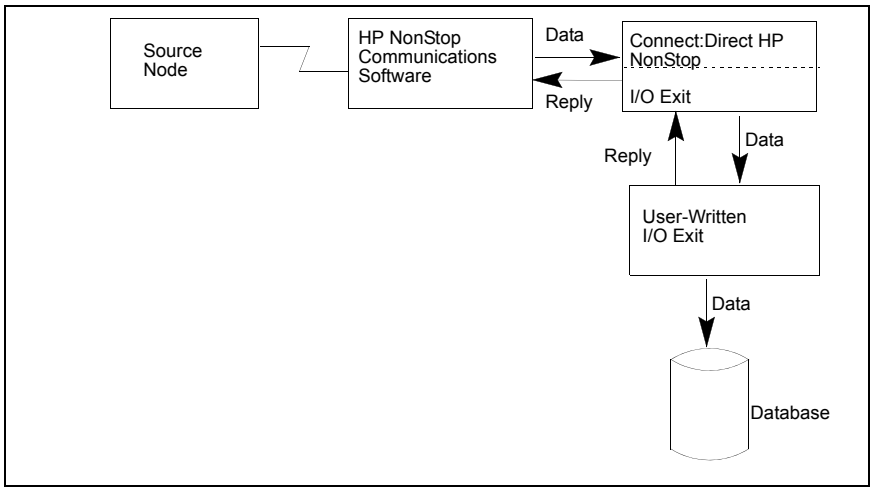
1. BEGIN request.
2. OPEN request.
3. INFO request.
4. Connect:Direct HP NonStop receives data from the HP NonStop communications software.
5. ADD request.
6. CLOSE request.
7. END request.

Note: Steps 4 and 5 repeat until the exit program receives all data from the HP NonStop communications software.

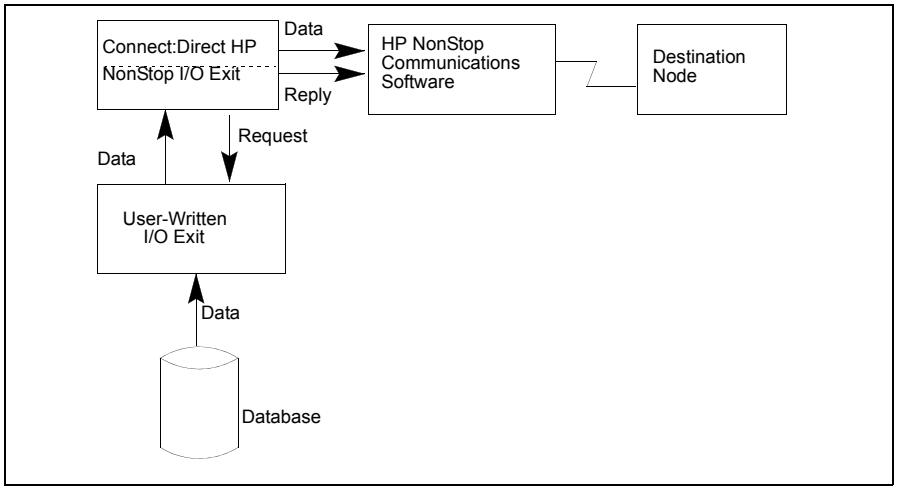
If the I/O exit program returns an error during the COPY operation, Connect:Direct HP NonStop sends a CLOSE request followed by an END request for an orderly shutdown of the I/O exit program. The status code returned by the I/O exit program in the exit control block as a response to the CLOSE or END request is ignored at this point. After the I/O exit program receives the END request, the I/O exit program stops itself.

If an ABEND occurs in the I/O exit program during the COPY operation, Connect:Direct HP NonStop cancels the operation and terminates the session.

The following figure displays the data flow of a standard I/O exit on the receiving side:



The following figure shows the data flow of a standard I/O exit on the sending side:



Sample I/O exits (IOEXITC and IOEXITT) are in the NDMSAMP subvolume. IOEXITC is written in C, and IOEXITT is written in TAL. Connect:Direct HP NonStop starts these sample exits and reads or adds data for specified data files depending on the types of requests (GET or ADD).

I/O Exit Requests

After you specify an I/O exit, Connect:Direct HP NonStop starts the user-written I/O exit program and sends it the startup parameters from the I/O exit statement. Then I/O exit requests are sent to the I/O exit program. All requests contain a common control block—the exit control block.

Note: If any error occurs during the I/O exit processing, Connect:Direct HP NonStop sends a CLOSE request followed by an END request to the I/O exit program. This procedure enables the I/O exit program to close the files that are open.

Connect:Direct HP NonStop sends the following requests to the I/O exit program:

Request	Description
ADD	Connect:Direct HP NonStop makes the ADD request to insert a record or block. The EXTOPER field in the exit control block has a value of ADD. The request consists of the exit control block and the record to be written to the database. The record is put in the buffer beginning at the offset indicated by the EXTOTARA field of the exit control block. The EXTOTLNG field contains the record length. The I/O exit program returns any error during the insertion of the record in the EXTRTNCD field.
BEGIN	Connect:Direct HP NonStop sends the BEGIN request to begin communication with the exit program. BEGIN is the first request that the I/O exit program receives. This request only consists of the exit control block. No user data is passed or expected. The EXTOPER field in the exit control block is a string of BEGIN.
CLOSE	Connect:Direct HP NonStop issues the CLOSE request to the I/O exit program to have the file closed. This request contains only the exit control block. No user data is passed or expected. The EXTOPER field in the exit control block has a value of CLOSE.
END	Connect:Direct HP NonStop makes the END request to an I/O exit program to end communication with the exit. END is the last request an I/O exit program allocates after it receives the BEGIN request. This request contains only the exit control block. No user data is passed or expected. The EXTOPER field has a value of END. Because all the processing is complete at this point, Connect:Direct HP NonStop ignores any error returned in the EXTRTNCD field in the exit control block. Connect:Direct HP NonStop sends this request to the I/O exit program, then stops the I/O exit program.
GET	Connect:Direct HP NonStop issues the GET request to the I/O exit program to read a record or block into the buffer. The I/O exit program must return the buffer where the record or block resides. This request consists of only the exit control block in which the EXTOPER field has a value of GET. The I/O exit program should return the user data in the reply. The EXTINARA field should indicate the offset within the reply at which the user data begins. The EXTINLNG field indicates the length of the user data returned. If an End of File (or End of Data) occurs, the I/O exit program sets the EXTRTNCD field to EXTRCEOD (hex 0004).

Request	Description

INFO	Connect:Direct HP NonStop sends the INFO request to the I/O exit program to retrieve the file attributes and place them into the INFOCB area.
	This request contains only the exit control block with INFO in the EXTOPER field. The I/O exit program should return the file attributes in the form of INFOCB that begins at the offset indicated by EXTVSWRK with a length indicated by EXTVSWKL.
	The following example describes the INFOCB structure:
	`* FILENAME file name or catalog name if SQL`
	`  03   FILENAME                    TYPE CHARACTER  36.`
	`* FILE-TYPE file type`
	`  03   FILE-TYPE                    TYPE BINARY     16.`
	`* FILE-CODE file code`
	`  03   FILE-CODE                    TYPE BINARY     16.`
	`* REC-LEN logical record length`
	`  03   REC-LEN                      TYPE BINARY     16.`
	`* BLOCKLEN Block Length`
	`  03   BLOCKLEN                     TYPE BINARY     16.`
	`* KEY-LEN primary key length for key-sequenced file`
	`  03   KEY-LEN                      TYPE BINARY     16.`
	`* KEY-OFF primary key offset for key-sequenced file`
	`  03   KEY-OFF                      TYPE BINARY     16.`
	`* PRI-EXT primary extents`
	`  03   PRI-EXT                      TYPE BINARY     32.`
	`* SEC-EXT secondary extents`
	`  03   SEC-EXT                      TYPE BINARY     32.`
	`* NUM-EXT number of extents allocated`
	`  03   NUM-EXT                      TYPE BINARY     16.`
	`* MAX-EXT maximum extents`
	`  03   MAX-EXT                      TYPE BINARY     16.`
	`END`

OPEN	Connect:Direct HP NonStop makes the OPEN request to an I/O exit program to allocate and open the file. The request indicates whether the file is to be read or written. This request contains only the exit control block with a string OPEN in the EXTOPER field. The EXTRTNCD field in the reply from the I/O exit program indicates whether the OPEN performed successfully.

Defining the Exit Control Block

The following example describes the exit control block, which serves as the basic message structure of the I/O exit requests.

```
* IOEXIT control block
DEF EXITCB.
* EXTCBLNG IOEXIT control block length
   03  EXTCBLNG                  TYPE BINARY 16.
* EXTIDENT IOEXIT Identification
*        IO = I/O Exit
*        SQ = SQL Exit
   03  EXTIDENT                  TYPE CHARACTER 2.
* EXTFORW forward chain pointer
   03  EXTFORW                   TYPE BINARY 16.
* EXTBACKW backward chain pointer
   03  EXTBACKW                  TYPE BINARY 16.
* EXTNAME name of user exit program
   03  EXTEXITN.
       05  VOL                   TYPE CHARACTER 8.
       05  SVOL                  TYPE CHARACTER 8.
       05  FNAME                 TYPE CHARACTER 8.
* EXTFNUM file number of user exit program
   03  EXTFNUM                   TYPE BINARY 16.
* EXTOPER requested exit operation
*           ADD, BEGIN, CLOSE, END, GET, INFO
   03  EXTOPER                   TYPE CHARACTER 8.
* EXTTASKN Task number
   03  EXTTASKN                  TYPE BINARY 16.
* EXTRTNCD return code from exit program
*           EXTRCOK    X'0000'   normal
*           EXTRCEOD   X'0004'   end of data
*           EXTRCLGC   X'FFFF'   logic error
   03  EXTRTNCD                  TYPE BINARY 16.
* EXTMSGID message id from exit program
   03  EXTMSGID                  TYPE CHARACTER 8.
   03  EXTWKARA                  TYPE BINARY 16.
* EXTINARA input record area offset
   03  EXTINARA                  TYPE BINARY 16.
* EXTOTARA output record area offset
   03  EXTOTARA                  TYPE BINARY 16.
* EXTINLNG input record length
   03  EXTINLNG                  TYPE BINARY 16.
* EXTOTLNG output record length
   03  EXTOTLNG                  TYPE BINARY 16.
* EXTMAXLEN maximum output record length
   03  EXTMAXLN                  TYPE BINARY 16.
* EXTSRECL source lrecl
   03  EXTSRECL                  TYPE CHARACTER 5.
* EXTSRECF source recfm
   03  EXTSRECF                  TYPE CHARACTER 4.
* EXTSBLKZ source blksize
   03  EXTSBLKZ                  TYPE CHARACTER 5.
* EXTDRECL destination lrecl
   03  EXTDRECL                  TYPE CHARACTER 5.
* EXTDRECF destination recfm
   03  EXTDRECF                  TYPE CHARACTER 4.
* EXTDBLKZ destination blksize
   03  EXTDBLKZ                  TYPE CHARACTER 5.
* EXTDIR direction of transfer,
*        S = sending side,R = receiving side
   03  EXTDIR                    TYPE CHARACTER 1.
```

Continued

```
* EXTFUNC requested function,
*        I = initialize, P = process record, E = exit
   03  EXTFUNC                    TYPE CHARACTER 1.
* EXTSFLAG source descriptor flags
*           EXTSRCL   X'80'   source lrecl specified
*           EXTSRCF   X'40'   source recfm specified
*           EXTSBKZ   X'20'   source blksize specified
*           EXTSIOX   X'10'   IOEXIT specified
*           EXTSSQL   X'08'   SQL specified
*           EXTSDBP   X'04'   DBPARMS specified
   03  EXTSFLAG                  TYPE CHARACTER 1.
* EXTDFLAG destination descriptor flags
*           EXTDRCL   X'80'   destination lrecl specified
*           EXTDRCF   X'40'   destination recfm specified
*           EXTDBKZ   X'20'   destination blksize specified
*           EXTDIOX   X'10'   IOEXIT specified
*           EXTDSQL   X'08'   SQL specified
*           EXTDDBP   X'04'   DBPARMS specified
   03  EXTDFLAG                  TYPE CHARACTER 1.
* EXTMISC general flags for exit processing
*           EXTCONI   X'80'   exit got cntl at least once
*           EXTCONP   X'40'   exit got cntl but has not returned
*           EXTCONR   X'20'   exit got cntl, return at least once
   03  EXTMISC                   TYPE CHARACTER 1.
* EXTFLAG1 more flags for exit processing
   03  EXTFLAG1                  TYPE CHARACTER 1.
* EXTIRECN # records read from database
   03  EXTIRECN                  TYPE BINARY 32.
* EXTIBLKN # blocks read from database
   03  EXTIBLKN                  TYPE BINARY 32.
* EXTORECN # records written to database
   03  EXTORECN                  TYPE BINARY 32.
* EXTOBLKN # blocks written to database
   03  EXTOBLKN                  TYPE BINARY 32.
* EXTIBYTN # bytes read from database
   03  EXTIBYTN                  TYPE BINARY 64.
* EXTOBYTN # bytes written to database
   03  EXTOBYTN                  TYPE BINARY 64.
* EXTCKPT1 for checkpointing
   03  EXTCKPT1                  TYPE CHARACTER 16.
* EXTVSWRK    INFOCB area offset        (waddr)
   03  EXTVSWRK                  TYPE BINARY 16.
* EXTVSWKL    INFOCB area length
   03  EXTVSWKL                  TYPE BINARY 16.
END
```

Sample Standard I/O Exit

Following is an example of standard IOEXIT Blocked processing:

```
BLOCKED    PROCESS PNODE=YOUR.PNODE                                    -
                   SNODE=YOUR.SNODE                                    -
                   SNODEID=(GROUP.USER,PASSWORD)

STEP10     RUN TASK   (PGM=FUP                                         -
           SYSOPTS="/OUT $S.#FUP.S10/PURGE DESTFILE!")

STEP20     RUN TASK   (PGM=FUP                                         -
           SYSOPTS="/OUT $S.#FUP.S20/CREATE DESTFILE")

STEP30     RUN TASK   (PGM=FUP                                         -
           SYSOPTS="/OUT $S.#FUP.S30/INFO DESTFILE,DETAIL")

COPYFILE   COPY                                                        -
                FROM (DSN=$VOLUME.SUBVOL.SRCFILE                       -
                      SNODE DISP=SHR)                                  -
                TO   (IOEXIT=($USERP)                                  -
                      PNODE DISP=RPL                                   -
                      SYSOPTS=("SET IPC N",                            -
                               "SET IPC.VB Y",                         -
                               "SET IPC.BLOCKLEN 32000"))
```

Specifying Generic IPC Processing

Connect:Direct HP NonStop supports a generic inter-processor communications (IPC) mechanism that gives you enhanced flexibility when communicating with user-defined Processes. You can write all data received by Connect:Direct to an alternate Process rather than directly to disk, spool, or tape. Connect:Direct can operate with any HP NonStop Process on an IPC level, but the receiving Process must be started before Connect:Direct attempts to perform an open operation.

When Connect:Direct HP NonStop receives incoming data from a remote node, the data is buffered, then written to a user-initiated process using IPC. The Connect:Direct HP NonStop Session Manager and your Process create a buffer to hold intermediate data and perform IPC write operations when the buffer is full (current maximum IPC size is 32,000 bytes). Unlike standard I/O exit processing where message overhead is required to communicate to the user-supplied I/O exit process, IPC I/O exit processing only sends data.

Types of Blocking

Generic IPC processing uses two forms of blocking of data between Connect:Direct and alternate processes: fixed and variable. To use IPC processing, you must ensure that data is fully processed and that all records are deblocked.

Fixed-block (FB) IPC assumes that all data sent to a Process is fixed-block records. No record lengths are contained within the IPC buffer sent to the alternate Process; therefore, you cannot generate a zero length record. When no more fixed length records fit into an IPC buffer, the buffer is written to the user Process for processing.

Variable-block (VB) IPC assumes that the data sent to the Process are of any record length. As with fixed-block transfers, when the IPC buffer fills, the buffer is written to the user Process. Variable-block IPC is the default for Connect:Direct HP NonStop. Within the buffer, each record is preceded by a 4-byte length indicator. The end of the buffer is indicated by a -1 (hex FFFF) in the final 4 bytes.

Specifying an IPC I/O Exit

Specify an IPC I/O exit on the HP NonStop node in the TO clause of the Connect:Direct HP NonStop COPY statement.

Specify the name of a Process with which Connect:Direct HP NonStop communicates. Connect:Direct HP NonStop assumes that the Process is already running when the COPY operation starts. The syntax for specifying an IPC I/O exit when the Process is already running follows:.

```
FILE =  $process-name     or    DSN = $process-name
```

Note: For generic IPC, you must use the keyword FILE or DSN and not IOEXIT.

Implementing an IPC I/O Exit

When you include an IPC I/O exit statement in the COPY statement, Connect:Direct HP NonStopopens the specified IPC I/O exit program.

If Connect:Direct HP NonStopcannot communicate with the IPC I/O exit process, it cancels the Process and terminates the session.

Note: IPC I/O exit processing occurs **only** on the HP NonStop platform.

The copy to an IPC I/O exit ignores DCB information specified in a Process. Refer to the Connect:Direct Process Statements Guide for syntax and parameter descriptions for the IOEXIT parameter.

Required Parameter

Generic IPC processing uses the following required SYSOPTS parameter:

Parameter	Description
IPC <yes-no>†	Specifies whether the RECEIVE task uses generic IPC processing to manage output. This parameter is required for IPC Processing, where <yes-no> ::= {Y[ES] | N[O]}.

† The receiving Process must be started and be in read mode when Connect:Direct HP NonStop attempts to open the user Process.

Optional Parameters

The optional parameters for generic IPC processing are listed in the following table. These parameters only apply if IPC=yes.

Parameter	Description
IPC.BLOCKLEN <nnnnn>	IF IPC=Yes: IPC.BLOCKLEN indicates the length of the buffer used for data that is read from the remote Connect:Direct HP NonStop node then written to a user initiated process. The value ranges from 4096 (4K) bytes to the current IPC maximum of 32000 bytes, where <length> ::= {4096..32000}.

Parameter	Description
IPC.PAD <nnn>	IPC.PAD represents the pad character value used for blocked IOEXIT processing. The pad character is uniform for fixed length blocks only. Valid values are 0–255.
IPC.RECLEN <nnnnn>	IPC.RECLEN represents the length of each record used for data that is either read from or written to a remote Connect:Direct HP NonStop node. Valid values range from 4096 to 32000. **Note:** The default value assumes 32,000-byte buffers for IPC_BLOCKLEN.
IPC.VARIN <yes |no>	IPC.VARIN specifies whether the RECEIVE task uses variable length records with generic IPC processing. Refer to the *HP NonStop Operations Utility Guide* for more details concerning the FUP VARIN format. If you use fixed-block records, you must explicitly request the IPC.VARIN parameter with a value of NO, where <yes-no> ::= { Y[ES] | N[O] }.

Integrating Dataloader/MP

Connect:Direct HP NonStop can integrate Dataloader/MP during all HP NonStop receiving operations. This improves performance when transmitting partitioned data, load operations, and SQL-based data. The Connect:Direct Dataloader/MP extensions increase normal functionality with newer file structures, such as Compaq FORMAT 2 files.

How Dataloader/MP Works

To maximize throughput, Dataloader/MP uses Inter-Process Communications (IPC) operations. When Connect:Direct receives incoming data from a remote node, the data is buffered, then written to a newly created Dataloader/MP process by using IPC. The Connect:Direct Session Manager and Dataloader/MP create a variable blocked structure to hold intermediate data and perform IPC write operations when the buffer is full (current maximum IPC size is 32,000 bytes). In addition to buffering, IPC I/O operations are queued so that disk I/O throughput is maximized.

For more details concerning HP NonStop Dataloader/MP capabilities, refer to the *HP NonStop Dataloader/MP Users Guide*.

Specifying a Dataloader/MP Exit

Specify the name of the DataLoader process with which Connect:Direct HP NonStop communicates. Connect:Direct HP NonStop assumes that the Process is already running when the COPY operation starts. The syntax for specifying a DataLoader/MP exit where the Process is already running follows:

```
IOEXIT = $process-name
```

Note: For DataLoader/MP, you must use the keyword IOEXIT and not DSN or FILE.

Dataloader/MP Parameters

To implement Dataloader/MP, use the following SYSOPTS parameters:

Parameter	Description
IPC.DATALOAD <yes-no>	Specifies if the RECEIVE task uses HP NonStop Dataloader/MP utility program to manage output. This parameter is required for IPC processing. The default is NO. Where <yes-no> ::= { Y[ES] | N[O] }
IPC <yes-no>†	Specifies if the RECEIVE task uses generic IPC process to manage output. This parameter is required for IPC processing. The default is NO. Where <yes-no> ::= {Y[ES] | N[O]}

† The receiving process must be started and be in read mode when Connect:Direct HP NonStop attempts to open the user process.

Note: The DataLoader/MP interface also supports all of the optional SYSOPTS parameters specified for generic IPC, that is, IPC.BLOCKLEN, IPC.PAD, IPC.RECLEN, and IPC.VARIN.

Example Process Stream

Following is an example of using the Dataloader/MP parameters:

```
IO102      PROCESS PNODE=YOUR.PNODE                                -
                   SNODE=YOUR.SNODE                                -
                   SNODEID=(GROUP.USER,PASSWORD)
/* Purgedata the OUTPUT file */
STEP10     RUN TASK PNODE PGM=FUP-
            SYSOPTS="/OUT $S.#FUP.KSDS/PURGEdata $vol.subvol.file"
/* RUN the IOEXIT sample */
/* OUTPUT file MUST be Key-Sequenced, 80 bytes with a 4 byte key  for testing */
STEP20     COPY                                                    -
           FROM (dsn=$vol.subvol.file                          -
                 SNODE disp=shr)                                   -
          TO (PNODE IOEXIT=$DLD1 -
                SYSOPTS=("SET IPC Y" -
                         "SET IPC.DATALOAD Y" -
                         "SET IPC.BLOCKLEN 80" -
                         "SET IPC.PAD 55" -
                         "SET IPC.RECLEN 80"))
```

Following is an example of Dataloader (special case of IPC IOEXIT processing):

```
DATALOAD   PROCESS PNODE=YOUR.PNODE                              -
                   SNODE=YOUR.SNODE                              -
                   SNODEID=(GROUP.USER,PASSWORD)

STEP10     RUN TASK   (PGM=FUP                                   -
           SYSOPTS="/OUT $S.#FUP.S10/PURGE DESTFILE!")

STEP20     RUN TASK   (PGM=FUP                                   -
           SYSOPTS="/OUT $S.#FUP.S20/CREATE DESTFILE")

STEP30     RUN TASK   (PGM=FUP                                   -
           SYSOPTS="/OUT $S.#FUP.S30/INFO DESTFILE,DETAIL")

COPYFILE   COPY                                                  -
                FROM (DSN=$VOLUME.SUBVOL.SRCFILE                 -
                      SNODE DISP=SHR)                            -
                  TO (IOEXIT=$DL1 -
                      PNODE DISP=RPL -
                       SYSOPTS=("SET IPC Y", -
                                "SET IPC.VB N", -
                                "SET IPC.DATALOAD Y", -
                                "SET IPC.BLOCKLEN 80"))
```

Following is an example of IPC IOEXIT processing:

```
IPCexit    PROCESS PNODE=YOUR.PNODE                              -
                   SNODE=YOUR.SNODE                              -
                   SNODEID=(GROUP.USER,PASSWORD)

STEP10     RUN TASK   (PGM=FUP                                   -
           SYSOPTS="/OUT $S.#FUP.S10/PURGE DESTFILE!")

STEP20     RUN TASK   (PGM=FUP                                   -
           SYSOPTS="/OUT $S.#FUP.S20/CREATE DESTFILE")

STEP30     RUN TASK   (PGM=FUP                                   -
           SYSOPTS="/OUT $S.#FUP.S30/INFO DESTFILE,DETAIL")

COPYFILE   COPY                                                  -
                FROM (DSN=$VOLUME.SUBVOL.SRCFILE                 -
                      SNODE DISP=SHR)                            -
                TO   (IOEXIT=($USERP)                            -
                      PNODE DISP=RPL                             -
                      SYSOPTS=("SET IPC Y",                      -
                               "SET IPC.VB N",                   -
                               "SET IPC.BLOCKLEN 32000"))
```

Chapter 5

Sample Code

The example in this chapter illustrates concepts and considerations that are useful when you write an API. The sample code, written in C, shows how to submit a Connect:Direct HP NonStop Process, handle messages, and check the status of Process execution with Connect:Direct HP NonStop SUBMIT and PROCESS commands.

Before issuing commands through an API, you must set parameters for both NDMCOM and the API and start the API as a named process. Refer to *Setting Parameters* on page 28 for information about creating and running APIs and to the source program, USERAPIC, in the NDMAPI subvolume for sample code.

Example

First, the API opens the message file (MSGFILE) to search for any message IDs that require action.

```
short    fnum_msgfile = -1;
short    error;

if (! MSG_OPEN    (&fnum_msgfile          /* file number        */
                  ,0x0400                 /* Read-only          */
                  ,"MSGFILE"              /* file name          */
                  ,&error))               /* return error       */

{
  fnum_msgfile = -1;
  fprintf (stdout, "Unable to open message file, error
           #(%d)\n", error);
  exit (-1);
}
```

Next, build the NDMCOM command to submit a Process.

For example: SUBMIT FILE=$SYSTEM.NDMPROC.SEND

```
        char     send_buffer [4096];
        short    rc;
        short    write_cnt;
        short    cnt_write;

strcpy (send_buffer, "sub file=$system.ndmproc.send");
write_cnt = strlen (send_buffer) + 1;
```

Perform a WRITEX to the NDMCOM file pointer to set the length in write_cnt and check for errors. The ndmcom_fp variable contains the file number obtained by the creation of NDMCOM through the API.

```
rc = WRITEX (ndmcom_fp, send_buffer,
write_cnt, &cnt_write);
if (rc != 0)
  exit (-1);
```

If you receive no indication of errors, perform a do-while loop to read data (control blocks) coming from NDMCOM to determine further steps. If no errors occur, two control blocks are returned.

This example searches for a MSGID of SSRV101I, which indicates that the Process was submitted successfully, and searches for the keyword *NEWPNUM* to extract the new Process number assigned to the Connect:Direct HP NonStop Process by NDMSRVR.

```
            short  read_cnt;
            short  cnt_read;
            short  save_proc_num;
            short  end_flag = 0;
            char   *data;
            char   receive_buffer[4096];
            char   *cbptr;
            char   *csptr;
do
{
   read_cnt = sizeof (receive_buffer);
   rc = READX (api_fp, receive_buffer, read_cnt, &cnt_read);
   csptr = cbptr = receive_buffer;
```

Find the N keyword in the CB to determine the number of error messages present in the CB ERR. Refer to *Error Control Structure* on page 25 for details on the N keyword.

The following example shows a control block passed to the API due to successful execution of the SUBMIT command. The character ! (exclamation point) denotes a binary zero (null).

```
CB!ERR!N!1!T!0!CBEND!ERR!CB!E1!NEWPNUM!5!
FDBK!0!RC!0!MSGID!SSRV101I!CBEND!E1!
```

```
   CB_FIND_FIELD (cbptr, "N", &data);
   if (atoi (data) == 1)
   {
```

If the data portion of the field *N* is equal to 1, skip to the next CB to find the MSGID and NEWPNUM fields.

```
         cbptr = CB_NEXT_CB (cbptr);
         CB_FIND_FIELD (cbptr, CBKEY_MSGID, &data);

      if (strcmp (data, "SSRV101I"))
      {
```

If the MSGID is not equal to SSRV101I, display the contents of CB ERR and terminate the API.

```
                ERRCS_DISPLAY (stdout, fnum_msgfile, csptr);
                exit (-1);
                }
                else
                {
```

NEWPNUM is the keyword for the field that returns the new Process number after a Process is submitted successfully.

```
                CB_FIND_FIELD (cbptr, "NEWPNUM", &data);
                fprintf (stdout, "process submitted & the proc # is \
                %s\n", data);
                save_proc_num = atoi(data);

           } /* end else */
```

If the ERR CB has no messages coming back, display the ERR CB and exit from the API.

```
           ERRCS_DISPLAY (stdout, fnum_msgfile, cbptr);
           exit (-1);

      } /* end if data */
```

NDMCOM always returns an error control block, with or without messages, after every Connect:Direct HP NonStop command. NDMCOM returns an NDMREADY message to the API after the error control block to indicate that there are no further error control blocks for that command.

```
   } while (strcmp (receive_buffer, "NDMREADY") != 0);
```

If there are no errors, issue the following command to monitor the status of the Process:

```
   SELECT PROCESS DETAIL FILE PNUM=N
```

You must specify the FILE parameter with any of the select commands for data to bypass the NDMCOM reportwriter and return in C-string control block format.

```
            short  cnt_write;
            short  write_cnt;
            char   command[50];

   strcpy (command, "select process detail file pnum=n");
   strcat (command, data);

   strcpy (send_buffer,command);
   write_cnt = strlen (send_buffer) + 1;
```

Perform a WRITEX to NDMCOM file pointer (ndmcom_fp) for the length of write_cnt and to check for errors.

```
do
{
     rc = WRITEX (ndmcom_fp, send_buffer, write_cnt,&cnt_write);
     if (rc != 0)
         exit (-1);
```

After you send the SELECT PROCESS command to NDMCOM, perform a READX operation to analyze the returning control blocks.

Note: After passing a command to NDMCOM, the API should always check for MSGID SSUB531I, indicating an invalid command.

Because the SELECT PROCESS command monitors data transmission, different information returns in control blocks every time the command is invoked during Process execution. The code in this example loops until the end of Process execution and the MSGID SAPI101I returns. Your API determines the action on the data returned.

```
   read_cnt = sizeof (receive_buffer);
   rc = READX (api_fp, receive_buffer,read_cnt,&cnt_read);
   csptr = cbptr = receive_buffer;
```

Determine whether the control block is an error control block. If an error control block is present, determine whether the Process completed its execution or was issued using invalid syntax.

The following control block indicates the Process was not found:

```
CB!ERR!N!1!T!0!CBEND!ERR!CB!E1!
FDBK!0!RC!0!MSGID!SAPI101I!CBEND!E1!
```

This control block shows an invalid command:

```
CB!ERR!N!1!T!0!CBEND!ERR!CB!E1!
ERR1!INVALID COMMAND! FDBK!0!RC!0!
MSGID!SSUB531I!CBEND!E1!
```

Following is a control block for a Process in execution state:

```
CB!SEL_PROC!DTL!!EXECUTING!!RECRD!0!RECWR!
RUSENT!1!RURCVD!0!SBYTES!3900!RBYTES!0!CLASSEX!1!
INBYTE!0!OBYTES!3900!NODE!F!PNUM!5!PNAM!SEND!
STEPNAME!SENDCOPY!
```

Note that the CB!SEL_PROC! control block does not end with CBEND!SEL_PROC!. The Process is still in its execution state and NDMSRVR is still writing to the TCQ file.

```
    if (CB_FIND_FIELD (cbptr, "CB", &data) && !strcmp (data,
       "ERR"))
    {
       CB_FIND_FIELD (cbptr, "N", &data);
       if (atoi (data) == 1)
       {
```

If the data for the N equals 1, then skip to the next CB to find the MSGID.

```
         cbptr = CB_NEXT_CB (csptr);
         CB_FIND_FIELD (cbptr, CBKEY_MSGID, &data);
         if (! strcmp (data, "SSUB531I"))
         {
```

If the MSGID equals SSUB531I, an invalid command, display the message ID and terminate the API.

```
              ERRCS_DISPLAY (stdout, fnum_msgfile, csptr);
              exit (-1);
         }
         else
         {
```

If the MSGID equals SAPI101I, notify the user that the Process finished and OPR.JONES must perform further action.

```
              if (!strcmp (data, "SAPI101I"))
              {
                 fprintf(stdout, "OPR.JONES, proc #%d completed; \
                   check for further action.\n", save_proc_num);
                 end_flag = 1;
                 continue;
               } /* end of proc */

         } /* end else */

     } /* end if data */
```

If the returned control block is not an error control block, pass the control blocks to STDOUT.

```
      dump (stdout, (char *) cbptr, (size_t) 300);

  } /* end if ERR */
```

Check for the end flag to equal TRUE to terminate processing, otherwise the do-while loop is executed again.

```
   } while (end_flag == 0);

  exit (0);
```

Chapter 6

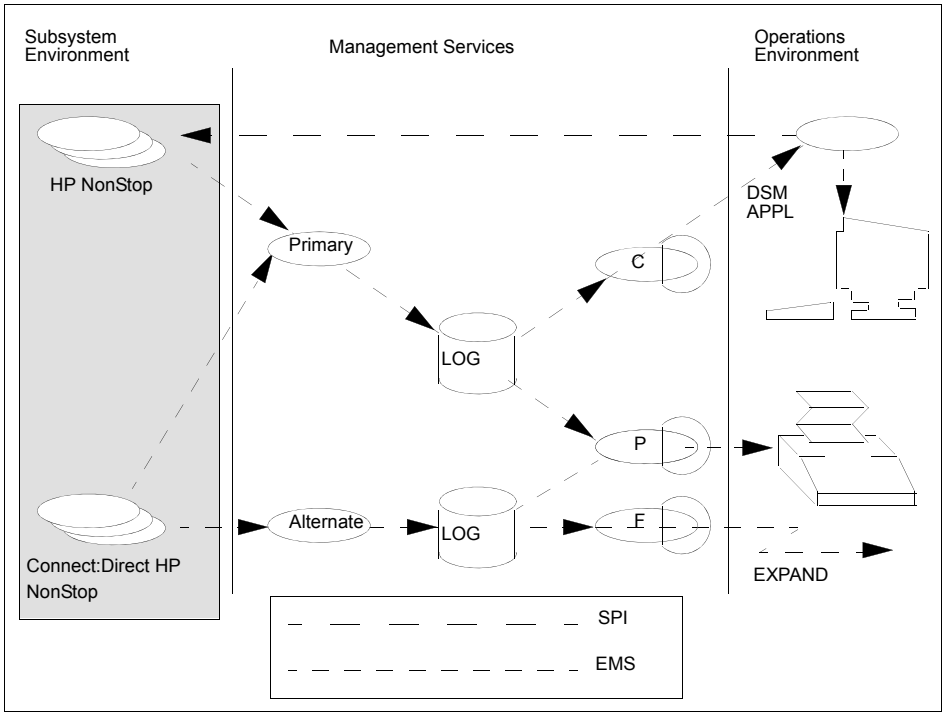
Using DSM/EMS Event Reporting

Event Management Service (EMS) is a collection of processes, tools, and interfaces that provide event-message collection and distribution in the HP NonStop Distributed Systems Management (DSM) environment. Connect:Direct HP NonStop generates and reports event messages to EMS, and writes event messages to statistics file. Each type of statistics record is a different event to EMS. In addition, Connect:Direct HP NonStop generates event messages for the startup and shutdown of such processes as NDMSRVR and session managers.

You can set up a dedicated console by using a printing distributor and a filter. The distributor uses a filter, and only the events that meet the specifications of the filter are passed. You can configure Viewpoint, a DSM console application, to filter Connect:Direct HP NonStop event messages. Refer to the HP NonStop documentation for EMS operation.

Connect:Direct HP NonStop provides all the facilities to customize operation in a DSM environment. For example, you can create complex filters using the samples provided with the product. Modify the sample templates as appropriate for your environment.

The following figure shows Connect:Direct HP NonStop in a DSM environment. Connect:Direct HP NonStop can log to a primary or alternate collector. The three types of distributors are consumer, printing, and forwarding. The DSM application can be Viewpoint.



Sample filters and a text template for event-message reporting are in the NDMDSM subvolume. Modify these samples to customize your Connect:Direct HP NonStop environment. Refer to *Integrating Connect:Direct HP NonStop in a DSM Environment* on page 70 for additional details.

Control the disposition of event messages from the initialization parameters file. You can override the values with a Connect:Direct HP NonStop command, UPDATE LOGGING. An overview of initialization parameters and logging commands follows.

Note: Connect:Direct HP NonStop reports the output resulting from server errors to EMS and the server output file as you defined in the RUN statement to start NDMMON and the server.

EMS-Specific Initialization Parameters

The following initialization parameters control the disposition of event messages:

Parameter	Description
EMS	Controls whether Connect:Direct HP NonStop logs events to EMS. Valid values are OFF and ON. The default is OFF.
STATS	Controls whether Connect:Direct HP NonStop logs events to the statistics file. Valid values are OFF and ON. The default is ON.
COLLECTOR	Specifies the collector you want to use for event-message generation. The collector can be the primary collector for the system ($0) or an alternate collector. The default collector is $0.

Set either EMS or STATS to ON. If you set both parameters to OFF, Connect:Direct HP NonStop forces STATS ON.

Logging Commands

You can use the following commands in conjunction with EMS logging:

- The UPDATE LOGGING command enables you to alter settings for EMS, STATS, and COLLECTOR. Refer to the *Controlling EMS Logging* chapter in the *Connect:Direct HP NonStop Administration Guide* for command syntax and examples.

  If you specify the collector parameter and EMS logging is on or if you issue the UPDATE LOGGING command to activate it, Connect:Direct HP NonStop closes and opens the collector file. This action automatically resets the connection between Connect:Direct HP NonStop and the collector.

- The DISPLAY LOGGING command displays the EMS settings, the STATS settings, and the name of the collector process. Optional parameters associated with this command are FILE, PRINT, and OUT.

Distribution Files

The following files, pertinent only to EMS, are in the NDMDSM subvolume on the distribution tape.

File	Description
NDMDDL	The source file for token definitions for Connect:Direct HP NonStop. Field type definitions for all EMS tokens are in this file. Use this file to produce a data dictionary. **WARNING:** Do not modify NDMDDL.
NDMC	The DDL output of NDMDDL for C.
NDMTACL	The DDL output of NDMDDL for TACL.
NDMTAL	The DDL output of NDMDDL for TAL.

File	Description
FLTSRC1	The sample filter source that passes all Connect:Direct HP NonStop events.
FLTSRC2	The sample filter source that passes Connect:Direct HP NonStop events that are step starts and step ends.
FLTSRC3	The sample filter source that passes Connect:Direct HP NonStop events with a return code greater than zero.
FLTSRC4	The sample filter source that passes Connect:Direct HP NonStop Spooler option events.
NDMFLT1	The filter object file for FLTSRC1.
NDMFLT2	The filter object file for FLTSRC2.
NDMFLT3	The filter object file for FLTSRC3.
NDMFLT4	The filter object file for FLTSRC4.
RUNDIST	The sample obey file for starting a printing distributor.
STALTCOL	The sample obey file for starting an alternate collector.
TEMPSRC	The sample template source file designed to be used with a printing distributor. TEMPSRC provides more detail than TEMPVIEW.
TEMPOBJ	The object file produced by the template compiler for TEMPSRC. Add this file to the CONFTEXT file to install templates during SYSGEN.
TEMPVIEW	The sample template source file designed to be used with Viewpoint or NonStop NET/MASTER. TEMPVIEW packs pertinent information on a single line.
TEMPVOBJ	The object file produced by the template compiler for TEMPVIEW. Add this file to the CONFTEXT file to install templates during SYSGEN.

Integrating Connect:Direct HP NonStop in a DSM Environment

You can integrate Connect:Direct HP NonStop into a DSM environment using the files in the NDMDSM subvolume. If you are customizing your environment, you can use the files as a starting point.

RUNDIST is a sample obey file for running a printing distributor. To start a printing distributor, modify the filter parameter in RUNDIST to point to the volume where Connect:Direct HP NonStop is installed. Three sample filter source files are provided.

Use one of the following filters for the type of messages you require:

Filter	Description
NDMFLT1	Passes all Connect:Direct HP NonStop events.
NDMFLT2	Passes Connect:Direct HP NonStop events that are step starts and step ends.
NDMFLT3	Passes Connect:Direct HP NonStop events with a return code greater than zero.
NDMFLT4	Passes Connect:Direct HP NonStop Spooler option events.

Templates are provided in object format. This format allows you to add the templates to your system in your CONFTEXT file or through the Configuration Utility Program (COUP). For temporary use or testing, add _EMS_TEMPLATES to TACL with the DEFINE statement.

The tokens in NDMTACL allow you to construct complex filters. To compile the filters you write, load the TACL variables in the NDMTACL file. Use the filter sources as references. For more information on the filter language and the compiler, refer to the *HP NonStop Event Management Service Manual*.

Modify the provided templates for Connect:Direct HP NonStop as appropriate to your environment. The source for the templates is in TEMPSRC. Use NDMDDL, the DDL source file, to compile the new templates and create a data dictionary containing Connect:Direct HP NonStop tokens and events.

You can use TEMPNRES to generate your system's nonresident template file. The TEMPLI utility constructed the delivered TEMPNRES on a D30.02 system. The TEMPNRES should be regenerated to run on an operating system version different from D30.02. Refer to the *HP NonStop DSM Template Services Manual* for complete information on templates.

The files, NDMC, NDMTACL, and NDMTAL, contain token definitions and are provided for you to use in any DSM applications you write.

Connect:Direct HP NonStop Tokens

The following table lists Connect:Direct HP NonStop tokens (in TACL format) and a brief description of each token. The tokens are found in events generated by Connect:Direct HP NonStop. Tokens are specified in TACL format when constructing filters. The tokens are in C and TAL format in NDMC and NDMTAL, respectively.

Token	Description
NDM^TKN^CDATE	End date
NDM^TKN^CLASS	Class
NDM^TKN^COMPRESS	Compress (Y|N)
NDM^TKN^CTIME	End time
NDM^TKN^DATA^RATE	Number of bytes per second
NDM^TKN^DATE	System date
NDM^TKN^FDBK	Feedback code
NDM^TKN^FDSN	From data set name
NDM^TKN^FILE	File name for SUBMIT
NDM^TKN^FROM^COMP^PER	Compression percentage on sending side
NDM^TKN^FUNC	Connect:Direct HP NonStop function, such as COPY and RUN TASK
NDM^TKN^INBYTE	Number of incoming bytes
NDM^TKN^INRECN	Number of incoming records
NDM^TKN^LNKFL	Link status
NDM^TKN^LU	LU name
NDM^TKN^MSGID	Connect:Direct HP NonStop message ID
NDM^TKN^NODE	Node type (PNODE or SNODE)

Token	Description
NDM^TKN^OTBYTE	Number of outgoing bytes
NDM^TKN^OTBLK	Number of outgoing blocks
NDM^TKN^OTRECN	Number of outgoing records
NDM^TKN^PC	Process control
NDM^TKN^PGM	Program name for RUN TASK
NDM^TKN^PNAME	Process name
NDM^TKN^PNUM	Process number
NDM^TKN^PRI	Priority
NDM^TKN^PROCESS	Session manager or server process name
NDM^TKN^RBYTES	Number of received bytes
NDM^TKN^RC	Return code
NDM^TKN^RSTRT	Step restart (Y|N)
NDM^TKN^RUSZ	RU size
NDM^TKN^RU^RCVD	Number of RUs received
NDM^TKN^RU^SENT	Number of RUs sent
NDM^TKN^SBYTES	Number of bytes sent
NDM^TKN^SDATE	Start date
NDM^TKN^SNODE	Secondary node name
NDM^TKN^STEPNAME	Step name
NDM^TKN^STIME	Start time
NDM^TKN^TDSN	To data set name
NDM^TKN^TIME	System time
NDM^TKN^TO^COMP^PER	Compression percentage on receiving size
NDM^TKN^TRANS	Translation (ATOE|ETOA) ATOE-ASCII to EBCDIC ETOA-EBCDIC to ASCII
NDM^TKN^UID	USERID of submitter
NDM^TKN^UNODE	User node of submitter
NDM^TKN^XNODE	Transmit node-primary or secondary (P|S)
SPL^TKN^NDM^FDBK	Feedback code
SPL^TKN^IO^PROCEDURE	System procedure call operation
SPL^TKN^IO^FILE	File name
SPL^TKN^IO^ERROR^NUM	Error number returned by the operating system
SPL^TKN^SPOOL^NUM	PERUSE job number

Token	Description
SPL^TKN^NDM^COMMAND	Connect:Direct HP NonStop command (for example, SUBMIT)
SPL^TKN^NDM^MESSAGE	Connect:Direct HP NonStop message ID
SPL^TKN^NDM^PNUM	Process number
SPL^TKN^CB^FIELD	Connect:Direct HP NonStop internal control block field name
SPL^TKN^NDMSPL^NAME	Operating system process name for Connect:Direct HP NonStop Spooler option
SPL^TKN^NDMCOM^NAME	Operating system process name for NDMCOM
NDM_TKN_IPADDR	IP address
NDM_TKN_PORTNUM	Port number

Following are event names, their description, tokens, and event numbers for all events generated by Connect:Direct HP NonStop. Event names and subject tokens are in TACL format.

Event Name	Description	Subject Token	Event Number
NDM^EVT^SESST	Session start	NDM^TKN^LU	50
NDM^EVT^PRCST	Process start	NDM^TKN^PNUM	51
NDM^EVT^STPST	Step start	NDM^TKN^PNUM	52
NDM^EVT^STPND	Step end	NDM^TKN^PNUM	53
NDM^EVT^PRCND	Process end	NDM^TKN^PNUM	54
NDM^EVT^SESND	Session end	NDM^TKN^LU	55
NDM^EVT^MSG	General message	NDM^TKN^LU	56
NDM^EVT^SUBMIT	Process submit	NDM^TKN^PNUM	57
NDM^EVT^SRVST	Server start	NDM^TKN^PROCESS	58
NDM^EVT^SRVND	Server end	NDM^TKN^PROCESS	59
NDM^EVT^SRVAB	Server ABEND	NDM^TKN^PROCESS	60
NDM^EVT^SMGST	Session manager start	NDM^TKN^PROCESS	61
NDM^EVT^SMGND	Session manager end	NDM^TKN^PROCESS	62
NDM^EVT^SMGAB	Session manager ABEND	NDM^TKN^PROCESS	63
NDM^EVT^STDST	NDMSTDL start	NDM^TKN^PROCESS	64
NDM^EVT^STDND	NDMSTDL end	NDM^TKN^PROCESS	65
NDM^EVT^STDAB	NDMSTDL ABEND	NDM^TKN^PROCESS	66
NDM^EVT^NTXST	NETEX start	NDM^TKN^PROCESS	67
NDM^EVT^NTXND	NETEX end	NDM^TKN^PROCESS	68

Event Name	Description	Subject Token	Event Number
NDM^EVT^NTXAB	NETEX ABEND	NDM^TKN^PROCESS	69
NDM^EVT^COMST	NDMCOM start	NDM^TKN^PROCESS	70
NDM^EVT^COMND	NDMCOM end	NDM^TKN^PROCESS	71
NDM^EVT^COMAB	NDMCOM ABEND	NDM^TKN^PROCESS	72
NDM^EVT^NETEX^MSG	NETEX message	NDM^TKN^LU	73
NDM^EVT^IO^ERR^MSG	Data file I/O error message	NDM^TKN^LU	74
NDM^EVT^FMH^ERR^MSG	Connect:Direct HP NonStop FMH error message	NDM^TKN^LU	75
NDM^EVT^INIT^ERR^MSG	Initialization error message	NDM^TKN^LU	76
NDM^EVT^SESS^ERR^MSG	Session error message	NDM^TKN^LU	77
NDM^EVT^TAN^ERR^MSG	HP NonStop operational error message	NDM^TKN^LU	78
NDM^EVT^NDM^ERR^MSG	Connect:Direct HP NonStop operational error message	NDM^TKN^LU	79
NDM^EVT^SEC^LIC^MSG	Security or licensing error message	NDM^TKN^LU	80
NDM^EVT^INFO^MSG	Informational message	NDM^TKN^LU	81
NDM^EVT^ALERT^MSG	Cautionary message	NDM^TKN^LU	82
NDM^EVT^BKUPREST	Backup and restore message	NDM^TKN^PNUM	83
SPL^EVT^IO^ERROR	I/O error message	SPL^TKN^IO^ PROCEDURE	84
SPL^EVT^NDM^COMMAND^ERR	Connect:Direct HP NonStop command failure message	SPL^TKN^NDM^ COMMAND	85
SPL^EVT^MISSING^CB^FIELD	Control block is missing an expected tokenized label (internal error)	SPL^TKN^CB^ FIELD	86
SPL^EVT^MISSING^CB	An expected control block is missing (internal error)	SPL^TKN^CB^ FIELD	87
SPL^EVT^SUPERVISOR^ERROR	Spooler supervisor command failure	SPL^TKN^IO^ PROCEDURE	88
SPL^EVT^START^MSG	NDMSPL startup message	SPL^TKN^NDMSPL^ NAME	89
SPL^EVT^STOP^MSG	NDMSPL termination message	SPL^TKN^NDMSPL^ NAME	90

Event Name	Description	Subject Token	Event Number
SPL^EVT^TEXT^MSG	General processing error message	SPL^TKN^NDMSPL^ NAME	91

The following table lists the tokens returned with each event. It is possible that message events only return a subset of the listed tokens.

Event	Token
NDM^EVT^SESST	NDM^TKN^SNODE NDM^TKN^CLASS NDM^TKN^LU NDM^TKN^NODE NDM_TKN_IPADDR NDM_TKN_PORTNUM
NDM^EVT^PRCST	NDM^TKN^CLASS NDM^TKN^LU NDM^TKN^UNODE NDM^TKN^UID NDM^TKN^PNAME NDM^TKN^PNUM NDM^TKN^SNODE NDM^TKN^XNODE NDM_TKN_IPADDR NDM_TKN_PORTNUM NDM^TKN^PC
NDM^EVT^STPST	NDM^TKN^PC NDM^TKN^FUNC NDM^TKN^STEPNAME NDM^TKN^UNODE NDM^TKN^UID NDM^TKN^PNAME NDM^TKN^PNUM NDM^TKN^SNODE NDM^TKN^FDSN NDM^TKN^TDSN NDM^TKN^XNODE

Event	Token
NDM^EVT^STPND	NDM^TKN^UNODE NDM^TKN^UID NDM^TKN^PNAME NDM^TKN^PNUM NDM^TKN^SNODE NDM^TKN^TRANS NDM^TKN^XNODE NDM^TKN^CTIME NDM^TKN^CDATE NDM^TKN^STIME NDM^TKN^SDATE NDM^TKN^FDBK NDM^TKN^RC NDM^TKN^MSGID NDM^TKN^STEPNAME NDM^TKN^RUSZ NDM^TKN^INBYTE NDM^TKN^INRECN NDM^TKN^RU^SENT NDM^TKN^RU^RCVD NDM^TKN^SBYTES NDM^TKN^FDSN NDM^TKN^TDSN NDM^TKN^COMPRESS NDM^TKN^OTBYTE NDM^TKN^OTRECN NDM^TKN^RBYTES NDM^TKN^FROM^COMP^PER NDM^TKN^TO^COMP^PER
NDM^EVT^PRCND	NDM^TKN^UNODE NDM^TKN^UID NDM^TKN^PNAME NDM^TKN^PNUM NDM^TKN^SNODE NDM^TKN^XNODE
NDM^EVT^SESND	NDM^TKN^SNODE NDM^TKN^CLASS NDM^TKN^LU NDM^TKN^NODE NDM_TKN_IPADDR NDM_TKN_PORTNUM
NDM^EVT^MSG	ZEMS^TKN^TEXT NDM^TKN^LU NDM^TKN^SNODE NDM^TKN^UNODE NDM^TKN^UID NDM^TKN^PNUM NDM^TKN^PNAME
NDM^EVT^SUBMIT	NDM^TKN^PNUM NDM^TKN^PNAME NDM^TKN^UNODE NDM^TKN^UID NDM^TKN^RC NDM^TKN^FDBK NDM^TKN^FILE

Event	Token
NDM^EVT^SRVST	NDM^TKN^PROCESS ZEMS^TKN^TEXT
NDM^EVT^SRVND	NDM^TKN^PROCESS ZEMS^TKN^TEXT
NDM^EVT^SRVAB	NDM^TKN^PROCESS ZEMS^TKN^TEXT
NDM^EVT^SMGST	NDM^TKN^PROCESS ZEMS^TKN^TEXT
NDM^EVT^SMGND	NDM^TKN^PROCESS ZEMS^TKN^TEXT
NDM^EVT^SMGAB	NDM^TKN^PROCESS ZEMS^TKN^TEXT
NDM^EVT^BKUPREST	NDM^TKN^PNUM ZEMS^TKN^TEXT
SPL^EVT^IO^ERROR	SPL^TKN^IO^PROCEDURE SPL^TKN^IO^FILE SPL^TKN^IO^ERROR^NUM
SPL^EVT^NDM^COMMAND^ERR	SPL^TKN^NDM^COMMAND SPL^TKN^NDM^MESSAGE NDM^TKN^RC SPL^TKN^NDM^FDBK SPL^TKN^SPOOL^NUM SPL^TKN^NDM^PNUM
SPL^EVT^MISSING^CB^FIELD	SPL^TKN^CB^FIELD
SPL^EVT^MISSING^CB	SPL^TKN^CB^FIELD
SPL^EVT^SUPERVISOR^ERR	SPL^TKN^IO^PROCEDURE SPL^TKN^IO^ERROR^NUM SPL^TKN^SPOOL^NUM
SPL^EVT^START^MSG	SPL^TKN^NDMSPL^NAME SPL^TKN^NDMCOM^NAME
SPL^EVT^STOP^MSG	SPL^TKN^NDMSPL^NAME SPL^TKN^NDMCOM^NAME
SPL^EVT^TEXT^MSG	SPL^TKN^NDMSPL^NAME

Glossary

A

Adjacent Node

An adjacent node is an entry in the Network Map that defines a Connect:Direct HP NonStop node with which the local Connect:Direct HP NonStop node can communicate. The adjacent node is also called a remote node.

AIMS

The automated installation and management system (AIMS) is a menu-driven system that guides you through the installation procedure for Connect:Direct HP NonStop.

Application Programming Interface (API)

The Application Programming Interface (API) is a Connect:Direct HP NonStop component that accepts commands and places them in an executable format.

API Manager

An API manager is a network map entity, that handles communications sessions between Connect:Direct HP NonStop and external applications on a TCP/IP network. After the API manager has been set up, users of these other Sterling Commerce products can configure, control, and operate Connect:Direct HP NonStop from any host on a TCP/IP network.

AUTHFILE

The authorization file contains records of user attribute defaults. Each record defines the features of Connect:Direct HP NonStop that you can access.

B

Background Mode

The background mode enables you to execute NDMCOM using a disk file containing Connect:Direct HP NonStop commands as input. All Connect:Direct HP NonStop commands, except the FC command, are used in this mode.

C

CB Function

The CB (Control Block) function is a group of statements that performs a specific task and often returns a value to the statement that calls it.

C-string Control Block

The C-string control block (CB) is the data format that returns output generated by Connect:Direct HP NonStop Processes and commands to the API. A C-string control block consists of two or more fields.

C-string Control Structure

The C-string control structure groups one or more related C-string control blocks.

Checkpoint Restart

The checkpoint restart feature eliminates the need to retransmit an entire file in the event of a transmission failure. If a copy procedure is interrupted, Connect:Direct HP NonStop restarts that copy at the last checkpoint.

Command Line Interface

The command line interface is a Connect:Direct HP NonStop interface that enables you to submit Connect:Direct HP NonStop Processes and commands from your native command line environment.

Commands

Connect:Direct HP NonStop commands initiate and monitor activity within the Connect:Direct HP NonStop system.

Connect:Direct HP NonStop Commands

Connect:Direct HP NonStop commands use a command structure common to the rest of the Connect:Direct family of products. The commands are issued three ways: in interactive mode directly from the command line, in background mode by issuing the Connect:Direct HP NonStop OBEY command, or through an API.

Connect:Direct HP NonStop Spooler Option

The Connect:Direct HP NonStop spooler option is a Connect:Direct HP NonStop application that permits an installation to transfer output spooler jobs automatically from a Connect:Direct HP NonStop node to a disk file on an adjacent node.

Connect:Direct/Plex

Connect:Direct/Plex is a Connect:Direct OS/390 (zOS) system consisting of a Connect:Direct/Manager and one or more Connect:Direct/Servers in a TCP/IP environment. Connect:Direct HP NonStop can establish sessions with Connect:Direct/Plex.

Cyclic Redundancy Checking (CRC)

CRC is a method used to validate data integrity during data transfers between Connect:Direct nodes across a TCP/IP network. CRC can be controlled using any of the following options:

- A global initialization parameter
- An adjacent node definition
- A Process statement parameter
- A SUBMIT command parameter

D

Domain Server

Connect:Direct HP NonStop can be configured to handle inbound connection requests from a TCP domain, that is, a range of IP addresses, using the ADJ NODE record type NDM.DOMAIN. This allows the application to recognize connection requests from IP addresses that are not explicitly configured in the network map, as long as they fall within one of the defined domains.

Downstream Connection

See Receiving Connection.

Dynamic LUs

Connect:Direct HP NonStop starts dynamic LUs as needed and automatically stops them upon Process completion. Dynamic LUs are options when using TCP/IP.

E

EMS Filters

The EMS filters provide a programmatic method for selecting events for processing.

Environment Commands

These commands enable you to perform various Connect:Direct HP NonStop functions, such as displaying environment values and invoking TEDIT. Some environment commands allow you to set specific environment parameter values in NDMCOM. These values remain in effect only for the duration of the current session, unless they are changed by you or another user logs on to the same NDMCOM.

ERR Control Block

The ERR control block is the first control block of an error control structure (ERRCS). The beginning and ending fields are: CB ! ERR ! and CBEND ! ERR !. The two other required fields in the ERR control block are: *N* (number) field and *T* (top message) field. *N* specifies the number of messages in the ERRCS; *T* specifies the number of the most important message.

Error Control Structure (ERRCS)

The error control structure (ERRCS) is a particular C-string control structure designed to identify the messages occurring when executing Connect:Direct HP NonStop Processes and commands.

Event Management Service (EMS)

Event management performs event-collection, logging, and distribution in the distributed systems management (DSM) environment.

F

FASTLOAD

This Connect:Direct HP NonStop function can reduce disk I/O overhead. It is used when the Connect:Direct HP NonStop node is the destination. With FASTLOAD, Connect:Direct HP NonStop passes data through SPI to FUP to load into a destination data file. The feature is particularly useful for key-sequenced files, but it is also supported for entry-sequenced and relative record files.

Field

A field is two null-terminated strings—key and data. Two or more fields make up a C-string control block.

I

I/O Exit Support

This support provides exit points for user-written programs to serve as application interfaces for data transfers.

Interactive Mode

This mode enables you to issue commands through NDMCOM and receive an immediate response.

L

Local Node

The local node is the Connect:Direct HP NonStop server.

M

Message Commands

The message commands allow you to display, add, delete, modify, and print Connect:Direct HP NonStop messages from the command interpreter (TACL).

Message Control Blocks

Message control blocks are part of an ERRCS. These blocks are sequenced as they occur. The fields in a message control block are CB ! En !, FDBK ! fb !, RC ! rc !, MSGID ! msgid !, and OK ! od ! (optional keyword ! optional data !).

N

NDMCOM

NDMCOM is the Connect:Direct HP NonStop user interface.

NDMMON

The monitor Process (NDMMON) ensures nonstop operation of Connect:Direct HP NonStop.

NDMSTDL

The statistics deletion program (NDMSTDL) ensures sufficient space is available to write statistics records in the statistics files. It deletes records from STATFILE and STATSRCH based on user-specified deletion criteria and maximum percentage of file capacity.

NETEX Option

NETEX is a connection option for Connect:Direct OS/390.

Network Map

The network map (netmap) is a file that identifies all valid Connect:Direct nodes in the network. One network map is associated with each Connect:Direct HP NonStop local node. The netmap has one entry for each of the other Connect:Direct nodes to which the local Connect:Direct HP NonStop node communicates. The netmap entries also contain the rules or protocol that the nodes adhere to when communicating.

Node

A node is any site in a network from which information distribution is initiated.

P

Primary Node

The primary node (PNODE) is the Connect:Direct HP NonStop node on which the Process is submitted. The primary node is also referred to as the controlling node or initiating node, but is not necessarily interpreted as the sending node, because PNODE can be the receiver. In every Process, one PNODE and one SNODE are specified. The submitter of a Process is always the PNODE.

PNODE=SNODE Transmission

This transmission enables you to create a Process to send data to another file on your node. In this type of transmission, your node is both the PNODE and the SNODE.

Primary Logical Unit

The primary logical unit (PLU) is the logical unit that controls an LU to LU session. The PLU formats and sends an NLD request that begins a session.

Process (Source File)

A Process is a series of statements that initiate Connect:Direct activity, such as copying files, running jobs, and so on.

Process Statements

Process statements are instructions for transferring files, running operating system jobs, executing programs, or submitting other Connect:Direct HP NonStop Processes. You use Process statements to build a Connect:Direct HP NonStop Process.

R

Receiving Connection

The receiving connection is a connection between Connect:Direct HP NonStop and other nodes (AS/400—TCP only) where the Connect:Direct HP NonStop node supports the primary functions of the data link and the HP NonStop LU functions as a primary LU (PLU).

Remote Node

A remote node is an entry in the network map that defines a Connect:Direct node with which the local Connect:Direct HP NonStop node can communicate. The remote node is also called an adjacent node.

Retry Interval

The retry interval is the interval at which retries are performed as a part of the checkpoint-restart feature.

S

SECFILE

The security file (SECFILE) relates the node name and user ID assigned to an incoming Connect:Direct HP NonStop operation to a HP NonStop user ID.

Secondary Logical Unit

The secondary logical unit (SLU) is the logical unit that functions under the control of a PLU. The SLU accepts the incoming NLD request from the PLU.

Secondary Node

The secondary node (SNODE) is the Connect:Direct HP NonStop node that interacts with the primary node (PNODE) during Process execution. SNODE is also referred to as the participating (non controlling) or partner node. Every Process has one PNODE and one SNODE.

Secure Point of Entry

The secure point of entry enables Processes from other nodes to be written without the use of passwords.

Sending Connection

The sending connection is between HP NonStop and the IBM 370 nodes (OS/390, VM, VSE) where the IBM node supports the primary functions of the data link and the IBM LU functions as a primary LU (PLU).

Server

The server (NDMSRVR) is responsible for processing command requests, communicating with the session manager when work is placed in the transmission control queue, and accepting session establishment requests from remote nodes.

Session Manager

The session manager (NDMSMGR) is responsible for establishing communication sessions, performing standard session management functions, and executing Processes.

SNA (Systems Network Architecture)

A network architecture designed to provide compatibility among a wide variety of hardware and software products that enable you to build complex networks. It defines protocols, standards, and message formats to which different hardware and software products must conform.

SNA Primary

SNA primary defines the LU as a primary LU (PLU).

SNA Secondary

SNA secondary defines the LU as a secondary LU (SLU).

SNAX Passthrough

SNAX passthrough is a function of the SNAX line access software that permits interaction between a host application program and an SNA device connected to a HP NonStop system. The Connect:Direct HP NonStop system, which is not a Connect:Direct HP NonStop node, appears to the host as a cluster controller.

SNODE

The secondary node (SNODE) is the node participating in Process execution initiated by another node (the PNODE).

Statistics File

The statistics file holds Connect:Direct HP NonStop statistics records that document the history of a Process.

Statistics Facility

The Connect:Direct HP NonStop statistics facility records Connect:Direct HP NonStop activities.

Static LUs

Static LUs are user-controlled and are quiesced and resumed with the MODIFY command. Static LUs are options when using TCP/IP.

T

Transmission Control Queue

The Transmission Control Queue (TCQ) holds information about Connect:Direct HP NonStop Processes that are currently executing or scheduled to execute in the future.

TCP/IP Option

TCP/IP is a connectivity option for Connect:Direct OS/390, UNIX, OpenVMS, VSE, OS/400, Stratus VOS, and Windows, and HP NonStop nodes.

Type File

The type file contains records that define file attributes for new files.

U

Upstream Connection

See Sending Connection.

Index

Symbols

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V