# Connect:Direct®
# for UNIX

## Administration Guide

**Version 3.8**

Sterling Commerce
An IBM Company

***Connect:Direct for UNIX Administration Guide***
**Version 3.8**

**First Edition**

Sterling Commerce, Inc.

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

CDUNXAG611

# Contents

Chapter 1

# About Connect:Direct for UNIX

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

Connect:Direct is based on client-server architecture. The Connect:Direct server components interact with the user interfaces (API, CLI, Connect:Direct Browser User Interface, and Sterling Control Center) to enable you to submit, execute, and monitor Connect:Direct statements and commands.

## Server Components

Connect:Direct has the following server components:

### Process Manager

The Process Manager (PMGR) is the daemon that initializes the Connect:Direct server environment. The PMGR provides the following functions:

- Initializes Connect:Direct
- Accepts connection requests from Connect:Direct client APIs and remote nodes
- Creates Command Manager and Session Manager child Processes to communicate with APIs and remote nodes
- Accepts requests from Command Managers and Session Managers when centralized Connect:Direct functions are required
- Stops Connect:Direct

**Note:** Any application, including End User Applications (EUA), can run on any computer as long as it can connect to the PMGR.

## Command Manager

A Command Manager (CMGR) is created for every API connection that is successfully established. The number of Command Managers that a PMGR can create is system-dependent and limited by the number of file descriptors available for each UNIX Process. The number of file descriptors set up by the UNIX operating system may affect Connect:Direct operation. You must define enough file descriptors to handle the number of concurrent Connect:Direct sessions allowed, which can be as many as 999.

The CMGR provides the following functions:

- Executes commands sent by the API and sends the results back to the API
- Carries out the Connect:Direct authentication procedure, in conjunction with the API, to determine access to Connect:Direct
- Interacts with the PMGR when executing commands

## Session Manager

The Session Manager (SMGR) is created and invoked by the PMGR when resources are available and either a Process is ready to run or a remote node requests a connection with a local node. The SMGR provides the following functions:

- Performs the necessary Connect:Direct work
- Acts as a primary node (PNODE) and initiates Process execution
- Acts as a secondary node (SNODE) to participate in a Process initiated by the PNODE

When an SMGR is created to execute a Process submitted to a node, it creates the connection to the remote node. If the SMGR is started by the PMGR to execute local Processes, the SMGR runs each Process on this session until all Processes are completed.

If an SMGR is created because a remote node initiated a connection, the SMGR completes the connection. If the SMGR is started by the PMGR to execute remote Processes, the SMGR executes remote Process steps supplied by the remote SMGR until the remote SMGR completes all of its Processes.

The SMGR depends on the PMGR for Transmission Control Queue (TCQ) services and other centralized services. Refer to the *Transmission Control Queue* on page 13 for an overview of the TCQ.

## File Agent

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

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

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

To achieve optimum performance, configure File Agent to communicate with the Connect:Direct node where it is installed. File Agent can be installed on UNIX, Windows, and z/OS operating systems. For information to help you plan how to implement File Agent, see the *Managing Files with Connect:Direct File Agent* chapter in your Connect:Direct administration guide or getting started guide. The Connect:Direct File Agent Help contains instructions for configuring File Agent.

### Connect:Direct Secure+ Option for UNIX

The Connect:Direct Secure+ Option application provides enhanced security for Connect:Direct and is available as a separate component. It uses cryptography to secure data during transmission. You select the security protocol to use with Secure+ Option.

To use Connect:Direct Secure+ Option for communications with remote nodes, you must have node records in the Secure+ Option parameters file that duplicate the adjacent node records in the Connect:Direct network map. You can populate the Secure+ Option parameters file from entries defined in an existing network map. For more information about creating the Connect:Direct Secure+ Option parameters file and configuring nodes for Secure+ Option, refer to the *Connect:Direct Secure+ Option for UNIX Implementation Guide*.

## User Interfaces

Connect:Direct has the following user interfaces, which enable you to create, submit, and monitor Processes.

### Applications Programming Interface

The UNIX Applications Programming Interface (API) enables you to write programs that work with Connect:Direct. Four API functions are provided to allow an End User Application (EUA) to perform the following tasks:

- ✦ Establish an API connection to the Connect:Direct server
- ✦ Terminate an API connection to the Connect:Direct server
- ✦ Send a command to Connect:Direct
- ✦ Receive responses from commands

### Command Line Interface

The Command Line Interface (CLI) enables you to perform the following tasks:

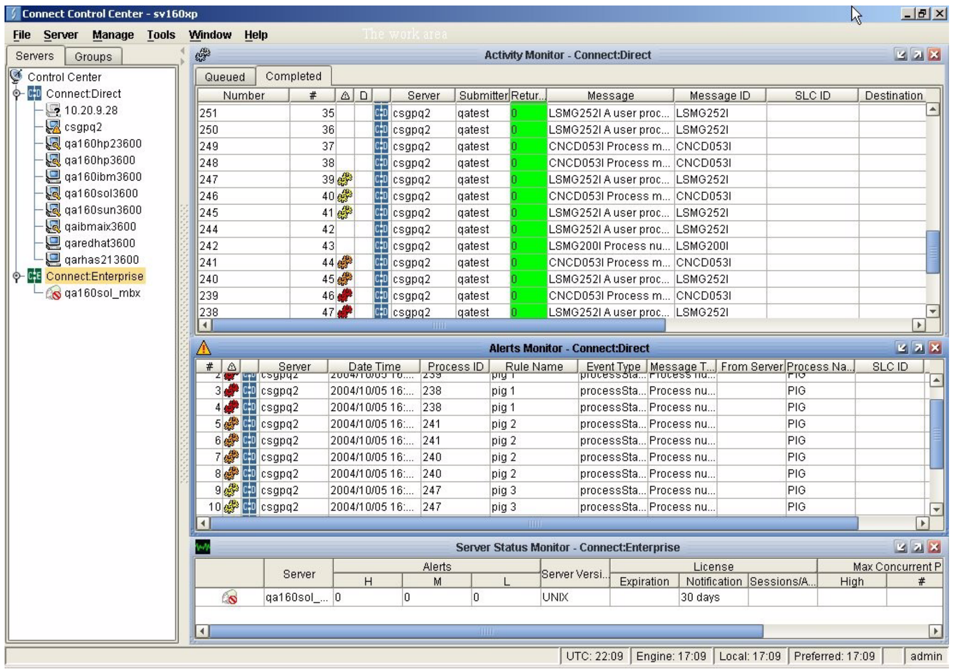
- ✦ Issue Connect:Direct commands
- ✦ Monitor Processes

The CLI command prompt is **Direct >**. Refer to Chapter 2, *Controlling and Monitoring Processes* in the *Connect:Direct for UNIX User's Guide* for additional information about the CLI.

## Sterling Control Center

Sterling Control Center is a centralized management system that provides operations personnel with continuous enterprise-wide business activity monitoring capabilities for Connect:Direct for z/OS, Connect:Direct for UNIX, Connect:Direct for Windows, Connect:Direct for HP NonStop, Connect:Direct Select, and Connect:Direct OS/400 (iSeries) servers, and Connect:Enterprise for UNIX and Connect:Enterprise for z/OS servers. Sterling Control Center enables you to:

- Manage multiple servers
  - Group individual servers into server groups for a single view of system-wide activity
  - View status and statistics on active or completed processing
  - Suspend, release, stop, and delete Connect:Direct Processes on z/OS, UNIX, Windows, Select, and HP NonStop platforms
  - Stop Connect:Direct servers on z/OS, Windows, HP NonStop, and UNIX platforms.
- Monitor service levels
  - View processing across Connect:Direct for z/OS, UNIX, Select, Windows, HP NonStop, and OS/400 (iSeries) servers, and Connect:Enterprise for UNIX and Connect:Enterprise for z/OS servers within your network and retrieve information about active and completed processing



| Number | # | ⚠ | D | | Server | Submitter | Retur... | Message | Message ID | SLC ID | Destination |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 251 | 35 | | | C:D | csgpq2 | qatest | 0 | LSMG252I A user proc... | LSMG252I | | |
| 250 | 36 | | | C:D | csgpq2 | qatest | 0 | LSMG252I A user proc... | LSMG252I | | |
| 249 | 37 | | | C:D | csgpq2 | qatest | 0 | CNCD053I Process m... | CNCD053I | | |
| 248 | 38 | | | C:D | csgpq2 | qatest | 0 | CNCD053I Process m... | CNCD053I | | |
| 247 | 39 | ⚙ | | C:D | csgpq2 | qatest | 0 | LSMG252I A user proc... | LSMG252I | | |
| 246 | 40 | ⚙ | | C:D | csgpq2 | qatest | 0 | CNCD053I Process m... | CNCD053I | | |
| 245 | 41 | ⚙ | | C:D | csgpq2 | qatest | 0 | LSMG252I A user proc... | LSMG252I | | |
| 244 | 42 | | | C:D | csgpq2 | qatest | 0 | LSMG252I A user proc... | LSMG252I | | |
| 242 | 43 | | | C:D | csgpq2 | qatest | 0 | LSMG200I Process nu... | LSMG200I | | |
| 241 | 44 | ⚙ | | C:D | csgpq2 | qatest | 0 | CNCD053I Process m... | CNCD053I | | |
| 240 | 45 | ⚙ | | C:D | csgpq2 | qatest | 0 | LSMG252I A user proc... | LSMG252I | | |
| 239 | 46 | ⚙ | | C:D | csgpq2 | qatest | 0 | CNCD053I Process m... | CNCD053I | | |
| 238 | 47 | ⚙ | | C:D | csgpq2 | qatest | 0 | LSMG252I A user proc... | LSMG252I | | |

Alerts Monitor - Connect:Direct

| # | ⚠ | | Server | Date Time | Process ID | Rule Name | Event Type | Message T... | From Server | Process Na... | SLC ID |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 239 | pig 1 | processSta... | Process nu... | | PIG | |
| 3 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 238 | pig 1 | processSta... | Process nu... | | PIG | |
| 4 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 238 | pig 1 | processSta... | Process nu... | | PIG | |
| 5 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 241 | pig 2 | processSta... | Process nu... | | PIG | |
| 6 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 241 | pig 2 | processSta... | Process nu... | | PIG | |
| 7 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 240 | pig 2 | processSta... | Process nu... | | PIG | |
| 8 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 240 | pig 2 | processSta... | Process nu... | | PIG | |
| 9 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 247 | pig 3 | processSta... | Process nu... | | PIG | |
| 10 | ⚙ | C:D | csgpq2 | 2004/10/05 16:... | 247 | pig 3 | processSta... | Process nu... | | PIG | |

Server Status Monitor - Connect:Enterprise

| | Server | Alerts | | | Server Versi... | License | | | Max Concurrent P | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | H | M | L | | Expiration | Notification | Sessions/A... | High | # |
| | qa160sol_... | 0 | 0 | 0 | UNIX | | 30 days | | | |

UTC: 22:09 | Engine: 17:09 | Local: 17:09 | Preferred: 17:09 | admin

  - Receive notification of data delivery events that occur or do not occur as scheduled
  - Define rules that, based on processing criteria, can generate an alert, send an e-mail notification, generate a Simple Network Management Protocol (SNMP) trap to an Enterprise Management System (EMS), run a system command, or issue a server command

- Monitor for alerts, such as a server failure or a Process not starting on time
- Create service level criteria (SLCs) that define processing schedules, monitor Processes, files within Processes, and file transfers for compliance with these schedules, and generate alerts when the schedules are not met

- Analyze key operational metrics
- Create customized reports to document and analyze processing activity based on criteria you define
- Validate the authenticity of a user logging on to Sterling Control Center, using one or more of four authentication methods including password validation, host name identification, Windows domain, and TCP/IP address
- Identify additional servers that may need to be monitored based on communications with a currently monitored server using the Guided Node Discovery feature

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

- Ensuring that critical processing windows are met
- Reducing impact on downstream processing by verifying that expected processing occurs
- Providing proactive notification for at-risk business processes
- Consolidating information for throughput analysis, capacity planning, post-processing operational or security audits, and workload analysis
- Reducing the risk of errors associated with manual system administration, including eliminating individual server logon to view activity, and the need to separately configure each server for error and exception notifications

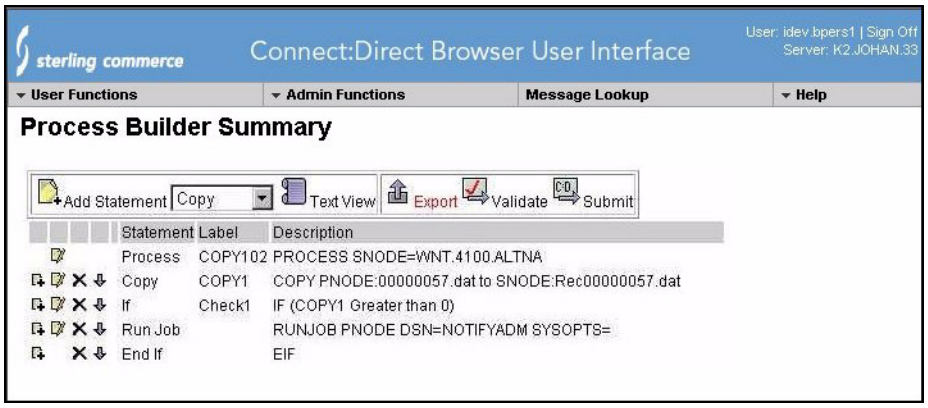
Sterling Control Center is available for purchase as a separate product. Contact your Sterling Commerce representative to learn more about Sterling Control Center.

## Connect:Direct Browser User Interface

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

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

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



Process Builder Summary

| Statement | Label | Description |
|---|---|---|
| Process | COPY102 | PROCESS SNODE=WNT.4100.ALTNA |
| Copy | COPY1 | COPY PNODE:00000057.dat to SNODE:Rec00000057.dat |
| If | Check1 | IF (COPY1 Greater than 0) |
| Run Job | | RUNJOB PNODE DSN=NOTIFYADM SYSOPTS= |
| End If | | EIF |

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

## Connect:Direct Concepts

This section introduces concepts and definitions to help you understand system operations.

### Local and Remote Nodes

Each data transfer involves a local and a remote node. The Connect:Direct server is the local node and the partner node is the remote node. The two servers (local and remote) function together to perform the work. Either Connect:Direct node can initiate the work.

Local and remote node connections are set up in the network map file. Refer to Chapter 5, *Maintaining the Network Map File*, in the *Connect:Direct for UNIX Administration Guide* for a description of the network map file.

Connect:Direct must be installed on each node. When Connect:Direct establishes a session between the local node and remote node, the node that initiates the session has primary control (PNODE). The other serves as the partner and has a secondary function (SNODE). The node that initiates the session has primary control, regardless of the direction of information flow. The Process can specify work destined for either the local or remote node. When Connect:Direct establishes a session, the two nodes work together to transfer the information.

### Processes

The Connect:Direct 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.

The following table lists Process statements and their functions:

| Process Statement | Function |
|---|---|
| copy | Copies files from one node to another. |
| conditionals | Alters the sequence of Process execution based on the completion code of previous steps with the **if**, **then**, **else**, **eif** (end if), **goto**, and **exit** statements. |
| process | Defines general Process characteristics. |
| run job | Enables you to specify UNIX commands in a Process. The Process does not wait until the job has finished running before executing the next step in the Process. |
| run task | Enables you to specify UNIX commands in a Process. The Process waits until the job has finished running before executing the next step in the Process. |
| submit | Starts another Process to either the local or remote node during execution of a Process. |
| pend | Marks the end of a Connect:Direct for UNIX Process. |

Following is a sample Process:

```
ckpt01  process  snode=unix.node
step01 copy from (
             file=file1
             snode
             )
         ckpt=1M
         to   (
             file=file2
             disp=new
             pnode
             )
    pend;
```

Refer to the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html for instructions on building a Process.

## Transmission Control Queue

The Transmission Control Queue (TCQ) controls when Processes run. Connect:Direct stores submitted Processes in the TCQ. The TCQ is divided into four logical queues: Execution, Wait, Timer, and Hold. Processes are run from the Execution queue.

Connect:Direct places a Process in the appropriate queue based on Process statement parameters, such as the **hold**, **retain**, and **startt** parameters.

Connect:Direct runs Processes based on their priority and when the Process is placed in the Execution queue. Higher priority Processes are selected for execution ahead of Processes with a lower priority. You can access the queues and manage the Processes through Connect:Direct commands.

Refer to the *Connect:Direct for UNIX User's Guide* for more information on scheduling Processes in the TCQ.

## Commands

You use commands to submit Processes to the TCQ. You can also use commands to perform the following tasks:

- Manage Processes in the queue
- Monitor and trace Process execution
- Produce reports on Process activities
- Stop Connect:Direct operation

The following table lists the commands and their functions:

| Command | Function |
|---|---|
| change process | Changes the status and modifies specific characteristics of a nonexecuting Process in the TCQ. |
| delete process | Removes a nonexecuting Process from the TCQ. |
| flush process | Removes an executing Process from the TCQ. |
| trace | Runs Connect:Direct traces. |
| select process | Displays or prints information about a Process in the TCQ. |
| select statistics | Displays or prints statistics in the statistics log. |
| stop | Stops Connect:Direct operation. |
| submit | Submits a Process for execution. |

For example, the following command submits the Process called **onestep** to the TCQ with a hold status of yes:

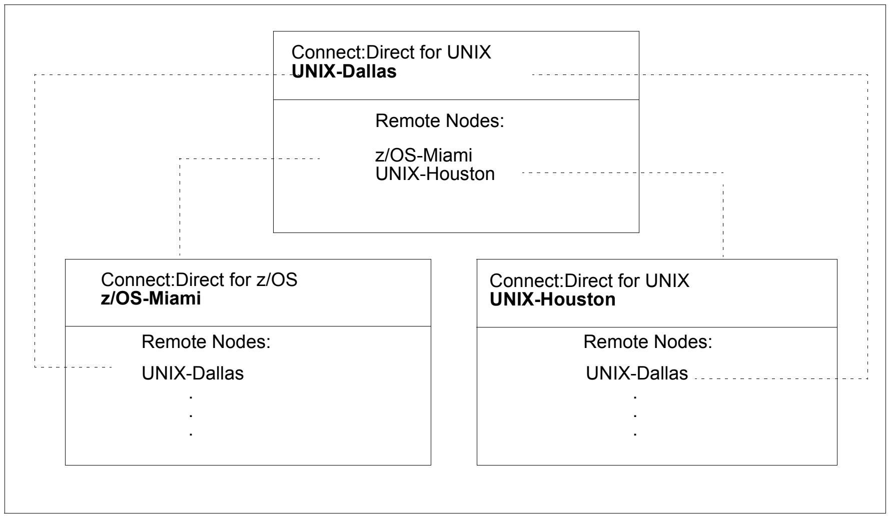
```
Direct> submit file=onestep hold=yes;
```

## Network Map

During the transfer of data, the Connect:Direct server where the Process is submitted is the primary node and the secondary node is the partner node. Connect:Direct identifies the remote nodes that the local node is able to communicate with through the use Connect:Direct network map (or netmap).

The network map includes the names of all the remote nodes that the Connect:Direct local node can communicate with, the paths to contact those remote nodes, and characteristics of the sessions for communication.

The remote Connect:Direct nodes also have network maps for their remote nodes. The following sample displays the corresponding network map entries of UNIX-Dallas, z/OS-Miami, and UNIX-Houston:



## User Authorization

Connect:Direct can authorize local and remote users to perform certain Connect:Direct tasks. In order to use Connect:Direct, each user must have a record defined in the user authorization file, called userfile.cfg. Each local user must have a record in the user authorization file, and each remote user must be mapped to a local user ID in a proxy relationship.

To provide a method of preventing an ordinary user from gaining root access through Connect:Direct for UNIX, a second access file called the Strong Access Control file (SACL) is created when you install Connect:Direct for UNIX and is named sysacl.cfg. Creating an SACL file improves the security of b and prevents a user from obtaining root access control. At installation, the default value for the root:deny.access parameter is y (yes). If you want to allow root access, you must access the sysacl.cfg and specify root:deny.access=n. If the file is deleted or corrupted, access to Connect:Direct is denied to all users.

The Connect:Direct administrator maintains these authorizations. Refer to Chapter 6, *Maintaining Access Information Files* in the *Connect:Direct for UNIX Administration Guide* for more information about authorizations.

## Process Restart

Several facilities are provided for Process recovery after a system malfunction. The purpose of Process recovery is to resume execution as quickly as possible and to minimize redundant data transmission after a system failure. The following Connect:Direct facilities are available to enable Process recovery:

- Process step restart—As a Process runs, the steps are recorded in the TCQ. If a Process is interrupted for any reason, the Process is held in the TCQ. When you release the Process to continue running, the Process automatically begins at the step where it halted.
- Automatic session retry—Two sets of connection retry parameters are defined in the remote node information record of the network map file: short-term and long-term. If you do not specify a value for these parameters in the remote node information record, default values are used from the local.node entry of the network map file. The short-term parameters allow immediate retry attempts. Long-term parameters are used after all short-term retries are attempted. Long-term attempts assume that the connection problem cannot be fixed quickly and retry attempts occur after a longer time period, thus saving the overhead of connection retry attempts.
- Checkpoint restart—This feature is available with the **copy** statement.

  Checkpoint restart can be explicitly configured within a **copy** step through the **ckpt** parameter. (Refer to the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html) If it is not configured in the **copy** step, it can be configured in the Initparms through the **ckpt.interval** parameter. (See the *Connect:Direct for UNIX Administration Guide* for more information on this parameter.)
- Run Task restart—This feature is used if the PNODE and SNODE cannot resynchronize during a restart after a run task interruption. If a Process is interrupted when a run task on an SNODE step is executing, Connect:Direct attempts to synchronize the previous run task step on the SNODE with the current run task step. Synchronization occurs in one of the following ways:
  - If the SNODE is executing the task when the Process is restarted, it waits for the task to complete, and then responds to the PNODE with the task completion status. Processing continues.
  - If the SNODE task completes before the Process is restarted, it saves the task results. When the Process is restarted, the SNODE reports the results, and processing continues.

  If synchronization fails, Connect:Direct reads the **restart** parameter in the **run task** step or the initialization parameters file to determine whether to perform the **run task step** again. The **restart** parameter on the **run task** step overrides the setting in the initialization parameter.

  For example, if the SNODE loses the run task step results due to a Connect:Direct cold restart, Connect:Direct checks the value defined in the **restart** parameter to determine whether to perform the **run task** again.

**Note:** Run task restart works differently when Connect:Direct for UNIX runs behind a connection load balancer. For more information on the considerations you need to know when running Connect:Direct for UNIX in a load balancing environment, see the *Connect:Direct for UNIX Release Notes* and the *Connect:Direct for UNIX Administration Guide*.

✦ Interruption of Process activity when the SNODE is a Connect:Direct for UNIX node—When the SNODE is a Connect:Direct for UNIX node and the PNODE interrupts Process activity by issuing a command to suspend Process activity, deleting an executing Process, or when a link fails or an I/O error occurs during a transfer, the Process is placed in the Wait queue in WS status.

If Process activity does not continue, you must manually delete the Process from the TCQ. Refer to the *Connect:Direct for UNIX User's Guide* for command syntax and parameter descriptions for the **delete process** command.

**Note:** You cannot issue a **change process** command from the SNODE to continue Process activity; the Process can only be restarted by the PNODE, which is always in control of the session.

## Archive Statistics Files

Connect:Direct for UNIX provides a utility to archive and purge statistics files. When you configure Connect:Direct for UNIX, you identify when to archive a statistics file by setting the parameter, **max.age**, in the stats record of the initialization parameters file. The **max.age** parameter defines how old a statistics file must be before you want to archive the file.

Once a day, the script called statarch.sh is started. This script identifies the statistics files that are equal to the **max.age**. It then runs the tar command and the compress command to create a compressed archived file of all the statistics records that match the **max.age** parameter. Once the statistics files are archived, these files are purged. For files archived on a Linux computer, the archived statistics files have the .gz suffix since these files are compressed with the gzip format. Archived files on all other UNIX platforms have the .Z suffix to indicate they are compressed using the compress format.

The archived files are stored in the directory where the statistics files and TCQ are stored. The shell script, statarch.sh, is located in the ndm/bin directory. If necessary, modify the script to customize it for your environment.

If you want to restore statistics files that have been archived, run the **statrestore.sh** script. It uses the **uncompress** and **tar** commands to restore all the statistics files in the archive. You supply two arguments to the **statrestore** command. The first argument is the directory path where the statistics files are located and the second argument identifies the archived file name followed by as many archived file names as you want to restore. Below is a sample **statrestore** command:

```
qa160sol: ./statrestore.sh /export/home/users/cd3800/ndm/bin archive1
```

After files are restored, the statistics records can be viewed using the select statistics command.

## Sample Processes, Shell Scripts, and API Processes

Connect:Direct provides sample Processes and shell scripts in *d_dir*/ndm/src, where *d_dir* indicates the destination directory of the Connect:Direct software. You can create similar files with a text editor. In addition, instructions for creating shell scripts are in the README file in the same directory.

The following table displays the file names of sample Processes. Modify the Processes as required.

| File Name | Type of Process |
|---|---|
| cpunx.cd | copy |
| rtunx.cd | run task |
| rjunx.cd | run job |
| sbunx.cd | submit |

The following table displays the file names of sample shell scripts. Modify the shell scripts as required.

| File Name | Type of Shell Script |
|---|---|
| selstat.sh | select statistics |
| send.sh | send |
| recv.sh | receive |
| wildcard | send multiple files to an OS/390 PDS |
| statarch.sh | archive statistics files |
| statrestore.sh | restore statistics files that have been archived |
| lcu.sh | launch the Local Connection Utility tool |
| spadmin.sh | launch the Secure+ Option Admin Tool |
| spcli.sh | launch the Secure+ Option CLI (SPCLI) |
| spcust_sample1.sh | configure Secure+ Option for the STS protocol |
| spcust_sample2.sh | configure Secure+ Option for the STS protocol |
| spcust_sample3.sh | configure Secure+ Option to use the SSL or TLS protocol |

## Connect:Direct for UNIX Files

This section describes the configuration files, illustrates the directory structure of Connect:Direct for UNIX, and lists the individual files that are installed.

### Connect:Direct for UNIX Configuration Files

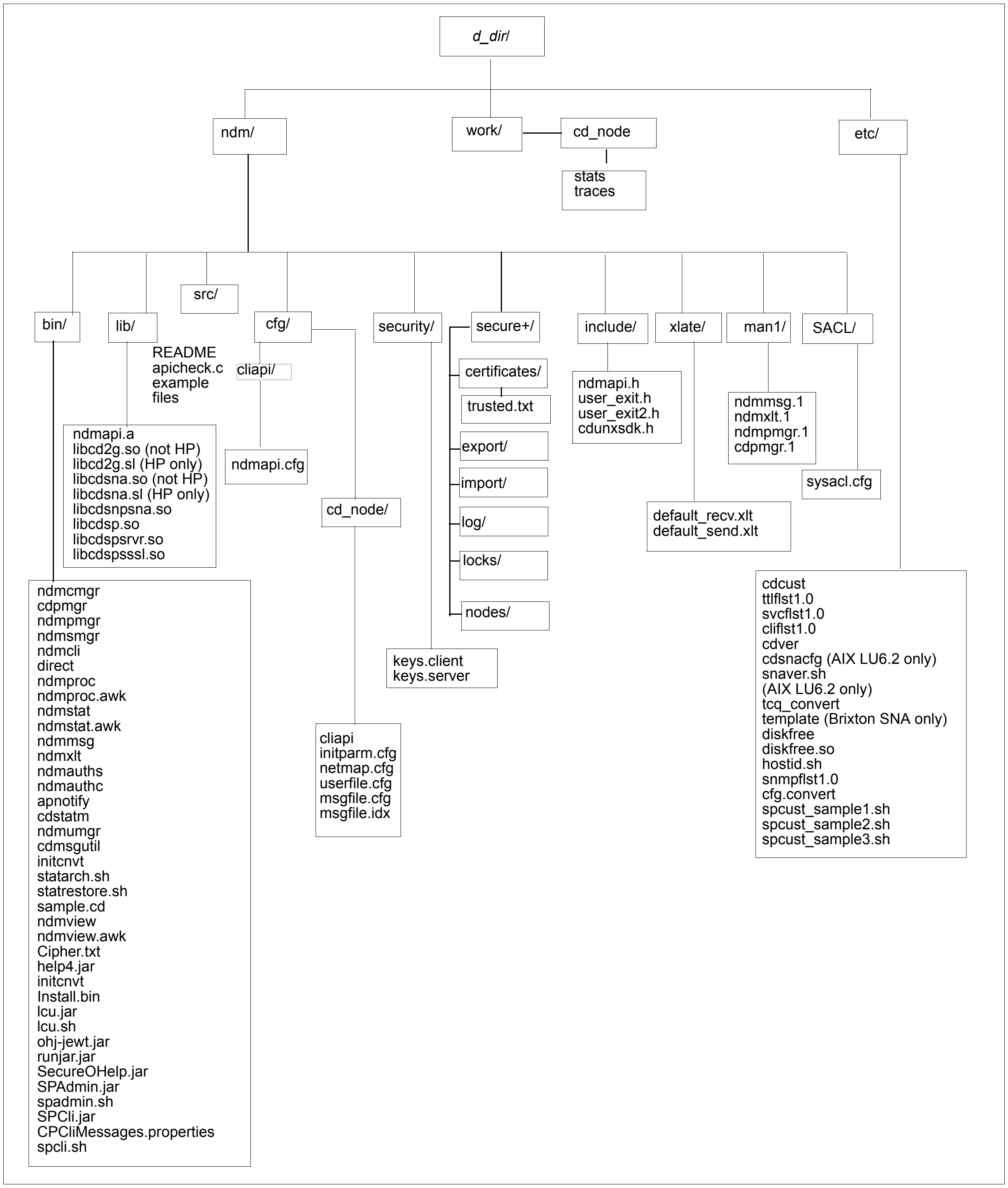
Connect:Direct for UNIX creates the following configuration files during installation and customization. These files are required for the Connect:Direct server to operate correctly.

| Configuration File | Description |
|---|---|
| Initialization parameters file | Provides information to the server to use at start up. During the installation, you identify the settings necessary for the initialization parameters file. |
| User authorization information file | Contains the local user information and remote user information record types. You customize this file during installation to map remote user IDs to local user IDs and create remote user information records in the user authorization information file. |
| Strong access control file | Improves the security of b and prevents a user from obtaining root access control. Initially, the file is empty and contains no restrictions. However, if you want to prevent the root user or any user with root permissions from gaining access to Connect:Direct for UNIX, you define a restriction parameter in this file. If the file is deleted or corrupted, access to Connect:Direct is denied to all users. |
| Network map file | Describes the local node and other Connect:Direct nodes in the network. You can define a remote node record for each node that Connect:Direct for UNIX communicates with. |
| Server authentication key file | Verifies client API connection requests. Only clients are granted a connection. |
| Client configuration file | Identifies the port and host name used by a client to connect to Connect:Direct. |
| Client authentication key file | Identifies Connect:Direct servers that a Connect:Direct node connects to. You can have multiple entries for multiple servers. |

## Connect:Direct for UNIX Directory Structure

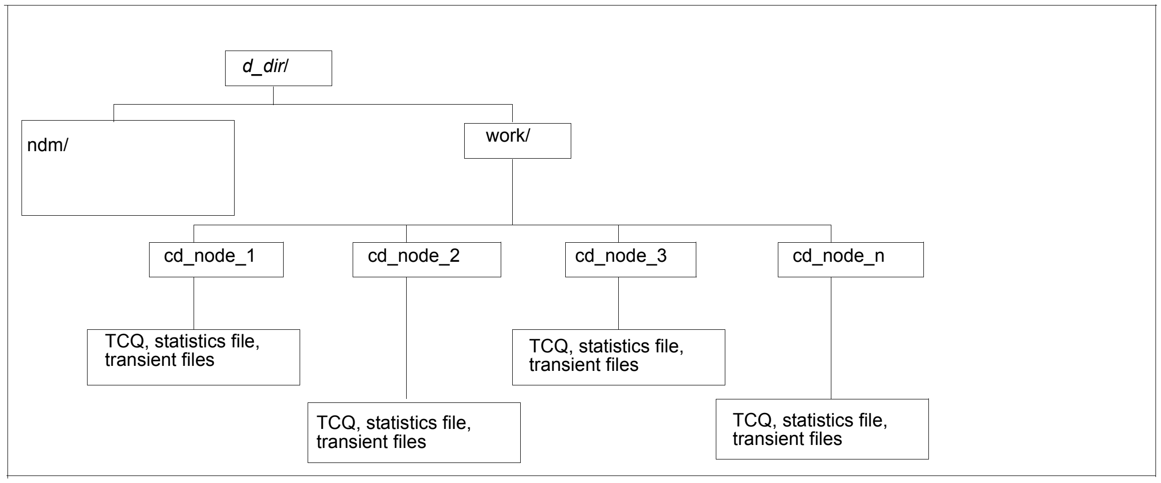
The following figure illustrates the Connect:Direct for UNIX directory structure. The directory tree starts at *d_dir/*, the destination directory where the software is installed. This directory structure provides for multiple nodes on the same network and possibly on the same computer. The directory structure organization enables you to share Connect:Direct programs, such as cdpmgr and dmcmgr. If multiple nodes exist, each node must have its own *d_dir*/ndm/cfg/*cd_node*/ directory structure for configuration files, where *cd_node* is the Connect:Direct node name.

**Note:** The secure+ directory is available only when Secure+ Option is purchased.



**Note:** See the following figure to view the work directory for a node.

A *d_dir*/work/*cd_node* directory is created for each node. The following figure displays the work directory for multiple nodes and illustrates the working files created for each node, such as TCQ files:


d_dir/
ndm/
work/
cd_node_1
cd_node_2
cd_node_3
cd_node_n
TCQ, statistics file, transient files
TCQ, statistics file, transient files
TCQ, statistics file, transient files
TCQ, statistics file, transient files

**Note:** See the previous figure for details on the ndm directory structure.

## Connect:Direct for UNIX Documentation

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

### About This Guide

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

This guide assumes knowledge of the UNIX operating system, including its applications, network, and environment. For LU6.2 connectivity, a proficiency in configuring the SNA package to support independent LU6.2 connections is required.

### Task Overview

The following table guides you to the information required to perform Connect:Direct for UNIX tasks:

| **Task** | **Reference** |
|---|---|
| Understanding Connect:Direct for UNIX | Chapter 1, *About Connect:Direct for UNIX* |
| Modifying the configuration files | Chapter 2, *Maintaining Configuration Files* |
| Understanding the parameters in the initialization parameters file | Chapter 3, *Maintaining the Initialization Parameters File* |
| Understanding the parameters in the client configuration file | Chapter 4, *Maintaining the Client Configuration File* |
| Understanding the parameters in the network map file | Chapter 5, *Maintaining the Network Map File* |
| Understanding the user authorization information file | Chapter 6, *Maintaining Access Information Files* |
| Understanding the key file parameters in the client and server authentication files | Chapter 7, *Maintaining Client and Server Authentication Key Files* |

Chapter 2

# Maintaining Configuration Files

Configuration files define the operating environment for Connect:Direct. The following configuration files are created during the customization procedure:

- Initialization parameters file
- Client configuration parameters file
- Network map file
- Two access files, called userfile.cfg and sysacl.cfg

After the initial customization, you can modify these files, if necessary. This chapter provides you with the information to modify the configuration files.

## About the Configuration Files

A configuration file is a text file composed of records. A record is a single logical line. A logical line is one or more physical lines that can be continued with the backslash (\) character. In the table on page 24, physical lines 4 and 5 illustrate a logical line. Line 4 ends with a backslash (\) character, to indicate that the line is continued on the next physical line. Line 1 of the sample begins with a pound (#) sign. The pound sign indicates this line contains a comment.

A record consists of a record name and one or more parameter pairs. A parameter pair is a parameter name and parameter value. Line 2 contains the record name, ndm.path. Line 2 also contains the parameter pair, path and /ndm/users/c, where the parameter name is path and the parameter value is /ndm/users/c. The parameter pair is bound by colons (:) and separated by an equal sign (=) in the following format. The following example displays a complete record, where ndm.path is the record name, path is the parameter name, and /ndm/users/c is the parameter value:

```
ndm.path:path=/ndm/users/c:
```

Record names and parameter names are not case sensitive. Parameter values are case sensitive.

Lines 7 through 23 illustrate a longer logical record. Line 7 contains the record name local.node followed by an optional colon (:) and a backslash (\) character. All lines between 7 and 23 end with a backslash (\) character. Line 23 does not contain a backslash (\) character, to indicate the end of the record.

The following table displays a portion of the initialization parameters file to illustrate the format of Connect:Direct configuration files:

| Line | Contents | Notes |
|---|---|---|
| 1 | #Miscellaneous Parameters | # indicates a comment |
| 2 | ndm.path**:**path=/ndm/users/c: | record name=ndm.path,<br>parameter=path<br>value= /ndm/users/c |
| 3 | proc.prio:default=8: | record name=proc.prio,<br>parameter=default<br>value= 8 |
| 4<br>5 | asset.protection:\<br>keyfile=${CDUNIX_PATH}/ndm/cfg/main_node/license.key: | License keyfile |
| 6 | #Local Connect:Direct connection information | # indicates a comment |
| 7 | local.node:\ | record name=local.node |
| 8 | :api.max.connects=11:\ | parameter=api.max.connects<br>value= 11 |
| 9 | :conn.retry.stwait=00.00.12:\ | parameter= conn.retry.stwait, value= 00.00.12 |
| 10 | :conn. retry.stattempts=2:\ | parameter=conn.retry. stattempts, value= 2 |
| 11 | :conn.retry.ltwait=00.01.01:\ | parameter=conn.retry.ltwait, value= 00.01.01 |
| 12 | :conn. retry. ltattempts=4:\ | parameter= conn.retry.ltattempts, value= 4 |
| 13 | . | |
| . . . | . | |
| 21 | . | |
| 22 | :tcp.api=rusty;3191:\ | parameter= tcp.api, value= rusty;3191 |
| 23 | :tcp.api.bufsize=**32768**: | parameter= tcp.api.bufsize<br>value= 32768 |

Configuration files allow duplicate but not identical records, in some cases. For example, you can define more than one remote node information (**rnode.listen**) record in the initialization parameters file.

## Modifying Configuration Files

You can modify Connect:Direct configuration files using any text editor or create a new configuration file using the **cdcust** command provided with Connect:Direct for UNIX.

- Modifying Configuration Files with a Text Editor—You can modify Connect:Direct configuration files with any text editor, such as vi editor.
- Creating Configuration Files with cdcust—Type the following command to start the customization procedure, where *d_dir* is the Connect:Direct for UNIX path name:

```
$ d_dir/etc/cdcust
```

Chapter 3

# Maintaining the Initialization Parameters File

Initialization parameters determine various Connect:Direct settings that control system operation. The initialization parameters file is created when you install Connect:Direct for UNIX and can be updated as needed.

You can modify Connect:Direct configuration files using any text editor. Before changing a value in the initialization parameters file, first shut down the Connect:Direct server. After you change a value and save the file, restart the server. Restarting the server validates the new values and generates an error message if a value is invalid. All available parameters are described in this chapter.

If you use Connect:Direct Browser User Interface to update parameters in the Local Node Connection Record, you do not have to stop and restart the server.

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

## About the Initialization Parameters File

The initialization parameters file resides in *d_dir*/ndm/cfg/*cd_node*/initparm.cfg, where *d_dir* is the destination directory where Connect:Direct for UNIX is installed and *cd_node* is the node name.

The initialization parameters file contains records. Each record includes parameters to define the attributes of the record. The records are summarized as follows:

✦ Miscellaneous parameters—Provide miscellaneous information including the name of the Connect:Direct for UNIX node; the location of Connect:Direct for UNIX, the Pluggable Authentication Modules (PAM) service configuration file, the shared work area for SNODE work files, and the license management key file; the default Process priority; and whether commands with special characters are restricted in the run directory.

✦ Remote node connection information—The rnode.listen record includes parameters to monitor inbound connections.

✦ Transmission Control Queue (TCQ) information—The tcq record defines how long a Process is held in error before being deleted.

- Global copy parameters—The **copy.parms** record defines default parameters used by the Copy operation including checkpoint parameters, file size limitations, translation table information, exception handling, CRC checking, file allocation retry parameters, and compression options.
- Global run task parameters—The **runtask.parms** record defines a parameter to define the restart option.
- Statistics file information—The **stats** record includes parameters to define default statistics file information including file size limitations, the type of information to write to the statistics file, and how long to maintain statistics files before archiving them.
- Server authentication information—The **authentication** record parameters to authenticate the server.
- User exit parameters—The **user.exits** record defines the programs used during a user exit procedure.
- Firewall navigation information—The **firewall.parms** record defines the ports or range of ports to use for outbound sessions when a server operates behind a firewall.

The following sample initialization parameters file shows how some of these parameters are specified:

```
# Miscellaneous Parameters
ndm.path:path=/sci/users/mscarbro/cd3800:\
        :snode.work.path=/sci/users/mscarbro/cd3800/shared:

ndm.node:name=mws_joshua_3800:
ndm.pam:service=cdlogin:

proc.prio:default=10:

asset.protection:keyfile=/sci/users/mscarbro/cd3800/ndm/cfg/mws_joshua_3800/licens
e.key:

restrict:cmd=y

# TCQ information
tcq:\
 :max.age=8:

# Global copy parameters.
copy.parms:\
 :ckpt.interval=2M:\
 :ulimit=N:\
 :xlate.dir=/sci/users/mscarbro/cd3800/ndm/xlate:\
 :xlate.send=def_send.xlt:\
 :xlate.recv=def_recv.xlt:\
 :continue.on.exception=y:

# Global runtask parameters.
runtask.parms:\
 :restart=y:

# Stat file info.
stats:\
 :file.size=1048576:\
 :log.commands=n:\
 :log.select=n:\
 :snmp.agent.port=1365:\
 :snmp.agent.activated=n:\
 :syslog.logd=daemon:

# Authenticator
authentication:\
 :server.program=/sci/users/mscarbro/cd3800/ndm/bin/ndmauths:\
 :server.keyfile=/sci/users/mscarbro/cd3800/ndm/security/keys.server:

# user exit information
user.exits:\
 :security.exit.program=:\
 :file.open.exit.program=:\
 :stats.exit.program=:

# Remote CDU nodes
rnode.listen:\
 :recid=mws_joshua_3800:\
 :comm.info=0.0.0.0;11364:\
 :comm.transport=tcp:
```

## Updating Miscellaneous Parameters

This section identifies the miscellaneous records and defines available parameters. Required parameters are displayed in bold.

### Updating the Path Record

The **ndm.path** record identifies the path to Connect:Direct files. The following table describes the parameter available for this record:

| Parameter | Description | Value |
|---|---|---|
| path | The path to all Connect:Direct subdirectories and files. | path specification |

### Updating the SNODE Work Path

The **snode.work.path** parameter is part of the **ndm.path** record and identifies the path to the shared work area for SNODE work files on a cluster file system (not an NFS). This optional parameter provides a means to share SNODE work files among nodes in a load balancing environment. SNODE return code files (steprc files) and **copy** checkpoint information are created in this area when the **snode.work.path** parameter is specified. The following table describes the **snode.work.path** parameter:

| Parameter | Description | Value |
|---|---|---|
| snode.work.path | The path to the shared work area for SNODE work files.<br>**Note:** Specify the same path for all nodes in a cluster. | path specification |

### Updating the Node Name Record

The **ndm.node** record identifies the name of the Connect:Direct node. The following table describes the parameter available for this record:

| Parameter | Description | Value |
|---|---|---|
| name | The name of the node. | The maximum length is 256 bytes. If a node name is longer, the name will be truncated. |

### Updating the PAM Service Record

The **ndm.pam** record identifies the PAM service configuration file used to authenticate the user authority for Connect:Direct Processes. If the service initialization parameter is defined and if PAM is installed on the Connect:Direct for UNIX server, PAM is used to authenticate users for service-providing applications. The following table describes the parameter available for this record:

| Parameter | Description | Value |
|---|---|---|
| service | PAM service configuration file name. | File name |

## Updating the Priority Record

The **proc.prio** record identifies the default value of the Process priority. The following table describes the parameter available for this record:

| Parameter | Description | Value |
|---|---|---|
| default | The default value of the Process priority. | 1–15. The default value is **10**. |

## Updating the License Management Key Record

The **asset.protection** record identifies the name and location of the license management key file. The following table describes the parameter available for this record:

| Parameter | Description | Value |
|---|---|---|
| keyfile | The license management key file. | name and location of the key |

## Restricting the Use of Special Characters in the Run Directory

If a directory restriction is defined in the user configuration file (userfile.cfg), the **restrict** record determines if commands containing certain special characters are allowed. For more information on the userfile.cfg file, see *Updating the Local User Information Record Format* on page 61 and *Updating the Remote User Information Record* on page 65. The following parameter is available for this record:

| Parameter | Description | Values |
|---|---|---|
| cmd | Determines if commands with certain special characters are allowed. | **n—**Does not restrict allowed commands.<br>**y:—**Restricts the ability to use commands with any of the following special characters:<br>; & ‘ \| |

## Updating the Remote Node Connection Record

The **rnode.listen** record contains parameters used by the local node to monitor inbound connection requests. Define an **rnode.listen** record for each remote node you communicate with. You can modify the IP address and port number in the **rnode.listen** record while the server is running. However, you must recycle the server before the change is active. The following table describes the remote node connection parameters:

| Parameter | Description | Values |
|---|---|---|
| recid | A unique identifier of an **rnode.listen** record. | A text string |
| comm.info | The information needed to monitor connection requests from remote nodes using TCP/IP or LU6.2. This parameter is required.<br>◆ For TCP/IP connections, specify the host name or the IP address and port name or number. If specifying IP address and port, separate parameters with a semicolon (;).<br>◆ For LU6.2 connections, identify the profile name to identify the name of an SNA configuration profile for the remote connection. Connect:Direct generates the default name, hostl1, during the customization procedure. For AIX SNA, hostl1 refers to the side information profile name. For HP SNA, SunLink SNA, and Brixton SNA, hostl1 refers to the SNA profile file located in the same directory as the configuration files. | For TCP/IP connections, specify the host name or the IP address and port number.<br>Set the IP address to monitor a specific adapter or to **0.0.0.0**, to monitor all adapters.<br>The default port is **1364**.<br>For LU6.2 connections, specify a profile name, up to 8 characters. |
| comm.transport | The transport protocol for the remote node. | tcp—TCP/IP connections<br>lu62—For AIX SNA LU6.2 connections<br>blklu62—For other LU6.2 connections |

## Updating the TCQ Record

The **tcq** record provides information pertaining to the Transmission Control Queue. The following parameter is available for this record:

| Parameter | Description | Value |
|---|---|---|
| max.age | The maximum number of days a Process with Held-in-Error status remains in the TCQ before it is automatically deleted. | A 3-digit decimal number. Connect:Direct does not automatically delete Processes when max.age=0.<br>The default is **8** days. |

## Updating the Global Copy Record

The Global Copy record called **copy.parms** provides default information for the Connect:Direct copy operation. The ecz parameters are only used when extended compression is defined in a Process. The following parameters are available for this record:

| Record | Description | Value |
|---|---|---|
| ckpt.interval | The default number of bytes transmitted in a copy operation before a checkpoint is taken. Following is a list of the maximum number of digits for each byte interval:<br>no—No checkpointing<br>nnnnnnnn—Up to an 8-digit decimal<br>*nnnnnnnn*K—Up to an 8-digit decimal, where K denotes 1024 bytes<br>*nnnnnnnn*M—Up to an 8-digit decimal, where M denotes 1048576 bytes<br>*nnnnnnnn*G—Up to an 8-digit decimal, where G denotes 1073741824 bytes | The maximum possible value is 1 terabyte (TB). The normal value is 64KB. |
| ulimit | The action taken when the limit on a user output file size is exceeded during a copy operation. | **n**—Ignores the limit. **n** is the default value.<br>**y**—Recognizes the user file size limit. If this limit is exceeded during a copy operation, the operation fails. |
| xlate.dir | The name of the directory containing the translation tables. | Any valid directory.<br>The default path is ***d_dir*/ndm/xlate**. |

| Record | Description | Value |
|---|---|---|
| xlate.send | The default translation table used when sending data to a remote node. | Any valid directory.<br>The default file name is **def_send.xlt**. |
| xlate.recv | The name of the default translation table used when copying data from a remote node. | The default file name is **def_recv.xlt** in the directory defined in the **xlate.dir** parameter. |
| continue.on.exception | The method to use to handle an exception condition in a Process. If a step fails due to a STOP IMMEDIATE or FLUSH exception issued on the remote node, the Process is placed in the Hold HE queue, regardless of the value of this parameter. | **y**—Continues Processing with the next step.<br>**n**—Places a Process in the Hold queue with a value of HE.<br>The default is **n**. |
| ecz.compression.level | Sets the compression level. | 1–9. The default is **1**.<br>1—The fastest but offers the least degree of compression.<br>9—Provides the greatest degree of compression but is the slowest. |
| ecz.memory.level | How much virtual memory to allocate to maintaining the internal compression state. | 1–9. The default is **4**.<br>1—Uses the least memory and 9 uses the most memory. |
| ecz.windowsize | The size of the compression window and history buffer. The larger the window, the greater the compression. However, increasing the window uses more virtual memory. | Valid values are 9–15. The default is **13**. |

| Record | Description | Value |
|---|---|---|
| retry.codes | The codes to recognize as a file allocation retry attempt. File allocation retry enables a Process with a file allocation or open error on either the local or remote node to run the Process again, beginning at the copy step where the error occurred. This feature supports the ability to retry a Process that failed when a file is already in use.<br>When a file allocation or open error occurs on either the local or remote node, the PNODE searches for the error or message ID in the retry.codes and retry.msgids parameters. If the error code or message ID is found, the Process is retried.<br>Since error codes can vary from one operating system to another and the same error code can have different meanings, use message IDs to identify retry conditions when communicating between two different platforms.<br>You can perform retry attempts based on codes only, IDs only, or a combination of the two.<br>When a retry condition is detected, the session is terminated cleanly and the Process is placed in the Timer queue. | Any valid error code |
| retry.msgids | Identifies the message IDs to use to support a file allocation retry attempt.<br>Since error codes can vary from one operating system to another and the same error code can have different meanings, use message IDs to identify retry conditions when communicating between two different platforms. When a file allocation or open error occurs on either the local or remote node, the PNODE searches for the message ID in the retry.msgids parameters. If the message ID is found, the Process is retried.<br>You can perform retry attempts based on codes only, message IDs only, or a combination of the two.<br>When a retry condition is detected, the session is terminated cleanly and the Process is placed in the Timer queue. | Any of the valid file allocation retry messages. |
| tcp.crc | Globally turn on or off the CRC function for TCP/IP processes. | y \| n |
| tcp.crc.override | Determines whether node and process statement overrides for CRC checking are allowed. If this value is set to n, setting overrides for CRC checking will be ignored. | y \| n |

| Record | Description | Value |
|---|---|---|
| strip.blanks | Determines whether trailing blank characters are stripped from the end of a record. If strip.blanks is not defined in the initialization parameter, the default value of i is used. | y \| n \| i<br>**y**—Strips blanks from the end of a record<br>**n**—Does not strip blanks from the end of a record<br>**i**—Ignores the setting defined in the .Local node record and uses the settings for strip.blanks as determined by the default value of the remote node type as follows:<br>OS/390, VM, VSE, and OS/400—y<br>All other platforms—n |

## Updating the Global Run Task Record

The Global Run Task record called **runtask.parms** is used if the PNODE and SNODE cannot resynchronize during a restart. If a Process is interrupted when a run task on an SNODE step is executing, Connect:Direct attempts to synchronize the previous run task step on the SNODE with the current run task step. If synchronization fails, Connect:Direct reads the **restart** parameter to determine whether to perform the **run task** step again. The following parameter is available for this record:

| Parameter | Description | Value |
|---|---|---|
| restart | If processing is interrupted when a **run task** on an SNODE step is executing and if synchronization fails after a restart, Connect:Direct reads the **restart** parameter to determine whether to perform the **run task** step again. Set this parameter in the initialization parameters file of the SNODE.<br>**Note:** When a load balancing cluster is used and the snode.work.path *is* specified, the **restart** parameter takes effect only when resynchronization fails. | y \| n<br>**y**—the run task program runs again<br>**n**—the Process skips the **run task** step. |

## Updating the Statistics File Information Record

The statistics file information record called **stats** define the statistics facility. The following parameters are available for this record:

| Parameter | Description | Value |
|---|---|---|
| file.size | The maximum size in bytes of an individual statistics data file. The statistics file name is written in the format of **Syyyymmdd.ext**, where **yyyy** indicates year, **mm** indicates month, and **dd** indicates day. The extension (**ext**) begins as 001. When a statistics file reaches the defined size within a 24-hour period, a new file is created with the same file name. The extension value is incremented by one. | **nnnnnnnn**, **nnnnnnnnK**, **nnnnnnnnM**, or **nnnnnnnnG**—Establishes a default output file size limit for the statistics files. K denotes 1024 bytes. M denotes 1048576 bytes. G denotes 1073741824 bytes. The maximum value you can specify is 1 TB. |
| log.commands | Determines whether commands are written to the statistics file. If you want to log all commands except the select statistics and select process commands, set this parameter to y and the log.select parameter to n. | y \| n |
| log.select | Specifies whether Connect:Direct creates a statistics record when a select process or select statistics command is executed. | y \| n |
| snmp.agent.port | The SNMP agent monitoring port number. This number must match the port number listed in the SNMP agent initialization file. | The default is **1365**. |
| snmp.agent.activated | Notifies the server if an SNMP agent is active. | **y** \| **n** |
| syslog.logd | Enables Connect:Direct to send license management failure messages to the UNIX syslogd daemon. Refer to manual pages for instructions on configuring syslog. | Available values are:<br>kern—Kernel<br>user—User level<br>mail—Mail subsystems<br>daemon—System daemons<br>auth—Security or authorization<br>syslog—syslogd daemon<br>lpr—Line-printer subsystem<br>news—News subsystem<br>uucp—uucp subsystem<br>The default value is **daemon.** |

| Parameter | Description | Value |
|---|---|---|
| max.age | Specifies how old a statistics file must be before it is archived. Once a day, a shell script is executed that identifies the statistics files that are as old as the max.age, runs the tar command and the compress command to create a compressed archive, and then deletes the statistics files that have been archived. | A 3-digit decimal number. The default is **8** days. 0—no archiving. |

Running a Process generates multiple statistics records. To accommodate the large number of statistics records generated, Connect:Direct closes the current statistics file and creates a new statistics file at midnight every day. It can also close the current file before midnight if the file size exceeds the value set for the **file.size** initialization parameter. The default file size is **1** megabyte.

Statistics files are stored in the *d_dir*/work/cd_node directory. Names of the statistics files are in the format **Syyyymmdd.ext**, where **yyyy** indicates year, **mm** indicates month, and **dd** indicates day. The extension (**ext**) begins as 001. The extension is incremented by one each time a new statistics file is created in a single day.

Connect:Direct for UNIX provides a utility to archive and purge statistics files. You identify when to archive a statistics file by setting the parameter, max.age. The max.age parameter defines how old a statistics file must be before you want to archive the file. Once a day, the script called statarch.sh is started. This script identifies the statistics files that are greater than or equal to the max.age. It then runs the tar command and the compress command to create a compressed archived file of all the statistics records that match the max.age parameter. Once the statistics files are archived, these files are purged.

The archived files are stored in the directory where the statistics files and TCQ are stored. The shell script, statarch.sh, is located in the ndm/bin directory. If necessary, modify the script to customize it for your environment.

If you want to restore statistics files that have been archived, run the statrestore.sh script. It uses the tar command to restore all the statistics files in the archive. Once files are restored, the statistics records can be viewed using the select statistics command.

## Updating the Server Authentication Record

The server authentication record called **authentication** is used during the authentication procedure. The following parameters are available for this record:

| Parameter | Description | Value |
|---|---|---|
| server.program | The name and location of the server program used during the authentication procedure. | The default is **ndmauths**. |
| server.keyfile | The name and location of the key file used during the authentication procedure. | The default is **keys.server**. |

## Updating the User Exit Record

The user exit record called **user.exits** provides interfaces to specified programs. The available user exits include Statistics Exit, File Open Exit, and Security Exit. The following parameters are available for this record:

| Parameter | Description | Value |
|---|---|---|
| stats.exit.program | The gateway control program used during the user exit procedure. This exit is given control for each statistics record that is written. | Name of the gateway control program. |
| file.open.exit.program | The file open exit program used during the user exit procedure. It enables you to control file names on both the sending and receiving node. The exit is located so that it takes control on the receiving (remote) node before the file is opened.<br>This exit applies only to the **copy** statement and provides access to all file control parameters (including the data set name file name, sysopt parameters, and disposition). | Name of the file open exit program. |
| security.exit.program | The security exit program used during the user exit procedure. This exit generates and verifies passtickets and it also supports other password support programs. An example of other programs includes PASSTICKET, part of the RACF™ security system available on MVS hosts and also supported by IBM on UNIX AIX and OS/2 computers using the NETSP product. | Name of the security exit program. |
| security.exit.flag | Modifies the default behavior of security.exit.program. This is an optional parameter. | Available values are:<br>snode_sec_exit_only—Causes Connect:Direct to use the security exit, when it is acting in the role of the SNODE. After Connect:Direct receives a valid message, it evaluates the proxy and the secure point-of-entry to establish the local user. The security exit is not used when Secure+ Option is the PNODE.<br>sec_exit_only—Causes Connect:Direct to always use the security exit. After Connect:Direct receives a valid message, it evaluates the proxy and the secure point-of-entry to establish the local user. |

## Updating the Firewall Navigation Record

The firewall navigation record called firewall.parms enables you to assign a specific TCP/IP source port number or a range of port numbers with a particular TCP/IP address for outbound Connect:Direct sessions. These ports also need to be open on the firewall of the trading partner to allow the inbound Connect:Direct sessions. This feature enables controlled access to a Connect:Direct server if it is behind a packet-filtering firewall without compromising security policies.

The following parameters are available for this record:

| Parameter | Description | Value |
|---|---|---|
| tcp.src.ports | IP addresses and the ports permitted for the addresses when using a packet-filtering firewall. At least one value is required.<br>Place all values for an address inside parentheses and separate each value for an address with a comma. | You can specify the valid IP address or pattern, subnet mask, and ports or range of ports in any combination.<br>Type one or more of the following values:<br>valid IP address or pattern with wildcard values (such as nnn.nnn.*.*). If the wildcard character is used, the optional subnet mask is not allowed.<br>subnet mask using any of the following subnet mask values: Dotted Quad notation, such as 255.255.0.0, or Hexadecimal notation, such as 0xffffff00The subnet mask value, when converted to binary format, must be contiguous ones and then contiguous zeros. Providing ones after the zeros causes the mask to be rejected.<br>**port name or number or range of port names or numbers**<br>The list of IP addresses can contain up to 255 unique IP addresses or address patterns, each with its own list of valid source ports |
| tcp.src.ports.list.iterations | The number of times that Connect:Direct scans the list of available ports to attempt a connection before going into a retry state. | Any numeric value from 1–255. The default value is **1**. |

Chapter 4

# Maintaining the Client Configuration File

The client configuration file consists of parameter records that interface with End User Applications (EUA). The client file includes the following parameters:

- ✦ Connect:Direct API configuration parameters
- ✦ Connect:Direct CLI configuration parameters
- ✦ Client authentication parameters

This chapter describes the parameters in the client configuration file. You can modify Connect:Direct configuration files using any text editor. If you want to create a new configuration file, use the **cdcust** command.

**Note:** You can also use Connect:Direct Browser to perform some of the procedures in this chapter. 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.

## About the Client Configuration File

The client configuration file is created during the customization procedure and resides in *d_dir*/ndm/cfg/cliapi/ndmapi.cfg, where *d_dir* is the directory where Connect:Direct is installed.

A sample client configuration file is displayed in the following example:

```
# Connect:Direct for UNIX Client configuration file

cli.parms:\
  :script.dir=/home/qatest/jsmith/cdunix/hp/ndm/bin/:\
  :prompt.string="Test CD on Medea":

api.parms:\
  :tcp.hostname=alicia:\
  :tcp.port=1393:\
  :wait.time=50:

# Authenticator
authentication:\
  :client.program=/home/qatest/jsmith/cdunix/hp/ndm/bin/ndmauthc:\
  :client.keyfile=/home/qatest/jsmith/cdunix/hp/ndm/sc/keys.client:
```

## Updating the API Configuration Record

The Connect:Direct API Configuration record, **api.parms**, is used by the API to communicate. The parameters for the API configuration record are described in the following table:

| Parameter | Description | Value |
|---|---|---|
| tcp.hostname | The host name or IP address to which the API usually connects. | Host name or IP address. |
| tcp.port | The TCP/IP port number or name to which the API usually connects. | Port name or number. The default is **1363**. |
| wait.time | The number of seconds to wait for responses from the server. If this limit is exceeded, the message ID XCMG000I is displayed. | Seconds to wait. The default is **50** seconds. |

## Updating the CLI Configuration Record

The CLI configuration record, **cli.parms**, identifies the location of the script files to format the output of the **select statistics** and **select process** commands and allows you to customize the CLI prompt. If you customize the script to format the output of the **select statistics** and **select process** command, update the **script.dir** parameter to identify the location of the scripts. If you want to display a customized prompt at the CLI command line, in place of the default "Direct" prompt, identify the prompt to use in the **prompt.string** parameter. The **cli.parms** parameters are described in the following table:

| Parameter | Description | Value |
|---|---|---|
| script.dir | The directory where customized script files are stored. Specify this parameter if you have created a custom script to format the output of the **select statistics** and **select process** commands. The file names must be ndmproc and ndmstat. | Directory name.<br>The default<br>directory is<br>**ndm/bin/**. |
| prompt.string | Identifies the CLI prompt to display on the command line when the client is started.<br>If the prompt string includes spaces or special characters, enclose it in single or double quotation marks.<br>You can set the customized prompt in this parameter and at the command line (using the -P parameter). If the prompt string is specified in both places, the -P parameter at the command line takes precedence.<br>When the default prompt is overridden, the new prompt string is displayed in the Welcome banner and at the command prompt. | Prompt string up to 32 characters. The default is “**Direct**”. |

## Updating the Client Authentication Record

The client authentication record, authentication, is used during the authentication procedure. The client authentication parameters are described in the following table:

| Parameter | Description | Value |
|---|---|---|
| client.program | The client program to use during authentication. | Client program name.<br>The default is **ndmauthc**. |
| client.keyfile | The key file to use during authentication. | Client key file. The default is **keys.client**. |

Chapter 5

# Maintaining the Network Map File

This chapter describes the parameters in the network map file. This file is created when you install Connect:Direct for UNIX. If necessary, use a text editor to add or modify remote node records in the network map file. You can modify the network map file dynamically while the server is running.

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

## About the Network Map File

The network map contains connectivity information that describes the local node and the remote nodes in the network. One remote node information record is created for each node with which the local node communicates.

The network map file resides in *d_dir*/ndm/cfg/*cd_node*/netmap.cfg where *d_dir* is the location where Connect:Direct is installed and *cd_node* is the node name.

**Note:** If you are using TCP/IP, the local node can communicate with a remote node without a remote node information record. Specify the required connection information in the submit command or the process statement. If no values are specified, the values in the local.node record are used.

## Sample Network Map Entry for Remote Node

The following sample shows network map remote node entries for a TCP/IP connection and a Sun LU6.2 connection to remote nodes.

```
# Sample Network Map remote node entry for a TCP/IP connection
# testhp is the hostname of the remote Connect:Direct node
testcdu:\
  :conn.retry.stwait=00.00.30:\
  :conn.retry.stattempts=3:\
  :conn.retry.ltwait=00.10.00:\
  :conn.retry.ltattempts=6:\
  :tcp.max.time.to.wait=0;\
  :runstep.max.time.to.wait=0:\
  :contact.name=:\
  :contact.phone=:\
  :descrip=:\
  :sess.total=255:\
  :sess.pnode.max=128:\
  :sess.snode.max=127:\
  :sess.default=1:\
  :comm.info=testhp;7777:\
  :comm.transport=tcp:\
  :comm.bufsize=16000:\
  :pacing.send.delay=0:\
  :pacing.send.count=0:
# Sample Network Map remote node entry for an Sun LU6.2 connection
# hostl1 is the profile name
MVS.SAM1.NODE:\
  :conn.retry.stwait=00.00.30:\
  :conn.retry.stattempts=3:\
  :conn.retry.ltwait=00.10.00:\
  :conn.retry.ltattempts=6:\
  :contact.name=:\
  :contact.phone=:\
  :descrip=:\
  :sess.total=255:\
  :sess.pnode.max=128:\
  :sess.snode.max=127:\
  :sess.default=1:\
  :comm.info=hostl1:\
  :comm.transport=blklu62:\
  :comm.bufsize=16000:
```

## Updating the Local Node Connection Record

The local.node record defines default information to use when communicating with a remote node. The local connection information consists of the local node connection information and the TCP/IP information. Two sets of connection retry parameters are created: short-term and long-term. Short-term parameters define retry attempts in the event of a short-term connection failure. Connect:Direct uses long-term parameters after exhausting short-term attempts. Long-term attempts are set for less frequent retries, because long-term attempts assume that the connection problem cannot be fixed quickly.

Following are the local.node parameters. The parameters in bold are required.

| Parameter | Description | Value |
|---|---|---|
| name | The name of the local node. | Up to 16 alphanumeric characters long. |
| api.max.connects | The maximum number of concurrent API connections permitted for the local node.<br>The value of api.max.connects and sess.total cannot exceed the number of file descriptors available. This value is system dependent.<br>A Command Manager (CMGR) is created for every API connection that is successfully established. The number of Command Managers that a PMGR can create is system-dependent and limited by the number of file descriptors available for each UNIX Process. The number of file descriptors set up by the UNIX operating system may affect Connect:Direct operation. | 1–256. The default is **16**. |
| conn.retry.stwait | The time to wait between retries immediately after a connection failure occurs. The format is hh.mm.ss, where **hh** specifies hours, **mm** specifies minutes, and ss specifies seconds. | The maximum value is limited to the highest value in the clock format, 23.59.59. The default is **00.00.30**, which is 30 seconds. |
| conn.retry.stattempts | The number of times to attempt connection after a connection failure occurs. | 0–9999. The default is **6**. |
| conn.retry.ltwait | The time to wait between long-term retry attempts after all short-term retry attempts are used. The format is hh.mm.ss, where hh specifies hours, mm specifies minutes, and ss specifies seconds. | 00.00.00–23.59.59.<br>The default is **00.10.00**, or 10 minutes. |
| conn.retry.ltattempts | The number of times to attempt a connection after all short-term retry attempts are used. | 0–9999. The default is **6**. |
| conn.retry.exhaust.action | Action to take after the specified short and long-term retries have been used. | hold—Places Processes in the hold queue in “Held in Error” status, after all retry attempts are used. This is the default value.<br>delete—Causes the Processes to be deleted from the TCQ. |
| contact.name | The name of the Connect:Direct administrator or operator. | |

| Parameter | Description | Value |
|---|---|---|
| contact.phone | The phone number of the Connect:Direct administrator or operator. | 999-999-9999 |
| descrip | Comments to include as part of the record. | An unlimited string. |
| sess.total | The maximum number of concurrent connections between all nodes and the local node.<br>The sum of api.max.connects and sess.total cannot exceed the number of file descriptors available. This value is system dependent.<br>You must define enough file descriptors to handle the number of concurrent Connect:Direct sessions allowed, which can be as many as 999.<br>If sess.total exceeds the number of sessions in the APS license key file, the license key file silently overrides this value. | A 1–3 digit number.<br>The default is **255**. |
| sess.pnode.max | The maximum concurrent connections, where the local node is the initiator of the session. Number of PNODE sessions cannot exceed the total number of sessions.<br>If sess.pnode.max is larger than sess.total, the value of sess.pnode.max is silently rounded down to the value of sess.total.<br>The sum of snode.max and pnode.max cannot be greater than sess.total. | The default is **255**. |
| sess.snode.max | The maximum concurrent connections, where the local node is the secondary node in a session.<br>Number of SNODE sessions cannot exceed the total number of sessions. If sess.snode.max is larger than sess.total, then it is silently changed to the value of sess.total.<br>The sum of snode.max and pnode.max cannot be greater than sess.total. | The default is **255**. |
| sess.default | The default session class for starting session managers. A Process executes on the specified class or any higher session class. The value for this parameter overrides the equivalent value in the local.node record. | The default is **1**. |
| comm.bufsize | The buffer size for transmitting data to and from a remote node. | The value for TCP/IP has no limit (up to 2,147,483,623).<br>For LU6.2, the maximum is 32000.<br>The default is **16000** bytes. |

| Parameter | Description | Value |
|---|---|---|
| tcp.api | The information needed to monitor connection requests from the CLI or API using TCP/IP. The host is the host name or IP address where Connect:Direct is running. The port identifies the communications port for Connect:Direct for UNIX. You can specify the host name or nnn.nnn.nnn.nnn and the port. This parameter is required. | host name is the name of the Connect:Direct host computer.<br>**nnn.nnn.nnn.nnn** is the IP address of the computer running Connect:Direct.<br>port identifies the communication port for Connect:Direct for UNIX. The format is a port name or nnn, where nnn is a decimal number. |
| tcp.api.bufsize | The buffer size for transmitting data to and from a Connect:Direct CLI/API. | This value has no limit. The default is **32768** bytes. |
| tcp.max.time.to.wait | The maximum time the local node waits for a message from the remote node when using TCP/IP. When the time expires, the Process is moved to the Timer queue and Connect:Direct attempts to re-establish a session with the remote node. When set to 0, wait time is unlimited unless limited by the operating system. | The value is in seconds. The default value is **180**. |
| runstep.max.time.to.wait | The maximum time to wait for remote run steps to complete. Remote run steps include remote run task, run job, or submit statements. This wait time is different from the wait time specified by the tcp.max.time.to.wait parameter. Using runstep.max.time.to.wait prevents a Process from failing when a remote step takes longer to complete than specified in tcp.max.time.to.wait. | The value is in seconds. The default value is **0**. |
| pacing.send.delay | The time to wait between send operations to the remote node. The decimal number is the number of milliseconds between the end of one packet and the beginning of the next packet. Time-based pacing does not contribute to network traffic.<br>The value for this parameter has no effect on LU6.2 connections. | The format is nnn. The size of this number has no limit. The default is **0**, which indicates no pacing of this type. |
| pacing.send.count | The number of send operations to perform before automatically waiting for a pacing response from the remote node. The value for this parameter has no effect on LU6.2 connections. | No limit exists for the size of this value. The default is **0**, which indicates no pacing of this type. |

| Parameter | Description | Value |
|---|---|---|
| netmap.check | Enhanced security testing performed on the SNODE. For TCP/IP connections, the remote IP address of the incoming socket connection is compared to the comm.info record of the netmap.cfg file. These values must match for a Connect:Direct session to be established. The comm.info record can be the official network name, an alias name listed in the appropriate file (for example, /etc/hosts, if the system is not running NIS or DNS), or the IP address. For SNA LU6.2 connections, the remote node name must be in the netmap.cfg. | **y**—Specifies that the security checks are made to verify that the remote node name is in the netmap.cfg file.<br>**n**—Specifies that none of these security checks are made. The default value is **n**. |
| proxy.attempt | Enables the ID subparameter of snodeid to contain a proxy, or dummy user ID to be used for translation to a local user ID on the remote system. Using a dummy user ID improves security because neither the local system nor the remote system requires a valid user ID from the other side. | **y**—Specifies that the remote users can specify a dummy user ID in snodeid parameter.<br>**n**—Specifies that the remote users cannot specify dummy user ID in snodeid parameter.<br>The default is **n**. |
|  | The following code illustrates the logic used to perform a security check for the user ID: |  |

```
if proxy.attempt = yes
  if snodeid (ID only or ID &
password) is specified

 if ID@PNODE found in userfile.cfg
       security checking OK
    else
if (ID, PSWD) is valid UNIX ID & password
          security OK
       else
          security checking failed
  else               /*snodeid is NOT specified */
    if submitter@PNODE found in userfile.cfg
       security OK
    else
       security checking failed
else    /*i.e. proxy.attempt = no, the default value */
  if snodeid (ID & password) is specified
    if (ID, PSWD) is valid UNIX ID & password
       security OK
    else
       security checking failed
  else               /*snodeid is NOT specified */
    if submitter@PNODE found in userfile.cfg
       security OK
    else
       security checking failed
```

| Parameter | Description | Value |
|---|---|---|
| outgoing.address | If using the high availability feature, this parameter enables you to specify an IP address for the remote node to use for network map checking and prevents the Process from failing when initiated from within a high availability environment. Specify the IP address for this value and network map checking verifies the address instead of the value set in comm.info in the SNODE network map record. | IP address |
| lu62.writex.wait | If you are using SNA on an IBM AIX operating system, use this parameter to identify how long to wait before retrying the connection. | The value in seconds. The default value is **1**. |
| lu62.writex.attempts | If you are using SNA on an IBM AIX operating system, use this parameter to identify how many times to attempt a connection. | The default value is **0**. |

## Updating TCP/IP Settings for a Local Node

The **tcp.ip.default** record defines default information to use when the remote node is specified by IP address. The **tcp.ip.default** record parameters are described in the following table:

| Parameter | Description | Value |
|---|---|---|
| conn.retry.stwait | The time to wait between retries immediately after a connection failure occurs. The format is hh.mm.ss, where **hh** specifies hours, **mm** specifies minutes, and **ss** specifies seconds. | The maximum value is limited to the highest value in the clock format, 23.59.59. The default is **00.00.30**, which is 30 seconds. |
| conn.retry.stattempts | The number of times to attempt connection after a connection failure occurs. | 0–9999. The default is **6**. |
| conn.retry.ltwait | The time to wait between long-term retry attempts after all short-term retry attempts are used. The format is hh.mm.ss, where hh specifies hours, mm specifies minutes, and ss specifies seconds. | 0–23.59.59.<br>The default is **00.10.00**, or 10 minutes. |
| conn.retry.ltattempts | The number of times to attempt a connection after all short-term retry attempts are used. | 0–9999. The default is **6**. |

| Parameter | Description | Value |
|---|---|---|
| comm.bufsize | The buffer size for transmitting data to and from a remote node. | The value for TCP/IP has no limit (up to 2,147,483,623).<br>For LU6.2, the maximum is 32000.<br>The default is **16000** bytes. |
| conn.retry.exhaust.action | Action to take after the specified short and long-term retries have been used. | hold—Places Processes in the Hold queue in Held in Error status, after all retry attempts are used. This is the default value.<br>delete—Causes the Processes to be deleted from the TCQ. |
| tcp.max.time.to.wait | The maximum time the local node waits for a message from the remote node when using TCP/IP. When the timer expires, the Process is moved to the Timer queue and Connect:Direct attempts to re-establish a session with the remote node. | The value in seconds. The default value is **180**.<br>When set to 0, wait time is unlimited unless limited by the operating system. |
| runstep.max.time.to.wait | The maximum time to wait for remote run steps to complete. Remote run steps include remote run task, run job, or submit statements. This wait time is different from the wait time specified by the tcp.max.time.to.wait parameter. Using runstep.max.time.to.wait prevents a Process from failing when a remote step takes longer to complete than specified in tcp.max.time.to.wait. | The value in seconds. The default value is **0**. |
| contact.name | The name of the administrator or operator. | Name of administrator or operator. |
| contact.phone | The phone number of the Connect:Direct administrator or operator. | 999-999-9999 |
| descrip | Comments to include as part of the record. | An unlimited string. |
| sess.total | The maximum number of concurrent connections between all nodes and the local node.<br>The sum of api.max.connects and sess.total cannot exceed the number of file descriptors available. This value is system dependent. | A 1–3 digit number.<br>The default is **255**. |
| sess.pnode.max | The maximum concurrent connections, where the local node is the initiator of the session. | The default is **255**. |

| Parameter | Description | Value |
|---|---|---|
| sess.snode.max | The maximum concurrent connections, where the local node is the secondary node in a session. | The default is **255**. |
| sess.default | The default session class for starting session managers. A Process executes on the specified class or any higher session class. The value for this parameter overrides the equivalent value in the local.node record. | The default is **1**. |
| pacing.send.delay | How long to wait between send operations to the remote node. The decimal number is the number of milliseconds between the end of one packet and the beginning of the next packet. Time-based pacing does not contribute to network traffic.<br>The value for this parameter has no effect on LU6.2 connections. | The format is nnn. The size of this number has no limit. The default is **0**, which indicates no pacing of this type. |
| pacing.send.count | The number of send operations to perform before automatically waiting for a pacing response from the remote node.<br>The value for this parameter has no effect on LU6.2 connections. | No limit exists for the size of this value. The default is **0**, which indicates no pacing of this type. |

## Updating Remote Node Connection Information

The remote node connection information record, **node.name**, includes remote node information. Update these parameters as necessary to define default values to use for a remote node connection or add a new set of parameters for each new remote node you define. Following are the remote node connection parameters:

| Parameter | Description | Value |
|---|---|---|
| alt.comm.outbound | Alternate communication address (communication path) used for outbound Processes. This parameter provides the alternate addresses for a remote node that has multiple NIC cards. When the local node is the PNODE, the alternate addresses are tried if an initial attempt to the primary address (specified in the comm.info parameter) fails. After a connection has been established, if the connection is subsequently lost, attempts to reestablish the connection through the retry mechanism use the same address as the initial connection.<br>When the local node is the SNODE, the alternate addresses are used in the Netmap check. | Fully qualified host name or IP address and port number.<br>A comma separates the list of alternate communication paths, and the list is processed from the top down. |
| conn.retry.stwait | Time to wait between retries immediately after a connection failure occurs. The format is hh.mm.ss, where **hh** specifies hours, mm specifies minutes, and **ss** specifies seconds. | The maximum value is limited to the highest value in the clock format, 23.59.59. The default is **00.00.30**, which is 30 seconds. |
| conn.retry.stattempts | Number of times to attempt connection after a connection failure occurs. | 0–9999. The default is **6**. |
| conn.retry.ltwait | Time to wait between long-term retry attempts after all short-term retry attempts are used. The format is hh.mm.ss, where hh specifies hours, mm specifies minutes, and ss specifies seconds. | 0–23.59.59.<br>The default is **00.10.00**, or 10 minutes. |
| conn.retry.ltattempts | Number of times to attempt a connection after all short-term retry attempts are used. | 0–9999. The default is **6**. |
| conn.retry.exhaust.action | Action to take after the specified short and long-term retries have been used. | hold places Processes in the Hold queue in Held in Error status, after all retry attempts are used. This is the default value.<br>delete causes the Processes to be deleted from the TCQ. |

| Parameter | Description | Value |
|---|---|---|
| runstep.max.time.to.wait | The maximum time to wait for remote run steps to complete. Remote run steps include remote run task, run job, or submit statements. This wait time is different from the wait time specified by the tcp.max.time.to.wait parameter. Using runstep.max.time.to.wait prevents a Process from failing when a remote step takes longer to complete than specified in tcp.max.time.to.wait. The value is in seconds. | The default value is **0**. |
| contact.name | The name of the Connect:Direct administrator or operator. | Name of administrator or operator. |
| contact.phone | The phone number of the Connect:Direct administrator or operator. | 999-999-9999 |
| descrip | Comments to include as part of the record. | An unlimited string. |
| sess.total | The maximum number of concurrent connections between all nodes and the local node.<br>The sum of api.max.connects and sess.total cannot exceed the number of file descriptors available. This value is system dependent.<br>If sess.total exceeds the number of sessions in the APS license key file, the license key file silently overrides this value. | A 1–3 digit number.<br>The default is **255**. |
| sess.pnode.max | The maximum concurrent connections, where the local node is the initiator of the session. Number of PNODE sessions cannot exceed the total number of sessions.<br>If sess.pnode.max is larger than sess.total, the value of sess.pnode.max is silently rounded down to the value of sess.total.<br>The sum of snode.max and pnode.max cannot be greater than sess.total. | The default is **255**. |

| Parameter | Description | Value |
|---|---|---|
| sess.snode.max | The maximum concurrent connections, where the local node is the secondary node in a session.<br>Number of SNODE sessions cannot exceed the total number of sessions. If sess.snode.max is larger than sess.total, then it is silently changed to the value of sess.total.<br>The sum of snode.max and pnode.max cannot be greater than sess.total. | The default is **255**. |
| tcp.crc | Turn on or off the CRC function for TCP/IP processes on the remote node. | y \| n |
| comm.info | The information needed to monitor connection requests from remote nodes using TCP/IP or LU6.2. This information refers to the network card that the local Connect:Direct node uses to monitor inbound requests. This value is required.<br>◆ For TCP/IP connections, specify the host name or the IP address and port number. If specifying IP address and port, separate parameters with a semicolon (;).<br>◆ For LU6.2 connections, identify the profile name to identify the name of an SNA configuration profile for the remote connection. Connect:Direct generates the default name, hostl1, during the customization procedure. For AIX SNA, hostl1 refers to the side information profile name. For HP SNA, SunLink SNA, and Brixton SNA, hostl1 refers to the SNA profile file located in the same directory as the configuration files. | For TCP/IP connections, specify the host name or the IP address and port number.<br>Set the IP address to monitor a specific adapter or set the value to 0.0.0.0, to monitor all adapters.<br>The default port is **1364**.<br>For LU6.2 connections, specify a profile name, up to 8 characters. |

| Parameter | Description | Value |
|---|---|---|
| alternate.comminfo | Provides support for sessions with multiple IP address environments, such as Connect:Direct/Plex OS/390. Use this parameter to list all IP addresses or host names that are part of the multiple IP address environment. For Connect:Direct/Plex, this list should include the address of each Connect:Direct/Server with a different IP address from the Connect:Direct/Plex Manager. If the IP address of the initiating node does not match the IP address specified on the comm.info parameter, the alternate.comminfo parameter is checked for other valid IP addresses. | host name\|nnn.nnn.nnn\|*<br>host name—Host name associated with the floating IP address. For example:<br>:alternate.comminfo=hops:\ (where hops is a machine on the local domain).<br>:alternate.comminfo=hops.csg.stercomm.com:\ (fully-qualified host name)<br>nnn.nnn.nnn.nnn—the IP address of a machine running Connect:Direct.<br>*—Accepts any IP address. This turns off IP address validation.<br>**Note:** b does not support partial pattern matches such as: *.mydomain.com, myplex??.mydomain.com. |
| comm.bufsize | The buffer size for transmitting data to and from a remote node. | The value for TCP/IP has no limit (up to 2,147,483,623).<br>For LU6.2, the maximum is 32000.<br>The default is **16000** bytes. |
| pacing.send.delay | The time to wait between send operations to the remote node. The decimal number is the number of milliseconds between the end of one packet and the beginning of the next packet. Time-based pacing does not contribute to network traffic.<br>The value for this parameter has no effect on LU6.2 connections. | The format is nnn. The size of this number has no limit. The default is **0**, which indicates no pacing of this type. |
| pacing.send.count | The number of send operations to perform before automatically waiting for a pacing response from the remote node. The value for this parameter has no effect on LU6.2 connections. | No limit exists for the size of this value. The default is **0**, which indicates no pacing of this type. |
| comm.transport | The transport protocol for the remote node. | Tcp—TCP/IP connections<br>lu62—AIX SNA LU6.2 connections<br>blklu62—Other LU6.2 connections |

Chapter 6

# Maintaining Access Information Files

This chapter contains information about the user authorization information file, the program directory, local and remote user information records, and the security exit.

## User Authorization Information File

In order for users to have access to Connect:Direct for UNIX and use Connect:Direct commands and statements, define a record for each user ID in the user authorization information file, called **userfile.cfg**. The user ID is the key to the local user information record. It must be a valid user ID on the local system and must be unique.

**Note:** Disable access to the software for a local user by deleting or commenting out the local user information record.

You can create a generic user ID by specifying an asterisk (*) as the user ID. If a user does not have a specific local user information record, the user authorizations will default to those specified in this generic record. If no generic local user information record is defined and no specific local user information record is defined for the user, the user cannot use Connect:Direct.

Connect:Direct may optionally use remote user information records to translate remote user IDs to valid local user IDs where Connect:Direct for UNIX is installed. If an snodeid parameter is not coded on the incoming Process, Connect:Direct for UNIX uses this proxy relationship to determine the rights of remote users to issue Connect:Direct commands and statements.

Connect:Direct for UNIX uses the asterisk (*) character to establish generic mappings that facilitate mapping remote user IDs to local user IDs. The asterisk matches the node name or the host name. For example, you can specify *@node name to map the remote user ID to all user IDs at one node name, specify id@* to map to a specific user ID at all node names, or specify *@* to match all users at all node names.

The following table displays sample remote user ID mappings to local user IDs using the special characters:

| Remote User ID | at | Remote Node Name | is mapped to | Local User ID | Result of Mapping |
|---|---|---|---|---|---|
| user | @ | * | = | | Remote user ID *user* on all remote nodes |
| * | @ | mvs.node3 | = | labs3 | All remote user IDs on remote node mvs.node3 are mapped to local user ID labs3. |
| * | @ | * | = | vip01 | All remote user IDs on all remote nodes are mapped to local user ID vip01. |

You can generate all the records through the script-based customization procedure or generate only one or two records and use a text editor to generate additional records. After customization, you may want to modify some of the parameters. Use **cdcust** to create a new user file or a text editor to modify the file as necessary.

## Sample User Authorization File

The following sample displays a user authorization file. In the sample, SAM1 is the remote user ID, MVS.SAM1.NODE is the remote node name, and sam is the local UNIX user ID.

```
SAM1@MVS.SAM1.NODE:\
  :local.id=sam:\
  :pstmt.upload=y:\
  :pstmt.upload_dir=/home/qatest/username/ndm/uploaddir:\
  :pstmt.download=y:\
  :pstmt.download_dir=/home/qatest/username/ndm/downloaddir:\
  :pstmt.run_dir=/home/qatest/username/ndm/rundir:\
  :pstmt.submit_dir=/home/qatest/username/ndm/submitdir:\
  :descrip=:
sam:\
  :admin.auth=y:\
  :pstmt.copy.ulimit=y:\
  :pstmt.upload=y:\
  :pstmt.upload_dir=/home/qatest/username/ndm/uploaddir:\
  :pstmt.download=y:\
  :pstmt.download_dir=/home/qatest/username/ndm/downloaddir:\
  :pstmt.run_dir=/home/qatest/username/ndm/rundir:\
  :pstmt.submit_dir=/home/qatest/username/ndm/submitdir:\
  :name=:\
  :phone=:\
  :descrip=:
```

## Updating the Local User Information Record Format

The local user record, userid, defines the default values for each user ID. Most of the parameters in the local user information record can take the following values:

- y—Indicates that a user can perform the function. In the case of process and select statistics commands, the user can affect Processes and view statistics owned by this user ID
- n—Indicates that a user cannot perform the function.
- a—Indicates that a user can issue commands for Processes owned by all users and generate statistics records for all users.

The following table defines the local user information parameters. The default values are underlined.

| Parameter | Description | Value |
|---|---|---|
| admin.auth | Determines if the user has administrative authority. If set to y, the user can perform all of the commands. If set to n, the specific command parameters must be granted individually. | y \| <u>n</u> |

| Parameter | Description | Value |
|---|---|---|
| cmd.chgproc | Determines if the user can issue the **change process** command. | y \| n \| a |
| cmd.delproc | Determines if the user can issue the **delete process** command. | y \| n \| a |
| cmd.flsproc | Determines if the user can issue the **flush process** command. | y \| n \| a |
| cmd.selproc | Determines if the user can issue the **select process** command. | y \| n \| a |
| cmd.viewproc | Determines if the user can issue the **view process** command. | y \| n \| a |
| cmd.selstats | Determines if the user can issue the **select statistics** command. | y \| n \| a |
| cmd.stopndm | Determines if the user can issue the **stop** command. | y \| n |
| cmd.submit | Determines if the user can issue the **submit process** command. | y \| n |
| cmd.trace | Determines if the user can issue the **trace** command. | y \| n |
| pstmt.crc | Enables the user to the override the initial settings to use the keyword CRC in a Process statement. | y \| n |
| descrip | Permits the administrator to add descriptive notes to the record. | unlimited text string |
| name | The name of the user. | user name |
| phone | The phone number of the user. | user phone number |
| pstmt.copy | Determines if the user can issue the **copy** statement. | y \| n |

| Parameter | Description | Value |
|---|---|---|
| pstmt.copy.ulimit | The action taken when the limit on a user output file size is exceeded during a copy operation. The value for this parameter overrides the equivalent value for the ulimit parameter in the initialization parameters file. | y \| n \| nnnnnnnn \| nnnnnnnnK \| nnnnnnnnM \| nnnnnnnnG<br>y—Honors the user file size limit. If this limit is exceeded during a copy operation, the operation fails.<br>n—Ignores the limit.<br>**nnnnnnnn**, **nnnnnnnnK**, **nnnnnnnnM**, or **nnnnnnnnG**—Establishes a default output file size limit for all copy operations. K denotes 1024 bytes. M denotes 1048576 bytes. G denotes 1073741824 bytes. The maximum value you can specify is 1 TB. |
| pstmt.upload | Determines if the user can send files from this local node. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. | y \| n |
| pstmt.upload_dir | The directory from which the user can send files. If a value is set for this parameter, then files can only be sent from this directory. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. If this parameter is defined, file names in Copy statements must be relative to this directory. Absolute path names cannot be used. | directory path name |
| pstmt.download | Determines if the user can receive files to this local node. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. | y \| n |
| pstmt.download_dir | The directory to which the user can receive files. If a value is set for this parameter, then files can only be received to this directory. Otherwise, they can be received from any directory. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. | directory path name |

| Parameter | Description | Value |
|---|---|---|
| pstmt.run_dir | The directory where Connect:Direct for UNIX is installed that contains the programs and scripts the user executes with **run job** and **run task** statements. Any attempt to execute a program or script outside the specified directory fails.<br>The UNIX Restricted Shell provides enhanced security by restricting the user to the commands contained in the pstmt.run_dir. If the user does not specify pstmt.run_dir, the commands are started with the Bourne shell.<br>To restrict the use of special characters in the run directory, be sure to configure Y for the **restrict:cmd** initialization parameter. For more information on specifying the **restrict:cmd** initialization parameter, see *Restricting the Use of Special Characters in the Run Directory* on page 31. | directory path name |
| pstmt.runjob | Specifies whether the user can issue the **run job** statement. | y \| n |
| pstmt.runtask | Specifies whether the user can issue the **run task** statement. | y \| n |
| pstmt.submit | Specifies whether the user can issue the **submit** statement. | y \| n |
| pstmt.submit_dir | The directory from which the user can submit Processes. This is for submits within a Process. | directory path name |
| snode.ovrd | Specifies whether the user can code the **snodeid** parameter on the **submit** command and **process** and **submit statements.** | y \| n |
| pstmt.crc | Gives the user the authority to specify the use of CRC checking in a Process statement.<br>Setting this parameter to y enables the user to override the initial settings in the initialization parameters or network map settings files. | y \| n |

If the same parameter is specified in the remote user information record and the local user information record, the parameter in remote user information record takes precedence unless it is a null value. When a null value is specified in the remote record, the local user record takes precedence.

## Updating the Remote User Information Record

The remote user information record contains a remote user ID and a remote node name that become the key to the record. The local.id parameter identifies a local user information record for this user. You must create a local user information record for the remote user.

**Note:** Prevent the remote user from using Connect:Direct by deleting or commenting out the remote user information, unless the remote user specifies an SNODEID parameter in the Process.

The remote user information record is remote userid@remote node name. It specifies the user and remote node name pair defined as a remote user. This value becomes the key to the record and must be unique. Create a remote user information record for each user on a remote node that will communicate with this local node.

Following are the parameters for the remote user information record:

| Parameter | Description | Value |
|---|---|---|
| local.id | The local user ID to use for security checking on behalf of the remote user. The **local.id** parameter must identify a local user information record. | local user ID |
| pstmt.copy | Determines if the user can issue the **copy** statement. | y \| <u>n</u> |
| pstmt.upload | Determines if the user can send files from this local node. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. | <u>y</u> \| n |
| pstmt.upload_dir | The directory to which the user can send files. If a value is set for this parameter, then files can only be sent to this directory. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. If this parameter is defined, file names in Copy statements must be relative to this directory. Absolute path names cannot be used. | directory path name |
| pstmt.download | Determines if the user can receive files to this local node. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. | <u>y</u> \| n |
| pstmt.download_dir | The directory from which the user can receive files. If a value is set for this parameter, then files can only be received from that directory. Otherwise, they can be received from any directory. If a file open exit is in use, this parameter is passed to the exit, but it is not enforced. | directory path name |

| Parameter | Description | Value |
|---|---|---|
| pstmt.run_dir | The directory that contains the programs and scripts the user can execute with **run job** and **run task** statements. Any attempt to execute a program or script outside the specified directory fails.<br>To restrict the use of special characters in the run directory, be sure to configure Y for the **restrict:cmd** initialization parameter. For more information on specifying the **restrict:cmd** initialization parameter, see *Restricting the Use of Special Characters in the Run Directory* on page 31. | directory path name |
| pstmt.submit_dir | The directory from which the user can submit Processes. This is for submits within a Process. | directory path name |
| pstmt.runjob | Specifies whether the user can issue the **run job** statement. | y \| n |
| pstmt.runtask | Specifies whether the user can issue the **run task** statement. | y \| n |
| pstmt.submit | Specifies whether the user can issue the **submit** statement. | y \| n |
| descrip | Permits you to add descriptive notes to the record. | text string |

## Updating the Strong Access Control File

To provide a method of preventing an ordinary user from gaining root access through Connect:Direct for UNIX, a strong access control file called **sysacl.cfg** is created at installation in the *d_dir*/ndm/SACL/ directory. By default, an ordinary user cannot access through Connect:Direct for UNIX. If you want to give an ordinary root user access through Connect:Direct for UNIX, you must access and update the **sysacl.cfg** file.

**Note:** Even if you do not want to limit root access through Connect:Direct for UNIX, the sysacl.cfg file must exist. If the file is deleted or corrupted, all users are denied access to Connect:Direct for UNIX.

The file layout of the sysacl.cfg file is identical to the user portion of the userfile.cfg file. Setting a value in the sysacl.cfg file for a user overrides the value for that user in the userfile.cfg file.

The **root:deny.access** parameter allows or denies root access to Connect:Direct for UNIX. To allow root access, specify the following in the sysacl.cfg file:

```
root:\
 deny.access=n:
```

If you want to deny root access, specify the following:

```
root:\
 deny.access=y:
```

If a user is denied access because the root:deny.access parameter is defined in the sysacl.cfg file for that user, a message is logged, and the session is terminated. If a user is running a limited ID, an informational message is logged.

## Automatic Detection of Shadow Passwords

Because shadow password files are available on some versions of the UNIX operating system, Connect:Direct for UNIX detects the use of shadow passwords automatically, if available.

## About the Program Directory

The program directory provides enhanced security for the run task and run job process statements by limiting access to specified scripts and commands. Any attempt to execute a program or script outside the specified directory fails. The program directory is identified with the **pstmt.run_dir** parameter. If the program directory is specified, the UNIX restricted shell is invoked, providing enhanced security. If the program directory is not specified, the regular (Bourne) shell is invoked for executing commands with no restrictions.

The restricted shell is very similar to the regular (Bourne) shell, but it restricts the user from performing the following functions:

- Changing the directory (cd)
- Changing PATH or SHELL environment variables
- Using command names containing a slash (/) character
- Redirecting output (> and >>)
- Executing programs

Additional information about the restricted shell can be found in the appropriate UNIX manual pages or UNIX security text books.

The restricted shell is started using only the environment variables HOME, IFS, PATH, and LOGNAME, which are defined as follows:

HOME=*run_dir*
IFS=*whitespace characters* (tab, space, and newline)
PATH=/usr/rbin and *run_dir*
LOGNAME=*user's UNIX ID*

Because environment variables are not inherited from the parent Process, no data can be passed to the script or command through shell environment variables. The restricted shell restricts access to specified scripts and commands, but it does not restrict what the scripts and commands can do. For example, a shell script being executed within the *run_dir* directory can change the value of PATH and execute command names containing a slash (/) character. For this reason, it is important that the system administrator controls which scripts and commands the user has access to and does not give the user write privileges to the *run_dir* directory or any of the files in the *run_dir* directory.

## Security Exit

The Security Exit in the initialization parameters file, **initparm.cfg**, provides an interface to password support programs.

This exit generates and verifies passtickets and it also supports other password support programs. An example of other programs is PASSTICKET, part of the RACF security system available on MVS hosts and also supported by IBM on UNIX AIX and OS/2 computers using the NETSP product.

Refer to Chapter 3, *Maintaining the Initialization Parameters File*, for information on the Security Exit.

Chapter 7

# Maintaining Client and Server Authentication Key Files

This chapter contains information about client and server authentication key files. You can edit both key files with any text editor installed on your system.

## About Client and Server Authentication Key Files

Connect:Direct client/server security depends on a key, similar to a password, in a Connect:Direct server and an identical key in each API that communicates with that server. The keys are defined and coordinated by the system administrator.

The client key file is called keys.client on the node on which the API resides. The server key file is keys.server on the node on which the server resides. The key files are located in the directory *d_dir*/security.

### Key File Format

A record in a key file can contain up to four keys that match entries in another API or server key file. The key file can contain as many key file records as necessary. The format of a key file entry is illustrated in the following sample:

```
  hostname    MRLN SIMP key [key [key [key] ] ]
```

## Updating Key File Parameters

The following are describes the available key file parameters:

| Parameter | Description | Value |
|---|---|---|
| hostname | The host name of the server with which you want to communicate or the host name of the API you will allow to communicate with your server. The hostname is followed by one or more space characters. If you replace the host name with an asterisk (*) character in the server configuration file, the server accepts a connection from any API with a matching key. You can use only one asterisk per file. Always place the entry with the asterisk after entries with specific host names. | 1—16 characters and must be unique within its key file. |
| MRLN SIMP | A required character string, separated from the other fields by one or more spaces. | Character string |
| key | The security key. Separate the key from SIMP by one or more spaces. | Up to 22 characters long including A to Z, a to z, 0 to 9, period (.), and slash (/). |

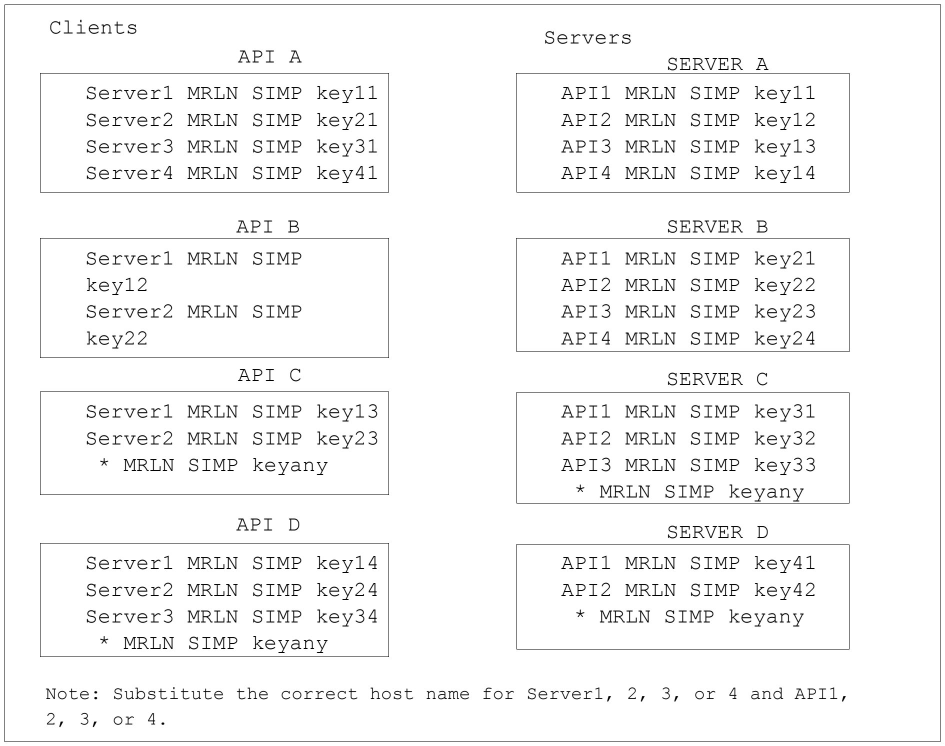
## Sample Client Authentication Key File

The following figure illustrates API key lists in the Clients column and server key lists in the Servers column.

✦ API A contains key11, key21, key31, and key41. Key11 enables API A to communicate with Server A because Server A also contains the key11 entry. You must ensure that API1 is the host name on which API A resides and that Server1 is the host name on which Server A resides.

✦ API D contains key14, key24, and key34. Key14 enables API D to communicate with Server A because Server A also contains the key14 entry. You must ensure that API4 is the host name on which API D resides and that Server1 is the host name on which Server A resides.

✦ API C can communicate with Server A and Server B through matching keys. API C also can communicate with Server C and Server D only through the *** MRLN SIMP keyany** line.

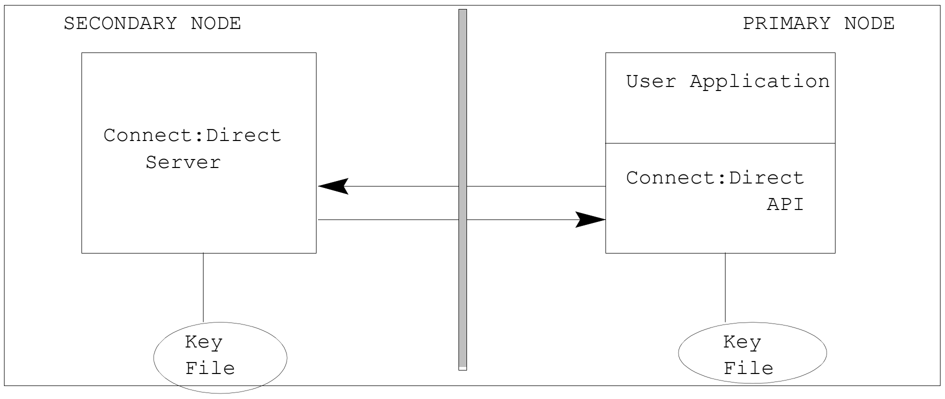


```
Clients                          Servers
API A                            SERVER A
   Server1 MRLN SIMP key11          API1 MRLN SIMP key11
   Server2 MRLN SIMP key21          API2 MRLN SIMP key12
   Server3 MRLN SIMP key31          API3 MRLN SIMP key13
   Server4 MRLN SIMP key41          API4 MRLN SIMP key14

API B                            SERVER B
   Server1 MRLN SIMP                API1 MRLN SIMP key21
   key12                            API2 MRLN SIMP key22
   Server2 MRLN SIMP                API3 MRLN SIMP key23
   key22                            API4 MRLN SIMP key24

API C                            SERVER C
   Server1 MRLN SIMP key13          API1 MRLN SIMP key31
   Server2 MRLN SIMP key23          API2 MRLN SIMP key32
    * MRLN SIMP keyany              API3 MRLN SIMP key33
                                     * MRLN SIMP keyany

API D                            SERVER D
   Server1 MRLN SIMP key14          API1 MRLN SIMP key41
   Server2 MRLN SIMP key24          API2 MRLN SIMP key42
   Server3 MRLN SIMP key34           * MRLN SIMP keyany
    * MRLN SIMP keyany

Note: Substitute the correct host name for Server1, 2, 3, or 4 and API1,
2, 3, or 4.
```

## About the Authentication Procedure

The Connect:Direct authentication procedure determines if the user is authorized to access the system.

The goal of Connect:Direct security is to reliably determine the identity of each user without requiring logon repetition. In addition, the security design ensures that all requests originate from the Connect:Direct API, to ensure that the authentication procedure is not bypassed by an unauthorized user. The following figure displays the components that perform authentication:



## Server Authentication Parameters

The server authentication parameters are specified in **initparm.cfg.** You must have ownership and permissions to modify these files. Ownership is established during the installation procedure.

Additionally, the directory containing the keys.server file must have UNIX permission 0700, and keys.server must have UNIX permission 0600. These files cannot be owned by root.

The following server authentication parameters are used by the CMGR during the authentication procedure:

| **Parameter** | **Description** |
|---|---|
| server.program | The server program to use during the authentication procedure. |
| server.keyfile | The key file to use during the authentication procedure. |

## Client Authentication Parameters

The client authentication parameters are specified in **ndmapi.cfg**. You must have ownership and permissions to modify these files. Ownership is established during the installation procedure.

Additionally, the directory containing the keys.client file must have UNIX permission 0700, and keys.client must have UNIX permission 0600.

The following client authentication parameters are used by the CLI/API during the authentication procedure:

| Parameter | Description |
|---|---|
| client.program | The client program to use during the authentication procedure. |
| client.keyfile | The key file to use during the authentication procedure. |

# Glossary

## A

### Application Programming Interface (API)

An application that enables End User Applications (EUAs) and the Connect:Direct Command Line Interface (CLI) to interact with the Connect:Direct software.

## C

### Client

The program that executes commands sent by the CLI/API and sends the results back to the CLI/API. Carries out the Connect:Direct authentication procedure in conjunction with API to determine access to Connect:Direct. Interacts with the PMGR when executing commands.

### Command Line Interface (CLI)

A program available to submit Connect:Direct Processes and commands from a command line environment.

### Command Manager (CMGR)

The program that executes commands sent by the API and sends the results back to the API. In conjunction with the API, the CMGR carries out the Connect:Direct authentication procedure, which determines if the user name and password are authorized to access the system. CMGR interacts with the PMGR when required by command execution.

### Connect Control Center

A centralized management system that provides operations personnel with continuous enterprise-wide business activity monitoring capabilities for Connect:Direct z/OS, UNIX, and Windows servers. It manages multiple Connect:Direct servers to suspend, release, and delete Processes, stops Connect:Direct servers, and views detailed statistics on running or completed Processes. It monitors service levels to view Connect:Direct processing across Connect:Direct z/OS, UNIX, and Windows servers within your network and retrieve information about active and completed

Processes. It receives notification of data delivery events that occur or do not occur as scheduled and defines rules that, based on processing criteria, can generate an alert, send an e-mail notification, generate a Simple Network Management Protocol (SNMP) trap to an Enterprise Management System (EMS), or run a system command. It monitors for alerts, such as a server failure or a Process not starting on time.

### Connect:Direct

The family of data transfer software products that distributes information and manages production activities among multiple data centers.

### Connect:Direct Browser User Interface

As an alternative to submitting Connect:Direct commands through the command line interface, you can use the Connect:Direct Browser User Interface to create, submit, and monitor Processes from an Internet browser, such as Microsoft Internet Explorer or Netscape Navigator. You can also use the Connect:Direct Browser to perform Connect:Direct system administration tasks, such as viewing and changing the network map or initialization parameters, if you have the appropriate Connect:Direct authority.

### Connect:Direct for UNIX

The UNIX implementation of the Connect:Direct product.

### Connect:Direct Node

Any computer/workstation running Connect:Direct.

### Connect:Direct Process

A series of statements, which can be predefined and stored in a directory, submitted through the API to initiate Connect:Direct for UNIX activity. Examples of Process functions are copying files and running jobs.

## D

### daemon

The long-running process that provides a service to a client. The PMGR is the Connect:Direct for UNIX daemon.

### Diagnostic Commands

Connect:Direct commands that assist in the diagnosis of Connect:Direct software problems.

## E

### End User Application (EUA)

An application program developed by an end user to accomplish a particular task.

### Execution Queue

A logical queue in the TCQ. A Process in the Execution Queue can be transferring data to or from a remote Connect:Direct node or it can be waiting for a connection to the remote Connect:Direct node before it can perform its tasks.

## F

### File Agent

An application program and component of Connect:Direct. It scans specified directories searching for the presence of a file. When a file appears in a watched directory, Connect:Direct either submits a Process or performs the action specified by the rules for the file.

## H

### Hold Queue

A logical queue in the TCQ. Processes in the Hold Queue are waiting for operator intervention before they move to the Wait Queue for scheduling.

## M

### Monitoring Commands

Connect:Direct commands that allow you to display information from the statistics file and the TCQ about Connect:Direct Process execution results.

## O

### Operational Control Commands

Connect:Direct commands that allow you to submit a Process, change specific characteristics of a Process in the TCQ, remove executing and nonexecuting Processes from the TCQ, and stop Connect:Direct.

## P

### Process Manager (PMGR)

The long-running Connect:Direct server that initializes the Connect:Direct software, accepts connection requests from Connect:Direct APIs and remote Connect:Direct nodes, creates Command Managers and Session Managers, accepts requests from Command Managers and Session Managers where centralized Connect:Direct functions are required, and terminates Connect:Direct software execution.

### PNODE (Primary Node)

The Connect:Direct node on which the Process is being executed. The primary node is also called the controlling or source node, but is not always the sending node because PNODE can be the receiver. Every Process has one PNODE and one SNODE. The submitter of a Process is always the PNODE. If defined in the **netmap.cfg** file, the PNODE name can be 1–16 characters long. If not defined in the **netmap.cfg** file, the PNODE name can be 1–256 alphanumeric characters long.

## S

### Session

A connection between two Connect:Direct nodes.

### Session Manager (SMGR)

The server component responsible for creating or completing a connection with a remote Connect:Direct node and carrying out the Connect:Direct work to be performed.

### SNODE (Secondary Node)

The Connect:Direct node that interacts with the Primary node (PNODE) during Process execution. The Secondary node (SNODE) also can be referred to as the participating, target, or destination node. Every Process has one PNODE and one SNODE. The secondary node is the node participating in Process execution initiated by another node (PNODE). If defined in the **netmap.cfg** file, the SNODE name can be 1–16 characters long. If not defined in the **netmap.cfg** file, the SNODE name can be 1–256 alphanumeric characters long.

## T

### TCQ Status Value

A two-letter code assigned to a Process by Connect:Direct when the Process is placed on the TCQ. The status of a Process can be examined with a **select process** command.

### TCQ (Transmission Control Queue)

A queue that holds all Processes that are submitted to Connect:Direct for UNIX. TCQ contains the following four logical queues:

- EXECUTION
- WAIT
- TIMER
- HOLD

### Timer Queue

A logical queue in the TCQ. Processes on the Timer Queue are waiting for a start time before they move to the Wait Queue for scheduling.

## W

### Wait Queue

A logical queue in the TCQ. Processes on the Wait Queue are waiting on a connection to or from the remote Connect:Direct node.

# Index