IBM Connect:Direct for UNIX
6.2

Documentation

IBM

Contents

Chapter 1. Release Notes

The IBM® Connect:Direct® for UNIX Release Notes document supplements Connect:Direct for UNIX documentation. Release notes are updated with each release of the product and contain last-minute changes and product requirements, as well as other information pertinent to installing and implementing Connect:Direct for UNIX.

New Features and Enhancements

IBM Sterling Connect:Direct for UNIX 6.2 and its related software have the following features and enhancements:

Attention: IBM Sterling Connect:Direct will end support for HP-UX platform on 31st December 2025. After this date, security fixes for IBM Java 8 on HP-UX will no longer be available. The following products are affected:

- IBM Sterling Connect:Direct for UNIX
- IBM Sterling Connect:Direct File Agent
- IBM Sterling Connect:Direct FTP+

Attention: IBM Sterling Connect:Direct will end support for Oracle Solaris on 31st December 2025. After this date, security fixes for IBM Java 8 on Solaris will no longer be available. The following products are affected:

- IBM Sterling Connect:Direct for UNIX
- IBM Sterling Connect:Direct Web Services
- IBM Sterling Connect:Direct File Agent
- IBM Sterling Connect:Direct FTP+

FixPack 7 (v6.2.0.7)

New Features or Enhancements
To install this software, you should go to the Fix Central and follow instructions described to complete the download. Connect:Direct for UNIX has the following features and enhancements: • Support for performing pre and post-upgrade actions from Control Center Director has been added to Connect:Direct for UNIX.

FixPack 6 (v6.2.0.6)

New Features or Enhancements
To install this software, you should go to the Fix Central (Traditional Software) or “Installing IBM Connect:Direct for UNIX using an IBM Certified Container Software” on page 39 website, and follow instructions described to complete the download. Connect:Direct for UNIX has the following features and enhancements: • IBM Certified Container Software for Connect:Direct for UNIX has been re-certified for next one year. It has been deployed and tested on AWS EKS v1.24 • The container image has been upgrade to latest UBI base image v8.7. **Attention:** The **`appuser.uid`** and **`appuser.gid`** has been deprecated from this fixpack. It is not recommended to use it. The support of these parameters will be completely removed from v6.3 and onwards.

FixPack 5 (v6.2.0.5)

New Features or Enhancements
To install this software, you should go to the Fix Central (Traditional Software) or “Installing IBM Connect:Direct for UNIX using an IBM Certified Container Software” on page 39 website, and follow instructions described to complete the download. Connect:Direct for UNIX has the following features and enhancements: • Connect:Direct for UNIX support for IBM Certified Container Software persistent data storage on NFS with root squash enabled. For more details refer to “Creating storage for Data Persistence” on page 61. **Attention:** The **`cduser.uid`** and **`cduser.gid`** has been deprecated from this fixpack. It is not recommended to use it. The support of these parameters will be completely removed from v6.3 and onwards.

FixPack 4 (v6.2.0.4)

New Features or Enhancements
To install this software, you should go to the Fix Central website, and follow instructions described to complete the download. Connect:Direct for UNIX has the following features and enhancements: • Connect:Direct for UNIX support for transferring data to and from Google Cloud Storage, Azure Blob, IBM Cloud Object Storage. This is in addition to the existing support for AWS S3 and S3-compatible Object Stores. For more information, refer to AWS Cloud Overview.

FixPack 3 (v6.2.0.3)

New Features or Enhancements
To install this software, you should go to the Fix Central website, and follow instructions described to complete the download. Connect:Direct for UNIX has the following features and enhancements: • Connect:Direct for UNIX introduces support for Port Check Ignore List (configured in initparms), which would terminate port check connections efficiently from addresses configured under the Ignore List. For more information, refer to Health Check. **Note:** Connect:Direct for UNIX does not write stats for health check connections.

FixPack 2 (v6.2.0.2)

New Features or Enhancements
To install this software, you should go to the Fix Central website, and follow instructions described to complete the download.
Connect:Direct for UNIX has the following features and enhancements:
• Standalone Docker support will be revoked from this release. It can still be used for versions prior to 6.2.0.2.
• IBM Certified Container Software has been re-certified for the next year.
• IBM Certified Container Software (helm chart) supports Integrated File Agent. Use values.yamlconfiguration to enable it inside a pod.
• Process library feature has been introduced in IBM Certified Container Software.

Base Release (v6.2)

New Features or Enhancements
To install this software, you should go to the Passport Advantage website, and follow instructions described to complete the download.
Connect:Direct for UNIX has the following features and enhancements:
• **IBM Sterling Integrated File Agent**
– With this release, Connect:Direct for UNIX introduces support for IBM Sterling Integrated File Agent as a component of Connect:Direct UNIX (selectable during installation). Integrated File Agent is simpler to install, configure and operate than its predecessor, Standalone File Agent. Use Connect:Direct Web Services 6.2 UI or later to configure Integrated File Agent. For more information, refer “cd.file.agent Record” on page 108 and Integrated File Agent.
– When Integrated File Agent is installed, if Secure+ .Local record has not already been assigned a key certificate, an automatically generated self-signed certificate is assigned to Secure+'s .Local record. For more information refer Self-signed Certificate for Integrated File Agent.
– An initparm to specify the location of the directory, which acts as the process library for Integrated File Agent has been added. For more information, refer Sterling Connect:Direct File Agent Overview and Path record.
– You can upgrade from an existing Standalone File Agent to Integrated File Agent, interactively or silently. For more information, refer Upgrading to Connect:Direct Integrated File Agent.
– Support for external stat logging has been added for use of Integrated File Agent. For more information, refer External stat logging.
• SSTA and STOP parameters for API functions (**ndmapi_recvresp()** or **ndmapi_recvresp_c()**) have been deprecated and will be removed in the future releases. Use STRT and STPT instead of SSTA and STOP respectively. For more details, refer Receiving Responses Using ndmapi_recvresp() or ndmapi_recvresp_c().
• OpenShift version has been updated to 4.7.
• While deploying IBM Connect:Direct for UNIX using an IBM Certified Container Software, Dynamic and Non Dynamic Provisioning are supported while creating storage for data persistence. For more information, refer Creating storage for Data Persistence.
• zFBA feature is deprecated in Connect:Direct version 6.2.

Hardware and Software Requirements

IBM® Connect:Direct® for UNIX and its related software require the following hardware and software: It supports systems running in 64-bit mode.

Attention: IBM Sterling Connect:Direct ends support for Oracle Solaris on 30 September 2025. After this date, security fixes for IBM Java on Solaris will no longer be available. The following products are affected:

- IBM Sterling Connect:Direct for UNIX
- IBM Sterling Connect:Direct File Agent
- IBM Sterling Connect:Direct FTP+

Component or Functionality	Hardware	Software	RAM (min.)	Disk Space (min.)
IBM Connect:Direct for UNIX with TCP/IP or FASP connectivity	HP Integrity system with Intel Itanium processor	HP-UX version 11iv3 or higher **Note:** Not supported with FASP.	2 GB	1.5 GB†
	IBM System pSeries, POWER7 or greater processor required	AIX versions 7.1, 7.2, and 7.3 **Note:** AIX 7.3 support requires IBM Sterling Connect:Direct for UNIX v6.2.0.5 or greater.	2 GB	1.5 GB†
	IBM System pSeries, POWER8 or greater processor required	SuSE Linux Enterprise Server (ppc64le) version 12.4 and greater. **Note:** Not supported with FASP.	2 GB	1.5 GB†
	Sun SPARC system	Solaris 11. **Note:** Not supported with FASP.	2 GB	1.5 GB†

† When upgrading IBM Connect:Direct for UNIX through Control Center Directer, an extra 3 GB is required for temporary storage.

Component or Functionality	Hardware	Software	RAM (min.)	Disk Space (min.)
	Intel and AMD x86-64	Red Hat Enterprise Linux version 7.8, 7.9. Red Hat Enterprise Linux version 8.1 and above[†††] Red Hat Enterprise Linux version 9.2 and above[†††]	2 GB	1.5 GB[†]
		CentOS version 7.8, 7.9.[††] **Note:** Not supported with FASP.		
		Amazon Linux 2.[††] **Note:** Not supported with FASP.		
		Ubuntu version 18 and above[†††] Ubuntu version 22.04 and above.[†††] **Note:** Not supported with FASP.	2 GB	1.5 GB[†]
		SuSE Linux Enterprise Server version 12.3 and above or 15.x.[†††]	2 GB	1.5 GB[†]

[††] IBM does not formally test and certify IBM Connect:Direct on CentOS and Amazon Linux 2. However as CentOS and Amazon Linux 2 are derived from the sources of Red Hat Enterprise Linux (RHEL), we believe that the product should work correctly. IBM will investigate and troubleshoot a problem until it is determined that the problem caused by a difference in behavior between CentOS, Amazon Linux 2 and RHEL. Defect support will only be available for problems that can be reproduced on a certified platform as documented in the Software Product Compatibility Reports (link: https://www.ibm.com/software/reports/compatibility/clarity/index.html?lnk=uctug_ratl_dw_2013-02-01_clarity_updated).

Component or Functionality	Hardware	Software	RAM (min.)	Disk Space (min.)
	Linux® zSeries	Red Hat Enterprise Linux version 7.8, 7.9 Red Hat Enterprise Linux version 8.1 and above.[†††]	2 GB	1.5 GB[†]
		SuSE Linux Enterprise Server version 12.3 and above or 15.x.[†††] **Note:** Not supported with FASP.	2 GB	1.5 GB[†]
Connect:Direct Integrated File Agent	Same as requirements for Connect:Direct for UNIX	Same as requirements for IBM Connect:Direct for UNIX.	2 GB	275 MB
Connect:Direct Secure Plus	Same as requirements for IBM Connect:Direct for UNIX.	Same as requirements for Connect:Direct Secure Plus. Java™ Standard Edition 8, installed with Connect:Direct Secure Plus.	2 GB	70 MB
High-Availability support	IBM System pSeries, POWER7 or greater processor required	IBM HACMP		
	Sun SPARC system	SunCluster 2.2, 3.0 or 3.2		

Virtualization and public cloud support

IBM cannot maintain all possible combinations of virtualized platforms and cloud environments. However, IBM generally supports all enterprise class virtualization mechanisms, such as VMware ESX, VMware ESXi, VMware vSphere, Citrix Xen Hypervisor, KVM (Kernel-based virtual machine), and Microsoft Hyper-V Server.

[†††] Due to a system library change on more recent versions of Linux, such as Red Hat Enterprise Linux version 8, you must set your current working directory (CWD) to {CDU install dir}/ndm/bin to invoke the executable modules there. To invoke these modules from another CWD, there are two options:

- As root, create the following symbolic link:
  - For RHEL and SLES systems, in/lib64: ln -s /lib64/libtirpc.so.3 /lib64/libtirpc.so.1
  - For Ubuntu systems, in /lib/x86_64-linux-gnu: ln -s /lib/x86_64-linux-gnu/libtirpc.so.3 /lib/x86_64-linux-gnu/libtirpc.so.1
  - **Note**: Interactive (cdinstall) installations and upgrades done from current maintenance will check to see if this symbolic link is needed and offer to add it for you when performing configurations requiring root privilege. Automated (cdinstall_a) installations and upgrades can achieve the same result via the cdai_tirpcCreateLink parameter, set to ‘y’ or ‘n’.
- Set the environment variable LD_LIBRARY_PATH={CDU install dir}/ndm/lib for the user that starts Connect:Direct.

IBM investigates and troubleshoots a problem until it is determined that the problem is due to virtualization. The following guidelines apply:

- If a specific issue is happening because the system is virtualized and the problem cannot be reproduced on the non-virtualized environment, you can demonstrate the issue in a live meeting session. IBM can also require that further troubleshooting is done jointly on your test environment, as there is not all types and versions of VM software installed in-house.
- If the issue is not able to be reproduced in-house on a non-virtualized environment, and troubleshooting together on your environment indicates that the issue is with the VM software itself, you can open a support ticket with the VM software provider. IBM is happy to meet with the provider and you to share any information, which would help the provider further troubleshoot the issue on your behalf.
- If you chose to use virtualization, you must balance the virtualization benefits against its performance impacts. IBM does not provide advice that regards configuring, administering, or tuning virtualization platforms.

Support policy for Container Delivery Models

The support policies for container delivery models are as follows.

Support statement for IBM Connect:Direct for UNIX certified containers for Red Hat that are deployed using OpenShift Container Platform

IBM Connect:Direct for UNIX certified containers for Red Hat are built to deploy on the Red Hat OpenShift Container Platform. The product containers and deployment model are certified by IBM to be production ready, enterprise-grade, resilient, secure and compliant in many public and private clouds which the OpenShift Container Platform supports. IBM Technical Support supports this delivery model across the lifecycle management of IBM Connect:Direct for UNIX certified containers, including container orchestration scripts in OpenShift Container Platform and product documentation.

Support statement for IBM Connect:Direct for UNIX certified containers deployed on non-OpenShift Container Platform

For users who choose to deploy the IBM certified containers on; proprietary container orchestration tools such as EKS/GKE/AKS/PCF, on public cloud such as Amazon/Azure/ Google or in their private cloud using native Kubernetes, the IBM Technical Support is limited to the base Connect:Direct for UNIX software, certified containers, and HELM package manager. IBM provides limited support for the container editions that are deployed in proprietary renderings for Kubernetes. The non-conformant characteristics of such tools hinder the ability for IBM to assist users in all scenarios.

Support policy statement for containers that are created by users

For users who have created custom docker containers for IBM Connect:Direct for UNIX and have deployed in any Kubernetes platform, the IBM Technical Support is limited to the technical inquiries in the core Connect:Direct for UNIX. IBM recommends using IBM provided certified containers and not the user created containers for Connect:Direct for UNIX.

Known Restrictions

Connect:Direct for UNIX has the following restrictions when using third-party hardware or software:

- On HP-UX, Port Check Ignore List does not apply to the API port. Port checks on the API port on HP-UX result in error messages in Statistics. On Solaris, Support for Port Check Ignore List on API port is available from version 6.2.0.4 iFix08 onwards.
- When performing a new or upgrade installation of Connect:Direct with Control Center Director, Integrated File Agent cannot be selected as an option.

- An issue occurs which causes invalid data to be written to the destination file when standard compression is enabled and transfer is text mode when sending to another Connect:Direct Unix node. This issue leads to inadvertent conversion of some spaces to EBCDIC space instead of ASCII. A possible work-around of this issue is to use extended compression or no compression or use binary mode.
- If you use the Hummingbird Exceed terminal emulator to access a Solaris workstation, you may not have all of the fonts needed to use Connect:Direct Secure Plus IBM Connect Direct for UNIX. Add the following command to the spadmin.sh file:

  ```
  xset fp default
  ```

  Insert this command before the following line of code:

  ```
  java -classpath $CLASSPATH:/SCI/USERS/... com.stercomm.csg.spadmin.spadmin
  ```

  This command maps all unknown fonts to a default value and prevents IBM Connect:Direct for UNIX from performing a core dump if it is unable to locate a font.
- Connect:Direct Secure Plus Connect Direct for UNIX is administered through Java and a graphical user interface (GUI). The standard UNIX telnet server does not support a GUI client session. To use the UNIX GUI you must be connected to the UNIX server via an X Windows client session, such as xterm. If you are connected to the UNIX server using a telnet session, you will not be able to run the GUI sessions required to install and administer IBM Connect:Direct for UNIX. If you do not have access to X Windows, you can use the Connect:Direct Secure Plusfor UNIX Command Line Interface (Secure+ CLI).
- Connect:Direct Secure Plus IBM Connect Direct for UNIX does not support server gated crypto (SGC) certificates.
- The Secure+ CLI does not support using $HOME or the tilde (~) to specify the path to your home directory.
- When using the Secure+ CLI on the Solaris platform, command entries may be limited by the buffer size. To resolve this limitation, add line breaks to a command entry. For example, enter the following command with line breaks:

  ```
  SslTlsEnableCipher=(TLS_RSA_WITH_AES_256_CBC_SHA,
  TLS_RSA_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_RC4_128_MD5,
  SSL_RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
  SSL_RSA_WITH_DES_CBC_SHA,SSL_RSA_EXPORT_WITH_RC4_40_MD5,
  SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,SSL_RSA_WITH_NULL_MD5);
  ```
- On the HP-UX, IBM System pSeries, and Linux platforms, when a run task defines an invalid UNIX command, the operating system return code is 127 and the completion code (CCOD) reported by Connect:Direct for UNIX is displayed in hexadecimal (7F) in the statistics output. This return code is correct for the error received, even though most return codes are defined as 0, 4, 8, or 16.

  If the return code value of 127 is the highest step return code, the Process End (PRED) statistics record message ID is set to the Message ID of the run task step. On other platforms, the run task return code is 1, resulting in the message ID of XSMG252I in the PRED statistics record.
- Connect:Direct Browser User Interface IBM Connect Direct for UNIX is not supported running on HP Integrity systems with Intel Itanium processors.
- Installation on Linux platforms displays the following message:`awk: cmd. line:6: warning: escape sequence `\.' treated as plain `.'`

  This is a known issue with Install Anywhere and does not affect installation or functionality of Connect:Direct File Agent IBM Connect Direct for UNIX on Linux.
- When IBM Connect:Direct for UNIX 6.1 attempts a session as a PNODE with CD i5/OS and CD i5/OS is running on IBM i 7.3 or earlier, the session may fail if IBM Connect:Direct for UNIX 6.1 has TLS 1.3 enabled for the session. The resolution is to disable TLS 1.3 for sessions with the IBM Connect:Direct for UNIX i5/OS snode.
- Installation of Integrated File Agent via IBM Sterling Control Center Director is not supported currently as it does not support the specification of silent installation parameters for an installation.

- Connect:Direct for UNIX 6.2's interactive and silent installations support conversion of a Standalone File Agent installed by an earlier version of Connect:Direct for UNIX to an Integrated File Agent. This functionality will not be available through IBM Sterling Control Center Director because it uses silent installation of Connect:Direct for UNIX. Whether conversion occurs or not is controlled by a silent installation parameter whose default value is "no". Since, Control Center Director does not support the specification of silent installation parameters during upgrade, the conversion is not available through Control Center Director.
- On HPUX, Secure+ transfers may fail with error "`Secure+ initialization failure, gsk_environment_init() failed`" when there is a `GSKit` installed globally.

  Note: When this happens, the user will have to take care of the following restriction on HP-UX:

  - A setuid executable (Connect Direct server in this case), when executed by a non-root user cannot load libraries (`GSKit` in this case) from any path other than standard system paths. For more information, refer to `man dld.so` on HP-UX.
  - A compatible `GSKit` is shipped with Connect:Direct for UNIX and gets installed at a non-standard system path. Connect Direct server will need help from the root user to load the compatible `GSKit` libraries:

    1. Create (if it does not exist) `/etc/dld.sl.conf` and make it writable by root ONLY.
    2. `/etc/dld.sl.conf` must contain a ":" separated list of following paths:
       - `${ndm.path}/ndm/lib/gsk/lib64/`
       - `${ndm.path}/ndm/lib/`
    3. Ensure that the above 2 paths exist.

    Note: Get the `${ndm.path}` from `initparm.cfg`

Upgrade Considerations

If you are upgrading from an existing version of Connect:Direct for UNIX, observe the following guidelines:

- SNMP is no longer supported. If you are using SNMP and upgrade to this version, other functionality will not be negatively impacted. However, you will no longer receive SNMP messages.
- Change the ownership on the statistics files in your `work` directory so that these files are owned by the user who starts the cdpmgr daemon. Use the following command sequence to change the ownership of the statistics files:

```
$ su root
Password: root_password
# cd cddir/work/node
# chown user_who_starts_cdpmgr S*.???
```

  The upgrade considerations for installing Connect:Direct for UNIX on EC2 are described in section " Prerequisites for activating Connect:Direct Unix on AWS cloud".

  The following variable definitions apply:

  - root_password - Root user's password
  - cddir - Directory in which Connect:Direct for UNIX is installed
  - node - Your Connect:Direct for UNIX node name
  - user_who_starts_cdpmgr - User name of the user who will start the cdpmgr daemon
- If you are upgrading on HPUXIT and SOLARIS, the netmap must be reconfigured. Add `rpc.pmgr.port` to the `local.`node entry in the `netmap.cfg` file. If this entry is not added, the RPC Server utilizes port 1367 by default. If the default port is busy, the connection with Connect:Direct Server does not come up.

Special Considerations

This section contains considerations in addition to the procedures defined. Refer to the following notes before installing the product.

- Although Connect:Direct for UNIX Process names can be up to 255 characters long, some IBM Connect:Direct platforms, such as Connect:Direct for z/OS® limit Process names to eight characters. Processes running between UNIX and platforms that limit Process names to eight characters can have unpredictable results if longer names are specified.
- The Connect:Direct Secure Plus initparms record and the user authority that have been added to Connect:Direct for UNIX version 4.1.00 support remote configuration of Connect:Direct Secure Plus. These configuration options are needed when using the Central Management feature available in Control Center 5.0 and later.
- If you are using a certificate that was created with an older version of Certificate Wizard, the certificate may contain a blank line between the "BEGIN" and "END" statements that define a private key. This version cannot process the blank line, resulting in an error. If a certificate generates an error, delete the blank line in the certificate.
- For Integrated File Agent installation, Secure+ installation is mandatory. If Secure+ is not already installed, Connect:Direct for UNIX installer will install Secure+ before installing Integrating File Agent.
- Customers may desire to replace configuration files or the directories created by configuration, namely `ndm/cfg/{CDU node name}` and `work/{CDU node name}`, with symbolic links. As long as the redirection points to a supported file system as documented in the "Installing Connect:Direct for UNIX" on page 18 section, we will provide best effort support for these symbolic links. This means that if an issue is reported, we will not deny support solely based on the existence of these symbolic links. However, if Support determines that the symbolic link is a factor in the issue, you may be asked to modify or delete the link to resolve the issue.

Special Considerations for Connectivity with the HP NonStop Kernel Operating System

This version of Connect:Direct for UNIX offers connectivity to Connect:Direct for HP NonStop Kernel version 3.2.00 or later using TCP/IP. Refer to the following notes when transferring files from the UNIX operating system to the HP NonStop Kernel operating system:

- Do not define the sysopts parameter with continuation marks. Type the text in a continuous string, with blanks separating each subparameter. The sysopts parameter is valid for the copy statement.
- When copying files from the UNIX operating system to the HP NonStop Kernel operating system, define the dcb parameter to allocate destination files. Define any additional options using the sysopts parameter. The dcb and sysopts parameters are valid for the copy statement.

  Use of the dcb parameter ensures that the attributes of the file being sent match the attributes of the file that is created on the remote node. If you do not define the dcb parameter, the default file types on the destination node are as follows:

  - If you are transferring a text file, the file type on the HP NonStop Kernel node defaults to an unstructured file, code 101.
  - If you are transferring a binary file, the file type on the HP NonStop Kernel node defaults to an unstructured file, code 0.
- When copying files from the HP NonStop Kernel operating system to the UNIX operating system, define the sysopts parameter to allocate destination files.

For syntax and parameter descriptions for Process statements, see the Connect:Direct Processes Web site.

Connect:Direct for UNIX Guidelines

Before you install Connect:Direct for UNIX, read all the information in this section and follow the guidelines.

- Print and Review *IBM Connect:Direct for UNIX Implementation Guide*.
- To install Connect:Direct Secure Plus at the same time that you install Connect:Direct for UNIX, following the instructions in *IBM Connect:Direct for UNIX Getting Started Guide*.
- When you upgrade from a previous version, the parameters file is converted and can be used with the new version.

Libraries to Install

Ensure that you have the following libraries installed:

UNIX Platform	Software	Library
Intel and AMD x86-64, Linux zSeries	All supported Linux	• libtirpc (for more information on libtirpc, refer to "Hardware and Software Requirements" on page 4) • 64 bit libstdc++ x86-64
Linux zSeries	All supported Linux.	• libtirpc (for more information on libtirpc, refer to "Hardware and Software Requirements" on page 4)
HP Integrity system with Intel Itanium processor	HP-UX version 11iv3 or higher	• For HP-UX systems that use the /etc/shadow password file, the Password Hash Infrastructure (PHI) package, beginning with Connect:Direct for UNIX 6.2.0.1.iFix022 **Note:** To check for package installation status: • **11iv3 (B.11.31)**: `swlist -a state SHA11i3` • **11iv2 (B.11.23)**: `swlist -a state SHA` **Note:** You can download and install the package as needed: • **11iv3 (B.11.31)**: https://myenterpriselicense.hpe.com/cwp-ui/free-software/PHI11i3 • **11iv2 (B.11.23)**: https://myenterpriselicense.hpe.com/cwp-ui/free-software/PHI

Chapter 2. Getting Started Guide

The getting started with IBM Connect:Direct for UNIX workflow describes steps to set it up. This workflow describes how to plan, install, and manage a Connect:Direct over UNIX deployment.

Connect:Direct for UNIX Overview

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

Note: The connections between some clients and a Connect:Direct Server are unsecure. Passwords sent by one of these clients to a C:D Server are obfuscated, but the session is not encrypted. The clients are: UNIX CLI or any user-written UNIX ndampi client, the CD Requester, the Windows CLI, any user-written Windows SDK client and FileAgent.

Process Manager

The Process Manager (PMGR) is the daemon that initializes the Connect:Direct for UNIX server environment. Any application, including End User Applications (EUA), can run on any computer as long as it can connect to the PMGR. The PMGR provides the following functions:

- Initializes Connect:Direct for UNIX
- Accepts connection requests from Connect:Direct for UNIX 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 for UNIX functions are required
- Stops Connect:Direct for UNIX

  Command Manager

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 for UNIX operation. You must define enough file descriptors to handle the number of concurrent Connect:Direct for UNIX 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 for UNIX authentication procedure, in conjunction with the API, to determine access to Connect:Direct for UNIX
- 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 for UNIX 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.

User Authorization

Connect:Direct for UNIX can authorize local and remote users to perform certain Connect:Direct for UNIX tasks. In order to use Connect:Direct for UNIX, 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 remote users may 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 (SACL) file is created when you install Connect:Direct for UNIX and is named sysacl.cfg. The root:deny.access parameter, which is specified in the sysacl.cfg file, allows, denies, or limits root access to Connect:Direct for UNIX. If the SACL file is deleted or corrupted, access to Connect:Direct for UNIX is denied to all users.

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 for UNIX 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. If it is not configured in the **copy** step, it can be configured in the Initparms through the **ckpt.interval** parameter.
- Run Task restart—If a Process is interrupted when a run task on an SNODE step is executing, Connect:Direct for UNIX 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 for UNIX 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 for UNIX cold restart, Connect:Direct for UNIX checks the value defined in the restart parameter to determine whether to perform the **run task** again.

    Run task restart works differently when Connect:Direct for UNIX runs behind a connection load balancer.

- 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. 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/cd4000/ndm/bin archive1

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

Sample Processes, Shell Scripts, and API Programs

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

The following list displays the file names of sample Processes and the type. Modify the Processes as required.

cpunx.cd

copy

rtunx.cd

run task

rjunx.cd

run job

sbunx.cd

submit

The following table displays the 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 a PDS
statarch.sh	archive statistics files
staterestore.sh	restore statistics files that have been archived
lcu.sh	launch the Local Connection Utility tool
spadmin.sh	launch the Secure+ Admin Tool
spcli.sh	launch the Secure+ CLI
spcust_sample1.sh	configure Secure+ for the SSL or TLS protocol

The following information displays the names of sample programs and a description:

- apicheck.c - Submits a Process to copy a file to a remote system. MAXDELAY is used in this example, which means that the program will not finish execution until the file has been transferred. A standard c compiler is used to compile this module.
- apicheck.C - Same as apicheck.c, except that it is compiled with one of the C++ compilers listed in the Connect:Direct for UNIX User Guide.
- exit_skeleton.c - This program is a skeleton of a user exit program that works in conjunction with Connect:Direct for UNIX. It demonstrates usage of all three user exits.
- exit_skeleton.C - Same as exit_skeleton_c, except that it is compiled with one of the C++ compilers listed in the Connect:Direct for UNIX User Guide.
- exit_sample.c - This is the same program as the skeleton user exit program, except that the security exit is demonstrated with code that approximates PassTicket functionality.

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 for UNIX server to operate correctly.

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 Connect:Direct for UNIX and allows, denies, or limits root access control. This file is created when you install Connect:Direct for UNIX. If the file is deleted or corrupted, access to Connect:Direct for UNIX is denied to all users.

Network map file

Describes the local node and other Connect:Direct for UNIX 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 verified clients are granted a connection.

Client configuration file

Identifies the port and host name used by a client to connect to Connect:Direct for UNIX.

Client authentication key file

Identifies Connect:Direct for UNIX servers that a Connect:Direct for UNIX client connects to. You can have multiple entries for multiple servers.

Installation Overview

The Installing section provides a quick reference to the installation process for Connect:Direct for UNIX followed by a detailed process. The installation and upgrade processes both include downloading installation items, completing prerequisites, preparing systems, running the installer, and completing the post-installation configurations.

Note: IBM Connect:Direct for UNIX can be deployed using an IBM Certified Container.

- The certified container offers a Red Hat certified IBM Connect:Direct for UNIX image and Helm chart and can be used to deploy a production-ready IBM Connect:Direct image into Red Hat OpenShift/ Kubernetes Service. For more information on see, "Installing IBM Connect:Direct for UNIX using an IBM Certified Container Software" on page 39.
- Additionally, this Red Hat certified container image can also be deployed in a standalone Docker environment. For more information see, Deploying IBM Connect:Direct for UNIX using a Docker container.

Install/Deploy Connect:Direct for UNIX

You can install and/or deploy Connect:Direct for UNIX using any of the following methods.

Table 1. Installing Connect:Direct for UNIX for the first time

Scenario	**Resource**
Install using the conventional method	"Installing Connect:Direct for UNIX" on page 18
Silent Installation method	Connect:Direct for UNIX Silent Installation
Deploy Connect:Direct for UNIX using a Docker container	Deploying IBM Connect:Direct for UNIX using a Docker container
Deploy Connect:Direct for UNIX using an IBM certified container	"Installing IBM Connect:Direct for UNIX using an IBM Certified Container Software" on page 39

Installing Connect:Direct for UNIX

Before you install Connect:Direct for UNIX, complete the worksheets to identify all information required to perform the installation.

Connect:Direct for UNIX may be installed on a local disk or a shared disk file system, also known as a clustered file system. Examples of clustered file systems are IBM's GPFS, Veritas Cluster File System, and Red Hat Global File System.

If Connect:Direct for UNIX is installed on a distributed file system, the distributed file system protocol must be NFSv4, and NFS root squash must be disabled. For example, a NAS device accessed via NFS v4.1 with root squash disabled is supported. (This restriction does not apply if Connect:Direct for UNIX is installed on a local file system and is accessing user files on an NFS file system. For more information about this situation, refer to NFS Restrictions while landing files with download directory set in the userfile.cfg

The only supported distributed file system protocol is NFSv4. For example, a NAS device accessed via NFS v4.1 is supported.

Note: When Connect:Direct for UNIX is installed on NFSv4, performance in high load scenarios may be reduced, significantly for NFSv4.0, as compared to the installation on a local or shared disk file system.

For example, in a development lab environment, a TCQ load test takes up to 3 times as long to run when Connect:Direct is installed on NFSv4.1 or NFSv4.2 than when it is installed on a local file system. When Connect:Direct is installed on NFSv4.0, the load test takes up to 30 times as long to run than when Connect:Direct is installed on a local file system- installing Connect:Direct on NFSv4.0 has limited applicability as a production solution.

Connect:Direct for UNIX requires that you install a server and at least one client location. You can install Connect:Direct for UNIX in different configurations:

- Install the server on a local system and the clients on remote systems
- Install the server and at least one client on a local system and the remaining clients on remote systems
- Install using a Silent Installation. See *IBM Connect:Direct for Unix silent installation* in the Mass Deployment documentation library.

Preparing to Install Connect:Direct for UNIX in a Cluster Environment

Connect:Direct for UNIX supports clustering software to allow two or more computers to appear to other systems as a single system. All computers in the cluster are accessible through a single IP address. Connect:Direct for UNIX can be installed in two types of clustering environments: high availability and load balancing clustering environments.

High-Availability Cluster Environments

Consider the following information when planning to use Connect:Direct for UNIX in a high-availability cluster environment.

Supported High-Availability Cluster Environments

Connect:Direct for UNIX is certified to operate in the following high-availability cluster environments:

- IBM high-availability cluster multiprocessing (HACMP) environment
- Hewlett-Packard MC/Service Guard
- SunCluster versions 2.2, 3.0, and 3.2
- Veritas Infoscale Availability (formerly Veritas Cluster Server)
- Red Hat High Availability Add-On

If you plan to install Connect:Direct for UNIX in a high-availability cluster environment, complete the following tasks:

- Install the clustering software on each computer in the cluster, including setting up a logical host or application package.
- Create a user with the same name and user ID on each cluster node.
- Create a Connect:Direct subdirectory on a shared file system on a shared volume group.
- Ensure that the shared file system is owned by the IBM Connect:Direct user.
- Install IBM Connect:Direct on the shared file system.
- Perform the procedures necessary to define the high-availability settings and configure the cluster environment.

Limitations of High-Availability Clusters

When running Connect:Direct for UNIX in a high-availability cluster environment, be aware of the following limitations:

- If a failure occurs, all Processes being held will be restarted when IBM Connect:Direct is restarted. This includes Processes that are held by the operator as well as Processes held in error. This could cause a security risk.
- When a IBM Connect:Direct ndmsmgr process associated with a IBM Connect:Direct Process is killed, the Process is not automatically restarted and is put in the Held in Error state. It must be manually restarted; otherwise, the IBM Connect:Direct Process is restarted when the cluster restart occurs.

Important: IBM offers standard support for Connect products that are installed according to the current documented installation instructions. Customers who encounter problems with a nonstandard installation, which cannot be reproduced in a standard environment, will be provided "best endeavor support". Offering best endeavor support, we will address questions and issues normally but if we determine the issue is due to a nonstandard installation environment you will be required to reinstall into a supported environment.

Load-Balancing Cluster Environments

In a load-balancing cluster environment, an incoming session is distributed to one of the Connect:Direct for UNIX instances based on criteria defined in the load balancer. Generally, from the point of view of the nodes behind the load balancer, only incoming or SNODE sessions are affected by the load balancer. PNODE, or outgoing sessions, operate the same way as non-cluster Connect:Direct for UNIX PNODE sessions.

SNODE Server Considerations for Load-Balancing Clusters

Consider the following when planning and setting up the Connect:Direct for UNIX SNODE servers in a load balancing cluster:

- The servers used for the Connect:Direct for UNIX instances behind the load balancer must all have access to common shared disk storage because of the following:
  - Any copy statement source and destination files for SNODE processes must reside in directories accessible to all servers.
  - All nodes must have access to a common SNODE work area and that area must be on a cluster file system or a Network File System version 4 (NFSv4) or greater resource. This includes the Amazon Elastic File System (EFS), as it is mounted via NFSv4 protocol. NFSv3 is not supported.
  - All servers must be of the same platform type (for example, all Solaris SPARC or all Linux Intel) and the same Connect:Direct for UNIX version and maintenance level.
- The system clocks on all servers must be synchronized in order for copy checkpoint/restart and run task synchronization to work.
- The administrator user ID used to install Connect:Direct for UNIX must be defined on each server and must be the same user and group number on each server.

SNODE Setup for Load-Balancing Clusters

Consider the following when planning and setting up the Connect:Direct for UNIX SNODEs in a load-balancing cluster:

- One Connect:Direct for UNIX node should be installed on each server behind the load balancer.
- Each node should be installed by the same user ID.
- Each node should have the same Connect:Direct for UNIX node name.
- Each node should have the same node-to-node connection listening port.
- A directory should be established for the shared SNODE work area used by the Connect:Direct for UNIX nodes behind the load balancer. This directory should be owned by the Connect:Direct for UNIX administrator ID and must be accessible to all of the servers behind the load balancer.
- Each node should specify the same path to the directory used for the shared SNODE work area. Specify this path in the **snode.work.path** parameter of the ndm.path record in the initialization parameter file.

Limitations of Load Balancing Clusters

When running Connect:Direct for UNIX in a cluster environment, be aware of the following limitations:

- If an incoming session fails and is restarted by the PNODE, then the restarted session may be assigned to any of the instances behind the load balancer and will not necessarily be established with the original SNODE instance.
- When shared SNODE work areas are configured and the **run task** is on the SNODE, then at restart time, Connect:Direct for UNIX cannot determine whether the original task is still active or not because the restart session may be with a different server. If you set the global run task restart parameters to yes in the initialization parameters file, a task could be restarted even though it may be active on another machine. Therefore, exercise caution when specifying restart=y.
- Each SNODE instance that receives a session for a given Process creates a TCQ entry for the Process. Each SNODE instance has its own TCQ file, and these files are not shared among SNODE instances. Only the work files created in the shared work area are shared among SNODE instances.
- When a Process is interrupted and restarted to a different SNODE instance, the statistics records for that Process is distributed between the two SNODE instances involved. As a result, you cannot select all the statistics records for a Process.

Conventions to Observe When Installing Connect:Direct for UNIX

Observe the following conventions when you install Connect:Direct for UNIX:

- Characters used in Netmap Node Names (or Secure+ Node Names or Secure+ Alias Names) should be restricted to A-Z, a-z, 0-9 and @ # $. _ - to ensure that the entries can be properly managed by Control Center, Sterling Connect:Direct Browser User Interface, or IBM Sterling Connect:Direct Application Interface for Java for Java (AIJ) programs.
- Acceptable responses to prompts are listed in brackets, where y specifies yes, n specifies no, and a specifies all.
- The default response is capitalized. Press Enter to accept the default value.
- Do not use colons (:) for values in the installation and customization scripts.
- Do not use keywords for values.
- Press Enter after each entry to continue.
- Terminate any procedure by pressing Ctrl-C.

Worksheet Instructions

Before you install IBM Connect:Direct for UNIX, complete the worksheets to help you gather the information needed to complete the installation.

Complete the following worksheets before you begin the installation.

- Installation Worksheet
- User Authorization Information File Worksheet
- CLI/API Configuration File Worksheet

The following worksheets are provided for your convenience:

- Network Map Remote Node Information File Worksheet
- Server Authentication key File Worksheet
- Client Authentication key File Worksheet

Installation Worksheet

Complete this worksheet to assist you during the installation procedure.

Parameter	Value
TCP/IP host name of the computer where the IBM Connect:Direct server is installed	
Directory or path on which the distribution media will be mounted	
Destination directory where IBM Connect:Direct will be installed, including the full path name	

Customization Worksheet

Use this worksheet during customization. Refer to the Customizing Connect:Direct for UNIX.

Parameter	Default Value	Value to Use
Connect:Direct node name you are customizing, up to 16 characters long. **Important:** Characters used in Netmap Node Names (or Secure+ Node Names or Secure+ Alias Names) should be restricted to A-Z, a-z, 0-9 and @ # $. _ - to ensure that the entries can be properly managed by Control Center, Sterling Connect:Direct Browser User Interface, or IBM Sterling Connect:Direct Application Interface for Java for Java (AIJ) programs.		
Initialization Parameters File Information		
TCP/IP port number that the server monitors for an API connection request.	1363	
TCP/IP port number that the server monitors for a remote Connect:Direct connection request: **Note:** Use the default port number, if available. If the default port number is being used by another service, use any other available port. Check the /etc/services file for a list of ports.	1364	
TCP/IP RPC port on Solaris SPARC and HPUX-IT	1367	
TCP/IP port number that the server uses to communicate with Install Agent included in Connect:Direct.	1365	

User Authorization Information File Worksheet

Use this worksheet when you are defining the user authorization information which includes the remote user information records and local user information records.

All IBM Connect:Direct users must have an entry in the user authorization information file.

Remote User Information Record

IBM Connect:Direct uses the remote user information record to establish a proxy relationship between remote and local user IDs. Remote user IDs are translated to valid local user IDs on the system where you are installing Connect:Direct for UNIX. IBM Connect:Direct also uses the remote and local user information records to determine the functionality of the user IDs that are translated and connected to it through a client using a IBM Connect:Direct API.

Use the following table to create a list of remote user IDs and the local user IDs to which they will be mapped. If necessary, make copies of this page to record additional remote user IDs and local user IDs.

For more information on creating remote user information records and for information on using special generic characters to map remote user IDs, refer to the IBM Connect:Direct for UNIX *Administration Guide*.

Remote User ID	at	Remote Node Name	mapped to	Local User ID
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	
	@		=	

Local User Information Record

Use the following table to record the local user ID records to create and the parameters to define. Define the additional parameters by editing the userfile.cfg file using any standard UNIX editor.

Default values are shown as capital letters in brackets. Before you begin defining local user information records, make copies of this worksheet for the number of users you plan to create.

Local User ID	Parameter	Description	Value to Assign
	admin.auth	Determines if the user has administrative authority. y—All the other command parameter capabilities in the local user information record are automatically assigned to this user. n—You must grant specific command parameters individually.	

Local User ID	Parameter	Description	Value to Assign
	client.cert_auth	Determines if the user can perform certificate authentication for client API connections. y—Enables client certificate authentication for the user n—Disables client certificate authentication for the user	y | n
	client.source_ip	Use this parameter to list all of the IP addresses and/or host names that are valid for this user's API connection. If you specify values for this field, the IP address of this user's API connection is validated with the client.source_ip list. If the IP address does not match the one specified on the list, the connection is rejected.	A comma-separated list of client IP addresses or host names associated with client IP addresses. The IP address of the client connection for this user must match the address configured in this field. For example: nnn.nnn.nnn.nnn, localhost
	cmd.chgproc	Specifies whether the user can issue the change Process command. y—Allows the user to issue the command. n—Prevents the user from issuing the command. a—Allows all users to issue this command.	
	cmd.delproc	Specifies whether the user can issue the delete Process command. y—Allows the user to issue the command. **n**—Prevents the user from issuing the command. **a**—Allows all users to issue this command.	y | n |a **y**—Default a—For all users

Local User ID	Parameter	Description	Value to Assign
	cmd.flsproc	Specifies whether the user can issue the flush Process command. y—Allows the user to issue the command. **n**—Prevents the user from issuing the command. **a**—Allows all users to issue this command.	y | n |a **y**—Default a—For all users
	cmd.selproc	Specifies whether the user can issue the select Process command. y—Allows the user to issue the command. n—Prevents the user from issuing the command. a—Allows all users to issue this command.	
	cmd.selstats	Specifies whether the user can issue the select statistics command. y—Allows the user to issue the command. n—Prevents the user from issuing the command. a—Allows all users to issue this command.	
	cmd.stopndm	Specifies whether the user can issue the stop command. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	
	cmd.submit	Specifies whether the user can issue the submit Process command. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	

Local User ID	Parameter	Description	Value to Assign
	cmd.trace	Specifies whether the user can issue the trace command. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	
	descrip	Permits the administrator to add descriptive notes to the record.	text string ________________
	name	Specifies the name of the user.	user name ________________
	phone	Specifies the telephone number of the user.	user phone ________________
	pstmt.copy	Specifies whether the user can issue the copy command. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	
	pstmt.copy. ulimit	Specifies the action to take 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. If a value is not defined in the initialization parameters file, the default is n. y or n or nnnnnnnK or nnnnM or nG where nnnnnnnK, nnnnM or nG establishes a default output file size limit for all copy operations. K—Thousands of bytes. M—Denotes millions of bytes. G—Denotes billions of bytes. The maximum value you can specify is 1 trillion byte.	
	pstmt.download	Specifies whether the user can download files. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	

Local User ID	Parameter	Description	Value to Assign
	pstmt.download _dir	Specifies the directory to which the user can download files.	
	pstmt.runjob	Specifies whether the user can issue the run job statement. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	
	pstmt.runtask	Specifies whether the user can issue the run task statement. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	
	pstmt.submit	Specifies whether the user can issue the submit statement. y—Allows the user to issue the command. **n**—Prevents the user from issuing the command.	
	snode.ovrd	Specifies whether the user can code the snodeid parameter on the submit command and Process and submit statements. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	
	pstmt.upload	Specifies whether the user can upload files. y—Allows the user to issue the command. n—Prevents the user from issuing the command.	
	pstmt.upload_dir	Specifies the directory from which the user can upload files.	
	run_dir	Specifies the directory that contains the programs and scripts the user can execute.	
	submit_dir	Specifies the directory from which the user can submit Processes.	

CLI/API Configuration File Worksheet

Use this worksheet to define the parameters needed to create a client configuration file. Create a separate file for each client attached to the server.

Parameter	Default Value	Value To Use
Port number of the Connect:Direct for UNIX server to which this client will connect. **Note:** Use the default port number if available. If the default port number is being used by another service, use any other available port. Check the /etc/services file for a list of ports.	1363	
Host name of the Connect:Direct for UNIX server to which this API will connect. You can also type the IP address of the server.		

Network Map Remote Node Information File Worksheet (TCP/IP)

The initial network map file containing a local node definition is created for you during the installation procedure; however, you must add a remote node record to the network map for each remote node you will communicate with unless you plan to specify the IP address or host name with the SNODE parameter when you submit a Process.

Use the information on this worksheet when you modify the network map. Make a copy of this worksheet for each remote node in the network.

Parameter	Default Value	Value To Use
Remote Connect:Direct node name		
Host name or IP address on which the remote IBM Connect:Direct server will run.		
Communication port number to call the remote Connect:Direct server:	**1364**	

Server Authentication Key File Worksheet

The initial server authentication key file is created during the installation procedure; however, you can update your key later. Use the information on this worksheet when you modify your key.

Parameter	Default Value	Value To Use
The host name on which the API is executed. An asterisk (*) stands for any host.	*	

IBM Connect:Direct security depends on a key (similar to a password) in a IBM 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 of the specific node or nodes, and should be kept secure. Be sure the authentication keys are available during installation, but do not record them on this worksheet or where they can be lost.

Client Authentication Key File Worksheet

The initial client authentication key file is created automatically during the installation; however, you can update your key at a later date. Use the information on this worksheet when you modify the key.

Parameter	Default Value	Value To Use
The host name on which a IBM Connect:Direct is executed. An asterisk (*) stands for any host.	*	

IBM Connect:Direct security depends on a key, similar to a password, in a IBM Connect:Direct server and an identical key in each API that will communicate with that server. The keys are defined and coordinated by the system administrator of the specific node or nodes, and should be kept secure.

Have the authentication keys you will use available during installation, but do not record them on this worksheet or anywhere else that could compromise security.

Installing IBM Connect:Direct

To install Connect:Direct for UNIX:

1. If you downloaded the software from IBM Passport Advantage, extract the installation files from the download folder.

   Note: For information on the how to download software using Passport Advantage see, https://www.ibm.com/software/passportadvantage/index.html.
2. Log on to the UNIX system with the privileges required to install software. You can create an account specifically for this purpose.

   Installing as root user is not recommended. Most files in the Connect:Direct installation directory does not need to be owned by the root, so an installation resulting in all Connect:Direct installation directory files owned by the root violates the Principle of Least Privilege. A malicious actor may be able to find a way to take advantage that wouldn't otherwise be available if only a limited number of files were owned by root.
3. Type the following command and press **Enter** to change to the directory that correspond to the UNIX platform:

   ```
   cd /<platform directory>
   ```

   Refer to the following information for the name of the platform directory for each platform.

   HP UX Itanium

   HP_Itanium

   IBM System pSeries

   IBM

   Sun SPARC systems

   Sun_Solaris

   Red Hat

   RedHat_linux

   SuSE

   SuSE_linux

   Linux for zSeries

   IBMS390_linux

4. Type the following command to start the installation and press **Enter**:

   ```
   cdinstall
   ```

   Installation on Linux platforms displays the following message: *awk: cmd. line:6: warning: escape sequence `\.' treated as plain `.'*

   This is a known issue with Install Anywhere and does not affect installation or functionality of Sterling Connect:Direct File Agent for UNIX on Linux.
5. Read the information displayed and press Enter.
6. Type the path name of the directory where Connect:Direct for UNIX will be installed and press **Enter**.
7. Press Enter to confirm the location
8. Do one of the following:
   - Press **Enter** to accept install the Server and Client on the same computer.
   - Type 2 to install the Server only and press Enter.
   - Type 3 to install the Client only and press Enter.

     The following screen is displayed:
9. Type the path and filename of the installation file and press Enter.

   If you are installing the Server and Client, a message is displayed to confirm that the server and client are being installed. If you selected option 2 or 3, the screen displays the software that will be installed.
10. Press Enter.

    If the destination directory does not have enough disk space, delete files to provide the necessary disk space. If disk space is available, the installation script copies files from the distribution media to the destination directory and verifies that the correct number of files and blocks are copied.

    The customization script starts automatically when the installation is complete.

Customizing Connect:Direct for UNIX

The customization script starts automatically after the installation is complete to set up the Connect:Direct for UNIX operating environment. It is located in d_dir/etc, where d_dir is the IBM Connect:Direct installation directory, and may be run by itself if needed for future configuration changes. The option you select determines what Connect:Direct for UNIX operating environment is configured: the Connect:Direct for UNIX Server only, the Connect:Direct for UNIX Client only, or the Connect:Direct for UNIX Server and Client.

After you customize the environment, you need to configure Connect:Direct for UNIX for using root privilege to create a Strong Access Control List (SACL) file and to set the owner and permissions of IBM Connect:Direct executables. You must create the SACL file and set the owner and permissions before you can run Connect:Direct for UNIX. See Configuring Connect:Direct for UNIX Using Root Privilege for more information about this process.

The customization script prompts you to begin the customization procedure:

1. Read the information and press Enter. The customization menu is displayed.
2. Do one of the following. Be sure to select the same configuration you selected during the installation.
   - Type 3 to customize the Server and Client and press Enter.

     If you are installing both the Client and the Server, complete the procedures in Setting Up the Connect:Direct for UNIX Server and Setting Up the Connect:Direct for UNIX Client.
   - Type 2 to customize the Client only and press Enter.

     If you are installing the Client only, complete the procedure Setting Up the Connect:Direct for UNIX Client.
   - Type 1 to customize the Server only and press Enter.

If you are installing the Connect:Direct for UNIX Server only, complete the procedure, Setting Up the Connect:Direct for UNIX Server.

Setting Up the IBM Connect:Direct Server

After you install Connect:Direct for UNIX, define the parameters needed by the Server for startup. If you installed the Server, the process to customize the Server starts automatically. To customize the server, enter the node to customize.

1. Type the name of the node, up to 16 characters, that you want to customize and press Enter.

   Important: Characters used in Netmap Node Names (or Secure+ Node Names or Secure+ Alias Names) should be restricted to A-Z, a-z, 0-9 and @ # $. _ - to ensure that the entries can be properly managed by Control Center, Connect:Direct Browser User Interface, or IBM Sterling Connect:Direct Application Interface for Java for Java (AIJ) programs. However, use of '@' characters in a node name is discouraged, as it will cause some clients to display proxy records incorrectly. If the '@' character is at the end of a node name, client display of a proxy record with this node name will fail.
2. Enter the TCP/IP port number that Control Center Director will use to communicate with Install Agent included with Connect:Direct. Use the default port, 1365, if available. If the default port number is being used by another service, use any other available port.
3. Type **y** to start Install Agent after installation or **n** to avoid starting Install Agent and press Enter. This value is entered into the initialization parameters file in the `agent.enable` parameter.
4. Enter the TCP/IP port number that Connect:Direct monitors for requests from remote nodes. Use the default port, 1364, if available. If the default port number is being used by another service, use any other available port. This value is entered into the initialization parameters file in the comm.info parameter.
5. Type the hostname or IP address that Connect:Direct Monitors for requests from remote nodes.

   If you use 0.0.0.0, Connect:Direct will listen for requests from remote nodes on all network adapters configured on the UNIX server.

   This value is entered into the initialization parameters file in the **comm.info** parameter.
6. Type the TCP/IP port number that Connect:Direct monitors for requests from Clients. If available, use the default value of **1363**. If the default port number is being used by another service, use any other available port.

   This value is entered into the network map file in the tcp.api parameter.
7. Type the hostname or IP address that Connect:Direct monitors for requests from Clients.

   This value is entered into the network map file in the **tcp.api** parameter.

   Connect:Direct creates the network map file and displays the directory path and file name.

   After you define the initialization parameters file, the customization script creates the network map file. A remote node record is added to the network map file. The remote node record is assigned the name of the local node you specified.
8. Type the TCP/IP port to listen for a PMGR RPC client request.

   This value is entered in the network map file in the **rpc.pmgr.port** parameter.

   Note: This prompt displays only on HPUXIT and SOLARIS-based deployments.
9. Press Enter. The netmap file is automatically created.

Customizing the User Authorization Information File

After the user authorization information file is created, you are ready to customize the file. Use this procedure to map remote user IDs to local user IDs and create remote user information records in the user authorization information file.

After the user authorization information file is created, the following message is displayed to prompt you to create an authorization information record for a remote user:

```
Insert remote user record? [Y/n]
```

1. Press Enter to add a remote user record.
2. Type the login or ID of the remote user and press Enter.
3. Type the name of the remote node and press Enter. The submitter ID and remote node name become the record name for the remote user information record.
4. Type the local user ID where the remote user ID will be mapped and press Enter. The local user ID is the UNIX account name. This value is associated with local.id in the remote user information record and defines the local user ID used to check security for the remote user.
5. Do one of the following:
   - To create another remote user record, press Enter and repeat steps 2-4.
   - Type n and press Enter if you do not want to create another remote user record.
6. Do one of the following:
   - If you do not want to create a local user record, type n and press Enter.
   - To create a local user record, press Enter.
7. Type the user ID for the local user and press Enter. This value is associated with userid in the user authorization information file.
8. Press Enter to grant administrative authority to the local user ID. All IBM Connect:Direct capabilities that you specify in the local user information record are assigned to the user.

   This value is assigned to admin.auth in the local user information record.
9. Do one of the following:
   - Press Enter and repeat this procedure to create another local user record.
   - Type n and press Enter to continue to the next task.

Creating an Authentication Key File

A server authentication key file verifies connection requests. Only authorized clients are granted a connection. IBM Connect:Direct generates the server authentication key file automatically. A message is displayed when the authenticaton key file is generated.

Press **Enter** to continue.

Setting Up the Connect:Direct for UNIX Client

After you install and customize Connect:Direct for UNIX Server, define the parameters needed by the Client for startup. To configure the client, configure the client configuration file and the client authentication key file to define all of the servers that this node connects to.

The Client configuration file is created during the customization process. A message is displayed after the Client configuration file is created.

To set up the client:

1. Type the port of the Server that the Client connects to and press Enter when ready.

   This value is associated with tcp.port in the Client configuration file.
2. Press Enter to accept the host name. This value is displayed in the tcp.hostname parameter in the Client configuration file.

   A message is displayed when the client authentication key file is created.
3. Press Enter .

Configuring Connect:Direct for UNIX Using Root Privilege

The Connect:Direct for UNIX Process Manager and Session Managers run as root to support Connect:Direct's user impersonation model. Before Connect:Direct accesses a source or destination file or executes a run task or a run job, it performs a programmatic logon to assume the identity of the appropriate user. This model enables a Connect:Direct user to use file system permissions to tightly control access to the user's data files. Supporting this model requires Connect:Direct to run as root.

You must create the SACL file and set the owner and permissions of the IBM Connect:Direct executables to run Connect:Direct for UNIX.

To configure the SACL file:

1. If you know the root password or if a system administrator is standing by who knows the root password, select option 4.
2. If you do not know the root password, but are authorized to gain root authority using sudo or a similar utility, type **5** to exit the Connect:Direct for UNIX customization script.

   A message is displayed to warn you that the SACL was not configured.
3. Read the information displayed and press **Enter**.

   A message is displayed to notify you of the creation of the test configuration.
4. To exit the customization, type **n** and press Enter.
5. If you did not select option 4 above, type cdcust (located in /<product install directory>/etc) using sudo to become root before creating the SACL and setting the owner and permissions of the executables.

Customizing the Owner and Permissions for the Executable Files

You must change the file attributes of the Session Manager (d_dir/ndm/bin/ndmsmgr), Process Manager (d_dir/ndm/bin/cdpmgr), Command Manager (d_dir/ndm/bin/ndmcmgr) User Manager (d_dir/ndm/bin/ndmumgr), Statistics Manager (d_dir/ndm/bin/cdstatm), Client Authenticator (d_dir/ndm/bin/ndmauthc), and Server Authenticator (d_dir/ndm/bin/ndmauths).

To customize the SACL file and set the owner and permissions of the IBM Connect:Direct executable files:

1. Type the full path of the IBM Connect:Direct destination directory and press **Enter**.
2. Press **Enter** to continue the customization. The following screen is displayed:
3. Type 4 to select Configurations requiring root privilege and press Enter.
4. Press Enter to configure the SACL file.
5. Press **Enter** to use root authority to create and check the SACL file.
6. If you have already assumed root authority by using a utility such as sudo, press **Enter**. Otherwise, type the root password and press **Enter**.

   If you type the root password incorrectly, a message informs you that the configuration tasks were not completed. Otherwise, a SACL file is created, the owner and permissions of the IBM Connect:Direct executable files are set, and the following messages and prompt are displayed.
7. Type y or n and press Enter. You are returned to the Customization menu.

   The following parameters are modified during the customization:

Parameter	**Value**	**File**
comm.info	Identifies the IP address and port that IBM Connect:Direct monitors for requests from remote nodes	Initialization parameters file
tcp.api	Identifies the IP address and port monitored by IBM Connect:Direct for requests from clients	Network map file

Parameter	Value	File
rnode.listen	Identifies the host used to monitor LU 6.2 connections	Initialization parameters file
admin.auth	Determines if user ID has administrative authority	User authorization information file
tcp.port	Specifies port number of the server that the client connects to	Client configuration file
tcp.hostname	Specifies host name of the server that the client connects to	Client configuration file

Installing Connect:Direct File Agent

After you install IBM Connect:Direct, install Connect:Direct File Agent at any time.

To install Connect:Direct File Agent:

1. Log on to the UNIX system with the privileges required to install software.

   Installing as root user is not recommended. Most files in the Connect:Direct installation directory does not need to be owned by the root, so an installation resulting in all Connect:Direct installation directory files owned by the root violates the Principle of Least Privilege. A malicious actor may be able to find a way to take advantage that wouldn’t otherwise be available if only a limited number of files were owned by root.
2. Type the following command and press **Enter** to change to the directory for your UNIX platform:

   ```
   cd /cdrom/<platform directory>
   ```

   Refer to the following list for the name of the platform directory for each platform:

   HP UX Itanium

      HP_Itanium

   IBM System pSeries

      IBM

   Sun SPARC systems

      Sun_Solaris

   Red Hat

      RedHat_linux

   SuSE

      SuSE_linux

   Linux zSeries

      IBMS390_linux
3. Type the following command to start the installation and press **Enter**:

   ```
   cdinstall
   ```
4. Read the information displayed and press **Enter**.
5. Type the path and press **Enter**. A warning that the directory exists is displayed:
6. Press **Enter** to continue. The following message is displayed:

```
Installed components detected in this directory.
A previous version of C:D for UNIX was detected.
Would you like this procedure to detect and upgrade your currently installed
options with minimal interaction?
If yes, the configuration files will be left in place and reused.
If not, the full installation procedure will prompt to either reuse, or purge
and rebuild, each configuration file.
Caution: If you are upgrading from earlier version of C:D for UNIX,
existing Processes in the tcq may encounter conversion error.
They will need to be deleted and resubmitted.
Type y or press Enter to continue with the upgrade procedure, or
type n to run the full installation procedure:[Y/n]
```

7. Type **n** and press **Enter**. The installation options menu is displayed:
8. Select 4 and press Enter.
9. Type the full Connect:Direct for UNIX installation path and filename and press **Enter**.

   Note: Steps 10 - 14 are applicable if Secure+ is not installed before starting File Agent installation. If Secure+ is already installed, go to Step 15.
10. If Secure+ is not installed, the following message is displayed. Press y or Enter to continue installation of Secure+.

    ```
    File Agent depends upon Secure+,
    which will be installed with File Agent.

    Connect:Direct for UNIX File Agent will be installed
    in your system. Do you want to continue?:[Y/n]
    ```
11. A message appears regarding the amount of disk space required to install Connect:Direct Secure Plus. If sufficient space is available, press Enter. If not, you are prompted to delete enough files to provide the enough space. The installation then exits. After you have cleared enough space, restart the installation.
12. Press Enter to confirm the installation location.
13. Press Enter if your node name is displayed. If your node name is not displayed, type your node name and press Enter.
14. Type a passphrase of at least 32 random characters and press Enter.
15. Press enter to confirm the installation. If sufficient space is available, the file agent installation begins. If not, you are prompted to delete files to provide the necessary disk space and the installation exits. After you have enough space, restart the installation.
16. After the installation completes, press Enter to return to the installation menu.

Upgrading to Connect:Direct Integrated File Agent

You can upgrade from an old release of Connect:Direct for UNIX node with Standalone file agent installed inside the Connect:Direct installation directory to Connect:Direct Integrated File Agent, through either interactive or silent installer.

Interactive Upgrade

While performing an interactive upgrade, you will get the following prompt:

```
Standalone File Agent detected in the Connect:Direct for UNIX installation directory.
Do you want to upgrade to the Integrated File Agent? If so, then please be aware
Secure+ and Connect:Direct Web services are required for Integrated File Agent.
Secure+ will be installed on your system if not already installed. Press Enter to
continue with the File Agent upgrade procedure:[Y/n]
```

Press Enter to upgrade from Standalone File Agent to Integrated File Agent.

Silent Upgrade

When you are following silent upgrade from an old release of Connect:Direct for UNIX, consider the following:

- If File Agent is not installed inside Connect:Direct installation directory, specifying the option cdai_installFA=”y” inside the options file or as a parameter to silent installer will install Integrated File Agent as a part of upgrade process.
- If Standalone File Agent is installed inside Connect:Direct installation directory, specifying the option cdai_installFA=”y” inside the options file or as a parameter to silent installer will convert Standalone File Agent to Integrated File Agent as a part of upgrade process.

Installing Connect:Direct Secure Plus

After you install Connect:Direct for UNIX, you can install Connect:Direct Secure Plus at any time.

To install Connect:Direct Secure Plus:

1. Log on to the UNIX system with the privileges required to install software. You can create an account specifically for this purpose.

   Installing as root user is not recommended. Most files in the Connect:Direct installation directory does not need to be owned by the root, so an installation resulting in all Connect:Direct installation directory files owned by the root violates the Principle of Least Privilege. A malicious actor may be able to find a way to take advantage that wouldn’t otherwise be available if only a limited number of files were owned by root.
2. From the distribution media, type the following command and press **Enter** to change to the directory that correspond to the UNIX platform:

   ```
   cd /cdrom/<platform directory>
   ```

   Refer to the following list for the name of the platform directory for each platform:

   HP UX Itanium

   HP_Itanium

   IBM System pSeries

   IBM

   Sun SPARC systems

   Sun_Solaris

   Red Hat

   RedHat_linux

   SuSE

   SuSE_linux

   Linux zSeries

   IBMS390_linux
3. Type the following command to start the installation and press **Enter**:

   ```
   cdinstall
   ```
4. Read the information displayed and press **Enter**.
5. Type the path and press **Enter**. A warning that the directory exists is displayed.
6. Press **Enter** to continue. The message that installed components are detected is displayed.
7. Type **n** and press **Enter** to run the full installation procedure. The following screen is displayed:

```
Connect:Direct for UNIX installation directory specified:
[directory path]
Please select one of the following installation options:
```

(1) Connect:Direct for UNIX Server and Client(CLI/API)

(2) Connect:Direct for UNIX Server

(3) Connect:Direct for UNIX Client(CLI/API)

(4) Connect:Direct for UNIX File Agent

(5) Connect:Direct for UNIX Secure+ Option for UNIX

(6) EXIT

```
Enter your choice:[1]
```

8. Type **5** and press **Enter**.
9. Type the full installation path and filename and press **Enter**.
10. Press **Enter** to confirm the installation.

    The program determines if space exists to complete the operation. If so, the Connect:Direct Secure Plus installation script and cpio files are extracted. If not, you are prompted to delete enough files. After you clear enough space, restart the installation procedure.
11. Read the information and press **Enter**.
12. Press **Enter** to confirm the installation location. A message is displayed regarding the amount of disk space required to install Connect:Direct Secure Plus. If sufficient space is available, press **Enter**. If not, you are prompted to delete enough files to provide the enough space. The installation then exits. After you have cleared enough space, restart the installation.A screen is displayed as the files are extracted and the JRE is configured. After the JRE is configured, the following prompt is displayed:
13. Press **Enter** if your node name is displayed. If your node name is not displayed, type your node name and press **Enter**.
14. Type a passphrase of at least 32 random characters and press **Enter**. The installation is complete.
15. Press **Enter** to return to the installation menu.

Connect:Direct Server and its Client connections

For client-server connections between a Connect:Direct Server and its clients be aware of the following limitations.

The connections between a Connect:Direct Server and the following clients are not secure:

- Connect:Direct Requester
- Windows CLI
- User-defined Windows SDK Client
- Connect:Direct for UNIX CLI
- User-defined UNIX ndampi Client
- Sterling File Agent

Though passwords sent by any of these clients to the Connect:Direct Server are obfuscated, the session is still not encrypted.

Defining High-Availability Settings in the Configuration Files

After you install Connect:Direct for UNIX on a shared file system, modify IBM Connect:Direct parameters to support a clustering environment. Install Connect:Direct for UNIX on a shared cluster file system to use it in a cluster environment. Complete the following procedure to modify the configuration files for a cluster environment:

1. Modify the following parameters:
   - In the initialization parameters file (initparm.cfg), set :comm.info=0.0.0.0;nnnn:\ where nnnn is the number of the listening port you defined during installation.
   - In the api.parms record of the NDMAPI configuration file (ndmapi.cfg), set :tcp.hostname=*logical_host_ip_name*:\ where *logical_host_ip_name* is the virtual address of the cluster.
   - In the network map file (netmap.cfg), set :tcp.api=logical_host_ip_name;nnnn:\ where nnnn is the API port you defined during installation.
2. In the network map file (netmap.cfg), set the outgoing address parameter in the local.node record to specify the local host IP name or address of the floating address to the following value. The remote node will also use this value for network map checking.

   ```
   :outgoing.address=(host name |nnnnnn.nnn):\
   ```
3. Modify the following records in the network map file:
   - Set :comm.info=logical_host_ip_name;1364:\ to configure the loopback remote node record.
   - Set tcp.max.time.to.wait to a value other than zero and less than the value set in the resource group manager of the cluster software to allow for clean shutdowns.
4. In the same volume group as the installation file system, create a user data file system that is shared by all cluster nodes.

Setting Up Additional Configuration Requirements in a SunCluster 3.X Environment

The High-Availability cluster commands shown below are not intended to be a complete set of instructions for setting up the High-Availability cluster software. Additional steps may be required to complete the configuration of the High-Availability environment. High-Availability cluster expertise is the responsibility of the customer. White papers detailing specific environments, setup steps, and testing of various High-Availability clusters are available on the Support On Demand web site. In addition to modifying the configuration files, complete the following procedure to set up a SunCluster 3.X cluster:

1. Type the following command to create the cluster resource:

   ```
   scdscreate -V SCI -T cd
   ```

   Use the V parameter to define the vendor ID and T parameter to define the resource ID.
2. Type the following command to configure the custom resource scripts:

   ```
   scdsconfig
   ```
3. Edit the SCI.cd resource file and change the value of RT_BASEDIR as follows:

   ```
   RT_BASEDIR=/opt/SCIcd/bin;

   RT_BASEDIR=/global/vol1/sci/cduserk1/3.5.00/suncluster+scripts/SCIcd/bin;
   ```

Configuring Additional Requirements in a SunCluster 2.2 Environment

In addition to modifying the configuration files, complete the following steps to complete the SunCluster 2.2 setup:

1. Place the following sample scripts and configuration files in a directory that is available to SunCluster 2.2 software:
   - cd_start.sh
   - cd_stop.sh

2. Update the scripts as required for your environment.
3. Copy the sample scripts to all nodes in the cluster.
4. Issue the **hareg** command to register the IBM Connect:Direct data service. Refer to the SunCluster documentation for more information. Following is a sample command:

```
hareg -r cd \
```

Setting Up Additional Configuration Requirements for IBM HACMP

In addition to modifying the configuration files, complete the following steps to complete the IBM high-availability cluster multiprocessing (HACMP) setup:

1. Place the following sample scripts and configuration files in a directory that is available to the IBM HACMP software:
   - cd_start_net.sh
   - cd_stop_net.sh
2. Update the scripts as required for your environment.
3. Copy the sample scripts to all nodes in the cluster.

Setting Up Additional Configuration Requirements for Hewlett-Packard MC/ServiceGuard

The HP Solutions Competency Center (SCC) has successfully integrated IBM Connect:Direct with MC/Service Guard. The implementation and certification of IBM Connect:Direct followed the SCC's high availability Implementation and Certification Process. Refer to the Implementation and Certification With Hewlett-Packard's MC/ServiceGuard High Availability Software document located on the Support on Demand Web site.

Verifying the Installation

1. Type the following command to identify the release and platform operating system release, where d_dir is the destination directory and binaryx is a file in the bin/ directory (for example, cdpmgr):

```
% d_dir/etc/cdver d_dir/ndm/bin/[binary1 binary2 ...]
```

2. Log in with the user account under which IBM Connect:Direct was installed.
3. Type the following command to start Connect:Direct for UNIX Server, where d_dir is the destination directory and cd_node is the IBM Connect:Direct node name.

```
$ d_dir/ndm/bin/cdpmgr -i d_dir/ndm/cfg/cd_node/initparm.cfg
```

4. Do one of the following to set the environment variable NDMAPICFG to point to the client configuration file:
   - If you are using the Bourne or Korn shell, type the following command:

   ```
   $ NDMAPICFG=d_dir/ndm/cfg/cliapi/ndmapi.cfg
   $ export NDMAPICFG
   ```

   - If you are using the C shell, type the following command:

   ```
   % setenv NDMAPICFG d_dir/ndm/cfg/cliapi/ndmapi.cfg
   ```

5. Type the following command to invoke the IBM Connect:Direct client:

```
$ d_dir/ndm/bin/direct
```

6. Type the following command:

```
Direct> select statistics;
```

Read the statistics information and ensure that the initialization started with no errors. If any errors are displayed, resolve the errors before continuing.

7. Type the following command to submit a sample Process:

   ```
   Direct> submit file=d_dir/ndm/bin/sample.cd;
   ```

   This sample Process copies a binary file named msgfile.cfg to the file cddelete.me in your HOME directory (your node is both the PNODE and the SNODE). The checkpointing interval is set to 2M and extended compression is used.
8. Type the following select process command to monitor data transmission activity:

   ```
   Direct> select process pnumber=1;
   ```

   IBM Connect:Direct generates a report with the Process name and number, user, submitter node, queue, and status.
9. After the Process is complete, type the following select statistics command to review the statistics log for the Process:

   ```
   Direct> select statistics pnumber=1;
   ```

10. Do one of the following:

    - Type the following command to manually shut down the IBM Connect:Direct server:

      ```
      Direct> stop;
      ```

    - Type the following command to quit the IBM Connect:Direct client without shutting down the server: Direct> quit;

      The client terminates automatically when you stop the server.

Uninstalling Connect:Direct for UNIX

Follow the below steps to unistall Connect:Direct for UNIX:

1. Login with root account.
2. Run the following command to uninstall Connect:Direct for UNIX, where **d_dir** is the destination directory `d_dir/ndm/bin/uninstall`
3. Type **y** to continue uninstall or **n** to exit and press **Enter.**
4. Type **y** and press **Enter** to stop CDU services, remove all configurations and uninstall Connect:Direct for UNIX or type **n** and press **Enter** to exit.

Installing IBM Connect:Direct for UNIX using an IBM Certified Container Software

IBM Certified Container Software (CCS) can be installed on the Kubernetes based cluster.

Note: The information in "Installing IBM Connect: Direct® for UNIX using an IBM Certified Container Software" section was deprecated and is no longer maintained. For the most up-to-date information, refer to the latest release of the Installing IBM Connect: Direct® for UNIX using an IBM Certified Container Software section.

Kubernetes is an open-source container orchestration engine to automate the deployment, scaling and management of containerized applications. This application release has been qualified and certified on an on-premise Red Hat® OpenShift® Container Platform (OCP) which is an enterprise-ready Kubernetes container platform with full-stack automated operations to manage the deployment life-cycle.

IBM CCS offers a container image and a helm chart. It meets the standard criteria for the packaging and deployment of containerized software. In addition, the container image is Red Hat certified.

For more details, refer to the below links:

- Kubernetes
- Red Hat OpenShift Container Platform (OCP)
- Helm chart

Planning

Below are the step by step guide to plan and install the IBM Certified Container Software (CCS) in your cluster. Go through each links one by one and perform the operations as applicable for your deployment needs.

Before you deploy the application, you must use the following information to plan the deployment:

- Verifying system requirements
- Application license requirements
- IBM Licensing and Metering service
- Certificates files for Secure Plus
- PSP and SCC requirements
- Hardening RedHat OpenShift cluster
- Encrypting etcd data

User Roles

Deployment tasks can be performed by cluster administrator and project administrator. The following table illustrates the type of task that are typically associated with each administrative role. The list is not intended to be exhaustive -

Role	Task
Cluster Administrator	• Creating namespaces (projects) • Creating PSP or SCC and assigning them to namespace (project) • Configuring Storage for data persistence • Providing environmental details • Installing and configuring IBM Licensing and Metering Service
Project Administrator	• Creating Secrets • Installing IBM Certified Container Software for Connect Direct UNIX • Validating the install • Post deployment tasks

Verifying System Requirements

Before you begin the deployment process, verify that your system meets the hardware and software requirements specified for this release.

The Certified Container Software for IBM Connect:Direct for UNIX has been verified on Red Hat Linux and requires the following minimum hardware and software resources:

Hardware Requirements

- 1 GB Disk space
- 500m CPU core
- 2 GB RAM
- 100 MB for Persistent Volume
- 3-5 GB of ephemeral storage

Note:

- 100 MB is minimum requirement for fresh deployment.
- For upgrade, make sure you have sufficient space on persistent volume so that backup of application data could reside on persistent volume.
- For Production, double all of the above requirements and make Persistent Volume size at least 1 GB.

Software Requirements

For Kubernetes cluster:

- Kubernetes >=v1.22 and <=1.26
- helm (client) v2.17 and >=3.0 and <=v3.11
- compatible kubectl client tool

For Red Hat OpenShift cluster:

- Red Hat OpenShift Container Platform (OCP) >=v4.9 and <=4.12
- helm (client) v2.17 and >=3.0 and <=v3.11
- compatible oc client tool

Other common requirements:

- docker/podman to manage container images
- If NFS is used as backed-storage for Persistent Volume, ensure version >=4.1

Note: For helm 3.2.1 and greater, no Tiller (server) is required.

A certified container deployment strictly enforces the above system requirements. If any of the requirements are not met, the deployment may fail. If the deployment fails, then review the deployment log for a list of non-compliant items.

Installing OpenShift Container platform

Note: This step is optional. It is only needed if IBM Certified Container Software for Connect:Direct for UNIX deployment is planned for deployment on Redhat OpenShift cluster.

OpenShift container platform brings together Docker and Kubernetes and provides an API to manage these services. OpenShift Container Platform allows you to create and manage containers.

It is an on-premise platform service that uses Kubernetes to manage containers built on a foundation of Red Hat Enterprise Linux. For more information on how to setup an OpenShift container platform cluster environment, see Installing OpenShift.

Application license requirements

You must read the IBM Connect:Direct for Unix License agreement terms before deploying the software. The license number is 'L-MTAE-C5RR2U'.

To accept the license, set `license` variable to `true` at Helm CLI installation command. If `license` variable is set to `false` then deployment of IBM Certified Container Software for Connect:Direct for UNIX would not be successful. For more information see, Configuring - Understanding values.yaml.

The IBM Certified Container Software for Connect:Direct for UNIX is deployed as non-production by default. You can override this default behavior by changing the *licenseType* variable to prod. The *licenseType* value would be used to annotate the IBM Certified Container Software for Connect:Direct for UNIX, which would be eventually used by Licensing and Metering service tool.

IBM Licensing and Metering service

The IBM Certified Container Software for Connect Direct UNIX has been integrated with IBM Licensing and Metering service using Operator. This service collects information about license usage of IBM Certified Container Software for Connect Direct UNIX.

You can use the `ibm-licensing-operator` to install the IBM Licensing and Metering service on any Kubernetes based cluster. License Service collects information about license usage of IBM Containerized Products. You can retrieve license usage data through a dedicated API call and generate an audit snapshot on demand cluster without IBM Cloud Pak.

For the installation overview see, License Service deployment.

For retrieving the licensing information see, Track license usage .

Certificates files for Secure Plus

The IBM Certified Container Software for Connect:Direct UNIX application provided enhanced security for IBM Connect:Direct and is available as a separate component. It uses cryptography to secure data during transmission. By default security protocols are TLS 1.2 and TLS.13.

For configuring Secure Plus while installing IBM Certified Container Software for Connect:Direct UNIX you need certificate file. You must ensure valid certificates files are used during the deployment. Keep the certificates files handy during the IBM Certified Container Software for Connect Direct UNIX deployment. For more information on Secure plus and certificate file, refer Introduction to Connect:Direct Secure Plus for UNIX and Certificate Files.

Pod Security Standard, Pod Security Policy and Security Context Constraints requirements

With Kubernetes v1.25, the Pod Security Admission (PSA) controller has been introduced replacing the older Pod Security Policy (PSP). Kubernetes has defined Pod Security Standards (PSS) which can be applied at the namespace level. This helm chart is compatible with baseline standards with enforce security level. For more information on PSS, refer - Pod Security Standards (https://kubernetes.io/docs/concepts/security/pod-security-standards/)

So, if deployment is planned on Kubernetes v1.25 and above, then consider PSS. Else, Pod Security Policy should be the way to go.

If planning to upgrade from older Kubernetes to v1.25 and above, refer to Kubernetes Migrate from PSP documentation (https://kubernetes.io/docs/tasks/configure-pod-container/migrate-from-psp/) to understand the migrating from PSP to the built-in PSA controller.

Both Pod Security Policy (PSP) and Security Context Constraints (SCC) are cluster-level resource and allows the administrator to control the security aspects of pods in Kubernetes and Red Hat OpenShift clusters respectively.

Depending on the cluster environment, IBM Certified Container Software for Connect Direct UNIX requires PSP or SCC to be tied to the target namespace prior to deployment. Since, it is a cluster level resource, discuss this with your Cluster Administrator as they are needed to create these resources.

For more information on PSP see, Creating Pod Security Policy for Kubernetes cluster and Security Context Constraints for OpenShift cluster.

Hardening RedHat OpenShift Cluster

This is not a mandatory requirement for IBM CCS installation but a security aspect to be understood to make your cluster more secure.

If you are planning to deploy on OpenShift cluster then, there are certain guidelines by OpenShift. See here for more details, Hardening RedHat OpenShift cluster.

Encrypting etcd data

This is not a mandatory requirement for IBM CCS installation but a security aspect to be understood to make your cluster more secure.

By default, etcd data is not encrypted in Kubernetes/OpenShift cluster. You can enable etcd encryption for your cluster to provide an additional layer of data security (data at rest). For example, it can help protect the loss of sensitive data if an etcd backup is exposed to the incorrect parties. For more information, refer Kubernetes and Red Hat OpenShift.

Installing

After you review the system requirements and other planning information, you can proceed to install IBM Certified Container Software for Connect Direct UNIX. The following tasks represent the typical task flow for performing your installation:

The following tasks represent the typical task flow for performing your IBM Connect:Direct for UNIX using an IBM Certified Container Software installation:

Brief about IBM Certified Container Software

- A Helm chart is organized as a collection of files inside a directory by the name of the Chart itself. For more information see, Helm Charts.

  Example Helm Chart

  ```
  <Name of a Chart/>
    Chart.yaml          # A YAML file containing information about the Chart.
    LICENSE             # OPTIONAL: A plain text file containing the license for the Chart.
    README.md           # OPTIONAL: A README file.
    requirements.yaml   # OPTIONAL: A YAML file listing dependencies for the Chart.
    values.yaml         # The default configuration values for this Chart.
    Charts/             # A directory containing any Charts upon which this Chart depends.
    templates/          # A directory of templates that, when combined with values, generates
  valid Kubernetes manifest files.
    templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes.
  ```

  – This Helm chart deploys IBM Connect:Direct for Unix on a container management platform with the following resource deployments:

    - statefulset pod <release-name>-IBM-connect-direct-0

      1 replica by default

    - configMap <release-name>-ibm-connect-direct

      This is used to provide default configuration in cd_param_file.

    - service <release-name>-ibm-connect-direct

      This is used to expose the IBM Connect:Direct services for accessing using clients.

    - service-account <release-name>-ibm-connect-direct-serviceaccount

      This service will not be created if serviceAccount.create is false.

    - persistence volume claim <release-name>-ibm-connect-direct.

- Certified Container Software commands

For more information on other commands and options, see Helm Commands.

1. To install a Chart

   ```
   $ helm install
   ```

2. To upgrade to a new release

   ```
   $ helm upgrade
   ```

3. To rollback a release to a previous version

   ```
   $ helm rollback
   ```

4. To delete the release from Kubernetes.

   ```
   $ helm delete
   ```

Setting up your registry server

To install IBM Certified Container Software for Connect:Direct for UNIX, you must have a registry server where you can host the image required for installation.

Using the existing registry server

If you have an existing registry server, you can use it, provided that it is in close proximity to cluster where you will deploy IBM Certified Container Software for Connect:Direct for UNIX. If your registry server is not in close proximity to your cluster, you might notice performance issues. Also, before the installation, ensure that pull secrets are created in the namespace/project and are linked to the service accounts. You will need to properly manage these pull secrets. This pull secret can be updated in values.yaml file `image.imageSecrets`.

Using Docker registry

Kubernetes does not provide a registry solution out of the based. However, you can create your own registry server and host your images. Please refer to the deployment of registry server.

Setting up Namespace or project

To install IBM Certified Container Software for Connect:Direct for UNIX, you must have an existing namespace/project or create a new if required.

You can either use an existing namespace or create a new one in Kubernetes cluster. Similarly, you either use an existing project or create a new one in OpenShift cluster. A namespace or project is a cluster resource. So, it can only be created by a Cluster Administrator. Refer the following links for more details -

For Kubernetes - Namespaces

For Red Hat OpenShift - Working with projects

The IBM Certified Container Software for Connect:Direct for UNIX has been integrated with IBM Licensing and Metering service using Operator. You need to install this service. For more information, refer to License Service deployment without an IBM Cloud Pak.

Installing and configuring IBM Licensing and Metering service

License Service is required for monitoring and measuring license usage of IBM Certified Container Software for Connect:Direct for UNIX in accordance with the pricing rule for containerized environments. Manual license measurements are not allowed. Deploy License Service on all clusters where IBM Certified Container Software for Connect:Direct for UNIX is installed.

The IBM Certified Container Software for Connect:Direct for UNIX contains an integrated service for measuring the license usage at the cluster level for license evidence purposes.

Overview

The integrated licensing solution collects and stores the license usage information which can be used for audit purposes and for tracking license consumption in cloud environments. The solution works in the background and does not require any configuration. Only one instance of the License Service is deployed per cluster regardless of the number of containerized products that you have installed on the cluster.

Deploying License Service

Deploy License Service on each cluster where IBM FHIR Server is installed. License Service can be deployed on any Kubernetes based orchestration cluster. For more information about License Service, how to install and use it, see the License Service documentation.

Validating if License Service is deployed on the cluster

To ensure license reporting continuity for license compliance purposes make sure that License Service is successfully deployed. It is recommended to periodically verify whether it is active. To validate whether License Service is deployed and running on the cluster, you can, for example, log into the Kubernetes or Redhat OpenShift cluster and run the following command:

For Kubernetes

```
kubectl get pods --all-namespaces | grep ibm-licensing | grep -v operator
```

For Redhat OpenShift

```
oc get pods --all-namespaces | grep ibm-licensing | grep -v operator
```

The following response is a confirmation of successful deployment:

```
1/1 Running
```

Archiving license usage data

Remember to archive the license usage evidence before you decommission the cluster where IBM Certified Container Software for Connect:Direct for UNIX Server was deployed. Retrieve the audit snapshot for the period when IBM Certified Container Software for Connect:Direct for UNIX was on the cluster and store it in case of audit.

For more information about the licensing solution, see License Service documentation.

Downloading the Certified Container Software

Before you install IBM Certified Container Software for Connect:Direct for UNIX, ensure that the installation files are available on your client system.

Depending on the availability of internet on the cluster, the following procedures can be followed. Choose the one which applies best for your environment.

Online Cluster

The cluster which has access to the internet is called Online cluster. You may have a Kubernetes or OpenShift cluster and it has access to the internet. The process to get required installation files consists of two steps:

1. **Create the entitled registry secret**: Complete the following steps to create a secret with the entitled registry key value:

a. Ensure that you have obtained the entitlement key that is assigned to your ID.

   i) Log in to My IBM Container Software Library by using the IBM ID and password that are associated with the entitled software.

   ii) In the Entitlement keys section, select Copy key to copy the entitlement key to the clipboard.

   iii) Save the entitlement key to a safe location for later use.

   To confirm that your entitlement key is valid, click View library that is provided in the left of the page. You can view the list of products that you are entitled to. If IBM Connect:Direct for Unix is not listed, or if the View library link is disabled, it indicates that the identity with which you are logged in to the container library does not have an entitlement for IBM Connect:Direct for Unix. In this case, the entitlement key is not valid for installing the software.

   Note: For assistance with the Container Software Library (e.g. product not available in the library; problem accessing your entitlement registry key), contact MyIBM Order Support.

b. Set the entitled registry information by completing the following steps:

   i) Log on to machine from where the cluster is accessible

   ii) export ENTITLED_REGISTRY=cp.icr.io

   iii) export ENTITLED_REGISTRY_USER=cp

   iv) export ENTITLED_REGISTRY_KEY=<entitlement_key>

c. This step is optional. Log on to the entitled registry with the following docker login command:

```
docker login "$ENTITLED_REGISTRY" -u "$ENTITLED_REGISTRY_USER" -p "$ENTITLED_REGISTRY_KEY"
```

d. Create a Docker-registry secret:

```
 kubectl create secret docker-registry <any_name_for_the_secret> --docker-
username=$ENTITLED_REGISTRY_USER --docker-password=$ENTITLED_REGISTRY_KEY --docker-
server=$ENTITLED_REGISTRY -n <your namespace/project name>
```

e. Update the service account or helm chart image pull secret configurations using `image.imageSecrets` parameter with the above secret name.

2. **Download the Helm chart**: You can follow the steps below to download the helm chart from the repository.

   a. Make sure that the helm client (CLI) is present on your machine. Execute/run helm CLI on machine and you should be able to see the usage of helm CLI.

   ```
   helm
   ```

   b. Check the `ibm-helm` repository in your helm CLI.

   ```
   helm repo list
   ```

   If the `ibm-helm` repository already exists with URL `https://raw.githubusercontent.com/IBM/charts/master/repo/ibm-helm`, then update the local repository else add the repository.

   c. Update the local repository, if `ibm-helm` repository already exists on helm CLI.

   ```
   helm repo update
   ```

   d. Add the helm chart repository to local helm CLI if it does not exist.

   ```
   helm repo add ibm-helm https://raw.githubusercontent.com/IBM/charts/master/repo/ibm-helm
   ```

   e. List ibm-connect-direct helm charts available on repository.

   ```
   helm search repo -l ibm-connect-direct
   ```

   f. Download the latest helm chart.

```
helm pull ibm-helm/ibm-connect-direct
```

At this point we have a locally present helm chart and an Entitled registry secret. Make sure you configure the helm chart to use the Entitled registry secret to download the required container image for deploying the IBM Connect:Direct for UNIX chart.

Offline (Airgap) Cluster

You have a Kubernetes or OpenShift cluster but it is a private cluster which means it does not have the internet access. Depending upon the cluster, follow the below procedures to get the installation files.

For Kubernetes Cluster

Since, your Kubernetes cluster is private and it does not have internet access, you cannot download the required installation files directly from the server. By following steps below, you can get the required files.

1. Get an RHEL machine which has
   - internet access
   - Docker CLI (docker) or Podman CLI (podman)
   - kubectl
   - helm
2. Download the Helm chart by following the steps mentioned in the Online installation section above.
3. Extract the downloaded helm chart.

   ```
   tar -zxf <ibm-connect-direct-helm chart-name>
   ```

4. Get the container image detail:

   ```
   erRepo=$(grep -w "repository:" ibm-connect-direct/values.yaml |cut -d '"' -f 2)
   ```

   ```
   erTag=$(grep -w "tag:" ibm-connect-direct/values.yaml | cut -d '"' -f 2)
   ```

   ```
   erImgTag=$erRepo:$erTag
   ```

5. This step is optional if you already have a docker registry running on this machine. Create a docker registry on this machine. Follow "Setting up your registry server" on page 44.
6. Get the Entitled registry entitlement key by following steps a and b explained in "Online Cluster" on page 45 under Create the entitled registry section above.
7. Get the container image downloaded in docker registry:

   ```
   docker login "$ENTITLED_REGISTRY" -u "$ENTITLED_REGISTRY_USER" -p
   "$ENTITLED_REGISTRY_KEY"
   ```

   ```
   docker pull $erImgTag
   ```

   Note: Skip step 8, 9 and 10, if the cluster where deployment will be performed is accessible from this machine and cluster can fetch container images from registry running on this machine.
8. Save the container image.

   ```
   docker save -o <container image file name.tar> $erImgTag
   ```

9. Copy/Transfer the installation files to your cluster. At this point you have both downloaded container image and helm chart for IBM Connect:Direct for UNIX. You need to transfer these two file to a machine from where you can access your cluster and its registry.
10. After transferring the files, load the container image into your registry.

    ```
    docker load -i <container image file name.tar>
    ```

For Red Hat OpenShift Cluster

If your cluster is not connected to the internet, the deployment can be done in your cluster via connected or disconnected mirroring.

If you have a host that can access both the internet and your mirror registry, but not your cluster nodes, you can directly mirror the content from that machine. This process is referred to as **connected mirroring**. If you have no such host, you must mirror the images to a file system and then bring that host or removable media into your restricted environment. This process is referred to as **disconnected mirroring**.

Before you begin

You must complete the steps in the following sections before you begin generating mirror manifests:

Important: If you intend on installing using a private container registry, your cluster must support ImageContentSourcePolicy (ICSP).

Prerequisites

Regardless of whether you plan to mirror the images with a bastion host or to the file system, you must satisfy the following prerequisites:

- Red Hat® OpenShift® Container Platform requires you to have cluster admin access to run the deployment.
- A Red Hat® OpenShift® Container Platform cluster must be installed.

Prepare a host

If you are in an air-gapped environment, you must be able to connect a host to the internet and mirror registry for connected mirroring or mirror images to file system which can be brought to a restricted environment for disconnected mirroring. For information on the latest supported operating systems, see ibm-pak plugin install documentation.

The following table explains the software requirements for mirroring the IBM Cloud Pak images:

Table 2. software requirements for mirroring the IBM Cloud Pak images

Software	**Purpose**
Docker	Container management
Podman	Container management
Red Hat OpenShift CLI (oc)	Red Hat OpenShift Container Platform administration

Complete the following steps on your host:

1. Install Docker or Podman.

   To install Docker (for example, on Red Hat® Enterprise Linux®), run the following commands:

   Note: If you are installing as a non-root user you must use sudo. For more information, refer to the Podman or Docker documentation for installing as a non-root user.

   ```
   yum check-update
   yum install docker
   ```

   To install Podman, see Podman Installation Instructions.
2. Install the `oc` Red Hat® OpenShift® Container Platform CLI tool.
3. Download and install the most recent version of IBM Catalog Management Plug-in for IBM Cloud Paks from the IBM/ibm-pak. Extract the binary file by entering the following command:

   ```
   tar -xf oc-ibm_pak-linux-amd64.tar.gz
   ```

Run the following command to move the file to the `/usr/local/bin` directory:

Note: If you are installing as a non-root user you must use sudo. For more information, refer to the Podman or Docker documentation for installing as a non-root user.

```
mv oc-ibm_pak-linux-amd64 /usr/local/bin/oc-ibm_pak
```

Note: Download the plug-in based on the host operating system. You can confirm that `oc ibm-pak -h` is installed by running the following command:

```
oc ibm-pak --help
```

The plug-in usage is displayed.

For more information on plug-in commands, see command-help.

Your host is now configured and you are ready to mirror your images.

Creating registry namespaces

Top-level namespaces are the namespaces which appear at the root path of your private registry. For example, if your registry is hosted at `myregistry.com:5000`, then `mynamespace` in `myregistry.com:5000/mynamespace` is defined as a top-level namespace. There can be many top-level namespaces.

When the images are mirrored to your private registry, it is required that the top-level namespace where images are getting mirrored already exists or can be automatically created during the image push. If your registry does not allow automatic creation of top-level namespaces, you must create them manually.

When you generate mirror manifests, you can specify the top-level namespace where you want to mirror the images by setting TARGET_REGISTRY to `myregistry.com:5000/mynamespace` which has the benefit of needing to create only one namespace mynamespace in your registry if it does not allow automatic creation of namespaces. The top-level namespaces can also be provided in the final registry by using `--final-registry`.

If you do not specify your own top-level namespace, the mirroring process will use the ones which are specified by the CASEs. For example, it will try to mirror the images at `myregistry.com:5000/cp` etc.

So if your registry does not allow automatic creation of top-level namespaces and you are not going to use your own during generation of mirror manifests, then you must create the following namespace at the root of your registry.

- cp

There can be more top-level namespaces that you might need to create. See section on Generate mirror manifests for information on how to use the `oc ibm-pak describe` command to list all the top-level namespaces.

Set environment variables and download CASE files

If your host must connect to the internet via a proxy, you must set environment variables on the machine that accesses the internet via the proxy server.

If you are mirroring via connected mirroring, set the following environment variables on the machine that accesses the internet via the proxy server:

```
export https_proxy=http://proxy-server-hostname:port
export http_proxy=http://proxy-server-hostname:port

# Example:
export https_proxy=http://server.proxy.xyz.com:5018
export http_proxy=http://server.proxy.xyz.com:5018
```

Before mirroring your images, you can set the environment variables on your mirroring device, and connect to the internet so that you can download the corresponding CASE files. To finish preparing your host, complete the following steps:

Note: Save a copy of your environment variable values to a text editor. You can use that file as a reference to cut and paste from when you finish mirroring images to your registry.

1. Create the following environment variables with the installer image name and the version.

   ```
   export CASE_NAME=ibm-connect-direct
   ```

   To find the CASE name and version, see IBM: Product CASE to Application Version.
2. Connect your host to the intranet.
3. The plug-in can detect the locale of your environment and provide textual helps and messages accordingly. You can optionally set the locale by running the following command:

   ```
   oc ibm-pak config locale -l LOCALE
   ```

   where LOCALE can be one of `de_DE, en_US, es_ES, fr_FR, it_IT, ja_JP, ko_KR, pt_BR, zh_Hans, zh_Hant`.
4. Configure the plug-in to download CASEs as OCI artifacts from IBM Cloud Container Registry (ICCR).

   ```
   oc ibm-pak config repo 'IBM Cloud-Pak OCI registry' -r oci:cp.icr.io/cpopen --enable
   ```
5. Enable color output (optional with v1.4.0 and later)

   ```
   oc ibm-pak config color --enable true
   ```
6. Download the image inventory for your IBM Cloud Pak to your host.

   Tip: If you do not specify the CASE version, it will download the latest CASE.

   ```
   oc ibm-pak get \
   $CASE_NAME \
   --version $CASE_VERSION
   ```

By default, the root directory used by plug-in is `~/.ibm-pak`. This means that the preceding command will download the CASE under `~/.ibm-pak/data/cases/$CASE_NAME/$CASE_VERSION`. You can configure this root directory by setting the `IBMPAK_HOME` environment variable. Assuming `IBMPAK_HOME` is set, the preceding command will download the CASE under `$IBMPAK_HOME/.ibm-pak/data/cases/$CASE_NAME/$CASE_VERSION`.

The logs files will be available at `$IBMPAK_HOME/.ibm-pak/logs/oc-ibm_pak.log`.

Your host is now configured and you are ready to mirror your images.

Mirroring images to your private container registry

The process of mirroring images takes the image from the internet to your host, then effectively copies that image to your private container registry. After you mirror your images, you can configure your cluster and complete air-gapped installation.

Complete the following steps to mirror your images from your host to your private container registry:

1. Generate mirror manifests
2. Authenticating the registry
3. Mirror images to final location
4. Configure the cluster
5. Install IBM Cloud® Paks by way of Red Hat OpenShift Container Platform

Generate mirror manifests

Note:

- If you want to install subsequent updates to your air-gapped environment, you must do a CASE `get` to get the image list when performing those updates. A registry namespace suffix can optionally be specified on the target registry to group mirrored images.
- Define the environment variable `$TARGET_REGISTRY` by running the following command:

  ```
  export TARGET_REGISTRY=<target-registry>
  ```

  The `<target-registry>` refers to the registry (hostname and port) where your images will be mirrored to and accessed by the oc cluster. For example setting TARGET_REGISTRY to `myregistry.com:5000/mynamespace` will create manifests such that images will be mirrored to the top-level namespace `mynamespace`.
- Run the following commands to generate mirror manifests to be used when mirroring from a bastion host (connected mirroring):

  ```
  oc ibm-pak generate mirror-manifests \
     $CASE_NAME \
     $TARGET_REGISTRY \
     --version $CASE_VERSION
  ```

  Example `~/.ibm-pak` directory structure for connected mirroring

  The `~/.ibm-pak` directory structure is built over time as you save CASEs and mirror. The following tree shows an example of the `~/.ibm-pak` directory structure for connected mirroring:

  ```
  tree ~/.ibm-pak
  /root/.ibm-pak
  ├── config
  │   └── config.yaml
  ├── data
  │   ├── cases
  │   │   └── YOUR-CASE-NAME
  │   │       └── YOUR-CASE-VERSION
  │   │           ├── XXXXX
  │   │           ├── XXXXX
  │   └── mirror
  │       └── YOUR-CASE-NAME
  │           └── YOUR-CASE-VERSION
  │               ├── catalog-sources.yaml
  │               ├── image-content-source-policy.yaml
  │               └── images-mapping.txt
  └── logs
     └── oc-ibm_pak.log
  ```

  Notes: A new directory `~/.ibm-pak/mirror` is created when you issue the `oc ibm-pak generate mirror-manifests` command. This directory holds the `image-content-source-policy.yaml`, `images-mapping.txt`, and `catalog-sources.yaml` files.

  Tip: If you are using a Red Hat® Quay.io registry and need to mirror images to a specific organization in the registry, you can target that organization by specifying:

  ```
     export ORGANIZATION=<your-organization>
     oc ibm-pak generate mirror-manifests
     $CASE_NAME
     $TARGET_REGISTRY/$ORGANIZATION
     --version $CASE_VERSION
  ```

You can also generate manifests to mirror images to an intermediate registry server, then mirroring to a final registry server. This is done by passing the final registry server as an argument to `--final-registry`:

```
   oc ibm-pak generate mirror-manifests \
      $CASE_NAME \
      $INTERMEDIATE_REGISTRY \
      --version $CASE_VERSION
      --final-registry $FINAL_REGISTRY
```

In this case, in place of a single mapping file (images-mapping.txt), two mapping files are created.

1. images-mapping-to-registry.txt

2. images-mapping-from-registry.txt

1. Run the following commands to generate mirror manifests to be used when mirroring from a file system (disconnected mirroring):

```
oc ibm-pak generate mirror-manifests \
   $CASE_NAME \
   file://local \
   --final-registry $TARGET_REGISTRY
```

Example `~/.ibm-pak` directory structure for disconnected mirroring

The following tree shows an example of the `~/.ibm-pak` directory structure for disconnected mirroring:

```
tree ~/.ibm-pak
/root/.ibm-pak
├── config
│   └── config.yaml
├── data
│   ├── cases
│   │   └── ibm-cp-common-services
│   │       └── 1.9.0
│   │           ├── XXXX
│   │           ├── XXXX
│   └── mirror
│       └── ibm-cp-common-services
│           └── 1.9.0
│               ├── catalog-sources.yaml
│               ├── image-content-source-policy.yaml
│               ├── images-mapping-to-filesystem.txt
│               └── images-mapping-from-filesystem.txt
└── logs
   └── oc-ibm_pak.log
```

Note: A new directory `~/.ibm-pak/mirror` is created when you issue the `oc ibm-pak generate mirror-manifests` command. This directory holds the `image-content-source-policy.yaml`, `images-mapping-to-filesystem.txt`, `images-mapping-from-filesystem.txt`, and `catalog-sources.yaml` files.

Tip: Some products support the ability to generate mirror manifests only for a subset of images using the `--filter` argument and image grouping. The `--filter` argument provides the ability to customize which images are mirrored during an air-gapped installation. As an example for this functionality `ibm-cloud-native-postgresql` CASE can be used, which contains groups that allow mirroring specific variant of `ibm-cloud-native-postgresql` (Standard or Enterprise). Use the `--filter` argument to target a variant of `ibm-cloud-native-postgresql` to mirror rather than the entire library. The filtering can be applied for groups and architectures. Consider the following command:

```
oc ibm-pak generate mirror-manifests \
   ibm-cloud-native-postgresql \
   file://local \
   --final-registry $TARGET_REGISTRY \
   --filter $GROUPS
```

The command was updated with a `--filter` argument. For example, for $GROUPS equal to `ibmEdbStandard` the mirror manifests will be generated only for the images associated with `ibm-cloud-native-postgresql` in its Standard variant. The resulting image group consists of images in the `ibm-cloud-native-postgresql` image group as well as any images that are not associated with any groups. This allows products to include common images as well as the ability to reduce the number of images that you need to mirror.

Note: You can use the following command to list all the images that will be mirrored and the publicly accessible registries from where those images will be pulled from:

```
oc ibm-pak describe $CASE_NAME --version $CASE_VERSION --list-mirror-images
```

Tip: The output of the preceding command will have two sections:

1. Mirroring Details from Source to Target Registry

2. Mirroring Details from Target to Final Registry. A connected mirroring path that does not involve a intermediate registry will only have the first section.

   Note down the `Registries found` sub sections in the preceding command output. You will need to authenticate against those registries so that the images can be pulled and mirrored to your local registry. See the next steps on authentication. The `Top level namespaces found` section shows the list of namespaces under which the images will be mirrored. These namespaces should be created manually in your registry (which appears in the Destination column in the above command output) root path if your registry does not allow automatic creation of namespaces.

Authenticating the registry

Complete the following steps to authenticate your registries:

1. Store authentication credentials for all source Docker registries.

   Your product might require one or more authenticated registries. The following registries require authentication:

   - `cp.icr.io`
   - `registry.redhat.io`
   - `registry.access.redhat.com`

   You must run the following command to configure credentials for all target registries that require authentication. Run the command separately for each registry:

   Note: The `export REGISTRY_AUTH_FILE` command only needs to run once.

   ```
   export REGISTRY_AUTH_FILE=<path to the file which will store the auth credentials generated
   on podman login>
   podman login <TARGET_REGISTRY>
   ```

   Important: When you log in to `cp.icr.io`, you must specify the user as `cp` and the password which is your Entitlement key from the IBM Cloud Container Registry. For example:

   ```
   podman login cp.icr.io
   Username: cp
   Password:
   Login Succeeded!
   ```

For example, if you export `REGISTRY_AUTH_FILE=~/.ibm-pak/auth.json`, then after performing `podman login`, you can see that the file is populated with registry credentials.

If you use `docker login`, the authentication file is typically located at `$HOME/.docker/config.json` on Linux or `%USERPROFILE%/.docker/config.json` on Windows. After `docker login` you should export `REGISTRY_AUTH_FILE` to point to that location. For example in Linux you can issue the following command:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

Table 3. Table 2. Directory description

Directory	Description
`~/.ibm-pak/config`	Stores the default configuration of the plug-in and has information about the public GitHub URL from where the cases are downloaded.
`~/.ibm-pak/data/cases`	This directory stores the CASE files when they are downloaded by issuing the `oc ibm-pak get` command.

Table 3. Table 2. Directory description (continued)

Directory	**Description**
`~/.ibm-pak/data/mirror`	This directory stores the image-mapping files, ImageContentSourcePolicy manifest in `image-content-source-policy.yaml` and CatalogSource manifest in one or more `catalog-sourcesXXX.yaml`. The files `images-mapping-to-filesystem.txt` and `images-mapping-from-filesystem.txt` are input to the oc `image mirror` command, which copies the images to the file system and from the file system to the registry respectively.
`~/.ibm-pak/data/logs`	This directory contains the `oc-ibm_pak.log` file, which captures all the logs generated by the plug-in.

Mirror images to final location

Complete the steps in this section on your host that is connected to both the local Docker registry and the Red Hat® OpenShift® Container Platform cluster.

1. Mirror images to the final location.

   - **For mirroring from a bastion host (connected mirroring):**

     Mirror images to the `TARGET_REGISTRY`:

     ```
     oc image mirror \
       -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping.txt \
       --filter-by-os '.*'  \
       -a $REGISTRY_AUTH_FILE \
       --insecure  \
       --skip-multiple-scopes \
       --max-per-registry=1 \
       --continue-on-error=true
     ```

     If you generated manifests in the previous steps to mirror images to an intermediate registry server followed by a final registry server, run the following commands:

     a. Mirror images to the intermediate registry server:

     ```
     oc image mirror \
       -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-to-registry.txt \
       --filter-by-os '.*'  \
       -a $REGISTRY_AUTH_FILE \
       --insecure  \
       --skip-multiple-scopes \
       --max-per-registry=1 \
       --continue-on-error=true
     ```

     b. Mirror images from the intermediate registry server to the final registry server:

     ```
     oc image mirror \
       -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-from-registry.txt \
       --filter-by-os '.*'  \
       -a $REGISTRY_AUTH_FILE \
       --insecure  \
       --skip-multiple-scopes \
       --max-per-registry=1 \
       --continue-on-error=true
     ```

The `oc image mirror --help` command can be run to see all the options available on the mirror command. Note that we use `continue-on-error` to indicate that the command should try to mirror as much as possible and continue on errors.

```
oc image mirror --help
```

Note: Sometimes based on the number and size of images to be mirrored, the `oc image mirror` might take longer. If you are issuing the command on a remote machine it is recommended that you run the command in the background with a nohup so even if network connection to your remote machine is lost or you close the terminal the mirroring will continue. For example, the below command will start the mirroring process in background and write the log to `my-mirror-progress.txt`.

```
nohup oc image mirror \
-f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping.txt \
-a $REGISTRY_AUTH_FILE \
--filter-by-os '.*' \
--insecure \
--skip-multiple-scopes \
--max-per-registry=1 \
--continue-on-error=true > my-mirror-progress.txt  2>&1 &
```

You can view the progress of the mirror by issuing the following command on the remote machine:

```
tail -f my-mirror-progress.txt
```

- **For mirroring from a file system (disconnected mirroring):**

  Mirror images to your file system:

  ```
  export IMAGE_PATH=<image-path>
  oc image mirror \
    -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-to-filesystem.txt \
    --filter-by-os '.*'  \
    -a $REGISTRY_AUTH_FILE \
    --insecure  \
    --skip-multiple-scopes \
    --max-per-registry=1 \
    --continue-on-error=true \
    --dir "$IMAGE_PATH"
  ```

  The `<image-path>` refers to the local path to store the images. For example, in the previous section if provided `file://local` as input during generate mirror-manifests, then the preceding command will create a subdirectory v2/local inside directory referred by `<image-path>` and copy the images under it.

The following command can be used to see all the options available on the mirror command. Note that `continue-on-error` is used to indicate that the command should try to mirror as much as possible and continue on errors.

```
oc image mirror --help
```

Note: Sometimes based on the number and size of images to be mirrored, the `oc image mirror` might take longer. If you are issuing the command on a remote machine, it is recommended that you run the command in the background with nohup so that even if you lose network connection to your remote machine or you close the terminal, the mirroring will continue. For example, the following command will start the mirroring process in the background and write the log to `my-mirror-progress.txt`.

```
export IMAGE_PATH=<image-path>
nohup oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-to-filesystem.txt \
  --filter-by-os '.*' \
  -a $REGISTRY_AUTH_FILE \
  --insecure \
  --skip-multiple-scopes \
```

```
   --max-per-registry=1 \
   --continue-on-error=true \
   --dir "$IMAGE_PATH" > my-mirror-progress.txt  2>&1 &
```

You can view the progress of the mirror by issuing the following command on the remote machine:

```
tail -f my-mirror-progress.txt
```

2. **For disconnected mirroring only:** Continue to move the following items to your file system:
   - The `<image-path>` directory you specified in the previous step
   - The `auth` file referred by `$REGISTRY_AUTH_FILE`
   - `~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-from-filesystem.txt`
3. **For disconnected mirroring only:** Mirror images to the target registry from file system

   Complete the steps in this section on your file system to copy the images from the file system to the `$TARGET_REGISTRY`. Your file system must be connected to the target docker registry.

   Important: If you used the placeholder value of TARGET_REGISTRY as a parameter to `--final-registry` at the time of generating mirror manifests, then before running the following command, find and replace the placeholder value of TARGET_REGISTRY in the file, `images-mapping-from-filesystem.txt`, with the actual registry where you want to mirror the images. For example, if you want to mirror images to `myregistry.com/mynamespace` then replace TARGET_REGISTRY with `myregistry.com/mynamespace`.

   a. Run the following command to copy the images (referred in the `images-mapping-from-filesystem.txt` file) from the directory referred by `<image-path>` to the final target registry:

   ```
   export IMAGE_PATH=<image-path>
   oc image mirror \
     -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-from-filesystem.txt \
     -a $REGISTRY_AUTH_FILE \
     --from-dir "$IMAGE_PATH" \
     --filter-by-os '.*' \
     --insecure \
     --skip-multiple-scopes \
     --max-per-registry=1 \
     --continue-on-error=true
   ```

Configure the cluster

1. Update the global image pull secret for your Red Hat OpenShift cluster. Follow the steps in Updating the global cluster pull secret.

   The documented steps in the link enable your cluster to have proper authentication credentials in place to pull images from your TARGET_REGISTRY as specified in the `image-content-source-policy.yaml` which you will apply to your cluster in the next step.
2. Create ImageContentSourcePolicy

   Important:

   - Before you run the command in this step, you must be logged into your OpenShift cluster. Using the `oc login` command, log in to the Red Hat OpenShift Container Platform cluster where your final location resides. You can identify your specific oc login by clicking the user drop-down menu in the Red Hat OpenShift Container Platform console, then clicking **Copy Login Command**.
     - If you used the placeholder value of TARGET_REGISTRY as a parameter to `--final-registry` at the time of generating mirror manifests, then before running the following command, find and replace the placeholder value of TARGET_REGISTRY in file, `~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml` with the actual registry where you want to mirror the images. For example, replace TARGET_REGISTRY with `myregistry.com/mynamespace`.

Run the following command to create ImageContentSourcePolicy:

```
   oc apply -f  ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-
policy.yaml
```

If you are using Red Hat OpenShift Container Platform version 4.7 or earlier, this step might cause your cluster nodes to drain and restart sequentially to apply the configuration changes.

3. Verify that the ImageContentSourcePolicy resource is created.

```
oc get imageContentSourcePolicy
```

4. Verify your cluster node status and wait for all the nodes to be restarted before proceeding.

```
oc get MachineConfigPool
```

```
$ oc get MachineConfigPool -w
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
master   rendered-master-53bda7041038b8007b038c08014626dc   True      False      False
3              3                   3                     0                      10d
worker   rendered-worker-b54afa4063414a9038958c766e8109f7   True      False      False
3              3                   3                     0                      10d
```

After the `ImageContentsourcePolicy` and global image pull secret are applied, the configuration of your nodes will be updated sequentially. Wait until all `MachineConfigPools` are in the `UPDATED=True` status before proceeding.

5. Go to the project where deployment has to be done:

   Note: You must be logged into a cluster before performing the following steps.

```
export NAMESPACE=<YOUR_NAMESPACE>
```

```
oc new-project $NAMESPACE
```

6. **Optional:** If you use an insecure registry, you must add the target registry to the cluster insecureRegistries list.

```
oc patch image.config.openshift.io/cluster --type=merge \
-p '{"spec":{"registrySources":{"insecureRegistries":["'${TARGET_REGISTRY}'"]}}}'
```

7. Verify your cluster node status and wait for all the nodes to be restarted before proceeding.

```
oc get MachineConfigPool -w
```

After the `ImageContentsourcePolicy` and global image pull secret are applied, the configuration of your nodes will be updated sequentially. Wait until all `MachineConfigPools` are updated.

At this point your cluster is ready for IBM Connect:Direct for UNIX deployment. The helm chart is present in `~/.ibm-pak/data/cases/$CASE_NAME/$CASE_VERSION/charts/ibm-connect-direct-1.2.x.tgz` directory. Use it for deployment. Copy it in current directory.

```
cp ~/.ibm-pak/data/cases/$CASE_NAME/$CASE_VERSION/charts/ibm-connect-direct-1.2.x.tgz .
```

Note: Replace with version information in above command.

8. Configuration required in Helm chart: To use the image mirroring in OpenShift cluster, helm chart should be configured to use the digest value for referring to container image. Set image.digest.enabled to true in values.yaml file or pass this parameter using Helm CLI.

Setting up a repeatable mirroring process

Once you complete a CASE save, you can mirror the CASE as many times as you want to. This approach allows you to mirror a specific version of the IBM Cloud Pak into development, test, and production stages using a private container registry.

Follow the steps in this section if you want to save the CASE to multiple registries (per environment) once and be able to run the CASE in the future without repeating the CASE save process.

1. Run the following command to save the CASE to ~/.ibm-pak/data/cases/$CASE_NAME/$CASE_VERSION which can be used as an input during the mirror manifest generation:

```
oc ibm-pak get \
$CASE_NAME \
--version $CASE_VERSION
```

2. Run the `oc ibm-pak generate mirror-manifests` command to generate the `image-mapping.txt`:

```
oc ibm-pak generate mirror-manifests \
$CASE_NAME \
$TARGET_REGISTRY \
--version $CASE_VERSION
```

   Then add the `image-mapping.txt` to the `oc image mirror` command:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping.txt \
  --filter-by-os '.*'  \
  -a $REGISTRY_AUTH_FILE \
  --insecure  \
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --continue-on-error=true
```

If you want to make this repeatable across environments, you can reuse the same saved CASE cache (~/.ibm-pak/$CASE_NAME/$CASE_VERSION) instead of executing a CASE save again in other environments. You do not have to worry about updated versions of dependencies being brought into the saved cache.

Applying Pod Security Standard or Creating Pod Security Policy for Kubernetes Cluster

Pod Security Standard should be applied to the namespace if Kubernetes cluster v1.25 and above is used. This helm chart has been certified with baseline security standards with enforce security level. For more details, refer to Pod Security Standards.

- In Kubernetes the Pod Security Policy (PSP) control is implemented as optional (but recommended). Click here for more information on Pod Security Policy.

  IBM CCS defines a custom Pod Security Policy which is the minimum set of permissions/ capabilities needed to deploy this chart and the Connect Direct for Unix container to function properly. This is the recommended PSP for this chart and it can be created on the cluster by cluster administrator.

  The PSP and cluster role for this chart is defined below. The cluster administrator can either use the snippets given below or the scripts provided in the Helm chart to create the PSP, cluster role and tie it to the namespace where deployment will be performed. In both the cases, same PSP and cluster role will be created. It is recommended to use the scripts in the Helm chart so that required PSP and cluster role is created without any issue.
- Below is the Custom `PodSecurityPolicy` definition that the cluster admin can use:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: ibm-connect-direct-psp
  labels:
    app: "ibm-connect-direct-psp"
spec:
  privileged: false
  allowPrivilegeEscalation: true
  hostPID: false
  hostIPC: false
  hostNetwork: false
  requiredDropCapabilities:
  allowedCapabilities:
  - FOWNER
  - SETUID
  - SETGID
  - DAC_OVERRIDE
  - CHOWN
  - AUDIT_WRITE
  - SYS_CHROOT
  allowedHostPaths:
  runAsUser:
    rule: MustRunAsNonRoot
  runAsGroup:
    rule: MustRunAs
    ranges:
    - min: 1
      max: 4294967294
  seLinux:
    rule: MustRunAs
  supplementalGroups:
    rule: MustRunAs
    ranges:
    - min: 1
      max: 4294967294
  fsGroup:
    rule: MustRunAs
    ranges:
    - min: 1
      max: 4294967294
  volumes:
  - configMap
  - downwardAPI
  - emptyDir
  - nfs
  - persistentVolumeClaim
  - projected
  - secret
  forbiddenSysctls:
  - '*'
```

- Custom ClusterRole for the custom PodSecurityPolicy

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "ibm-connect-direct-psp"
  labels:
    app: "ibm-connect-direct-psp"
rules:
- apiGroups:
  - policy
  resourceNames:
  - ibm-connect-direct-psp
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

- From the command line, you can run the setup scripts included in the Helm chart as cluster admin (untar the downloaded Helm chart archive).

```
ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/clusterAdministration/
createSecurityClusterPrereqs.sh
```

```
ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/namespaceAdministration/
createSecurityNamespacePrereqs.sh <Namespace where deployment will be performed>
```

Note: If the above scripts are not executable, you will need to make the scripts executable by executing following commands:

```
chmod u+x ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/namespaceAdministration/
createSecurityNamespacePrereqs.sh
```

```
chmod u+x ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/clusterAdministration/
createSecurityClusterPrereqs.sh
```

Creating security context constraints for Red Hat OpenShift Cluster

- The IBM Connect:Direct for Unix chart requires an SecurityContextConstraints (SCC) to be tied to the target namespace prior to deployment.

  Based on your organization security policy, you may need to decide the security context constraints for your OpenShift cluster.

  This chart has been verified on privileged SCC which comes with Redhat OpenShift. For more info, please refer this link. This chart defines a custom SCC which is the minimum set of permissions/capabilities needed to deploy this chart and the Connect Direct for Unix container to function properly. It is based on the predefined restricted SCC with extra required privileges. This is the recommended SCC for this chart and it can be created on the cluster by cluster administrator.

  The SCC and cluster role for this chart is defined below. The cluster administrator can either use the snippets given below or the scripts provided in the Helm chart to create the SCC, cluster role and tie it to the project where deployment will be performed. In both the cases, same SCC and cluster role will be created. It is recommended to use the scripts in the Helm chart so that required SCC and cluster role is created without any issue.
- Below is the Custom `SecurityContextConstraints` definition that the cluster admin can use

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: ibm-connect-direct-scc
  labels:
    app: "ibm-connect-direct-scc"
allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegedContainer: false
allowPrivilegeEscalation: true
allowedCapabilities:
- FOWNER
- SETUID
- SETGID
- DAC_OVERRIDE
- CHOWN
- SYS_CHROOT
- AUDIT_WRITE
defaultAddCapabilities: []
defaultAllowPrivilegeEscalation: false
forbiddenSysctls:
- "*"
fsGroup:
  type: MustRunAs
  ranges:
  - min: 1
    max: 4294967294
readOnlyRootFilesystem: false
requiredDropCapabilities:
- ALL
runAsUser:
  type: MustRunAsNonRoot
seLinuxContext:
  type: MustRunAs
supplementalGroups:
  type: MustRunAs
  ranges:
  - min: 1
    max: 4294967294
volumes:
- configMap
- downwardAPI
- emptyDir
- nfs
- persistentVolumeClaim
- projected
- secret
   priority: 0
```

- Custom ClusterRole for the custom `SecurityContextConstraints`

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "ibm-connect-direct-scc"
  labels:
    app: "ibm-connect-direct-scc"
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - ibm-connect-direct-scc
  resources:
  - securitycontextconstraints
  verbs:
  - use
```

- From the command line, you can run the setup scripts included in the Helm chart (untar the downloaded Helm chart archive).

```
ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/clusterAdministration/
createSecurityClusterPrereqs.sh
```

```
ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/namespaceAdministration/
createSecurityNamespacePrereqs.sh <Project name where deployment will be perfromed>
```

Note: If the above scripts are not executable, you will need to make the scripts executable by executing following commands:

```
chmod u+x ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/namespaceAdministration/
createSecurityNamespacePrereqs.sh
```

```
chmod u+x ibm-connect-direct/ibm_cloud_pak/pak_extensions/pre-install/clusterAdministration/
createSecurityClusterPrereqs.sh
```

Creating storage for Data Persistence

The containers are ephemeral entity, all the data inside the container will be lost when the containers are destroyed/removed, so data must be saved to Storage Volume using Persistent Volume. Persistent volume is recommended for Connect:Direct for UNIX deployment files. A Persistent Volume (PV) is a piece of storage in the cluster that is provisioned by an administrator or dynamically provisioned using storage classes. For more information see:

- Kubernetes - Persistent Volumes
- Red Hat OpenShift - Persistent Volume Overview

As a prerequisite, create a persistent volume supported via NFS, and host path mounted across all worker nodes. IBM Certified Container Software for CDU supports:

- Dynamic Provisioning using storage class
- Pre-created Persistent Volume
- Pre-created Persistent Volume Claim

- The only supported access mode is `ReadWriteOnce`

Dynamic Provisioning

Dynamic provisioning is supported using storage classes. To enable dynamic provisioning use following configuration for helm chart:

- **persistence.useDynamicProvisioning-** It must be set to true. By default, it is set to false, which means dynamic provisioning is disabled.
- **pvClaim.storageClassName-** The storage class is blank by default. Update this parameter value using valid storage class. Consult your cluster administrator for available storage class as required by this chart.
- **secret.certSecretName-** Specify the certificate secret required for Secure plus configuration or LDAP support. Update this parameter with valid certificate secret. Refer "Creating secret" on page 64 for more information.

Non-Dynamic Provisioning

Non-Dynamic Provisioning is supported using pre-created Persistent Volume and pre-created Persistent Volume Claim. The Storage Volume should have Connect:Direct for UNIX secure plus certificate files to be used for installation. Create a directory named "CDFILES" inside mount path and place certificate files in the created directory. Similarly, the LDAP certificates should be placed in same directory.

Using pre-created Persistent Volume- When creating Persistent Volume, make a note of the storage class and metadata labels, that are required to configure Persistent Volume Claim's storage class and label selector during deployment. This ensures that the claims are bound to Persistent Volume based on label match. These labels can be passed to helm chart either by `--set flag` or custom `values.yaml` file. The parameters defined in`values.yaml` for label name and its value are `pvClaim.selector.label` and `pvClaim.selector.value` respectively.

Refer below yaml templates for Persistent Volume creation. Customize as per your requirement.

Example: Create Persistent volume using NFS server

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <persistent volume name>
  labels:
    app.kubernetes.io/name: <persistent volume name>
    app.kubernetes.io/instance: <release name>
    app.kubernetes.io/managed-by: <service name>
    helm.sh/chart: <chart name>
    release: <release name>
    purpose: cdconfig
spec:
  storageClassName: <storage classname>
  capacity:
    storage: <storage size>
  accessModes:
    - ReadWriteOnce
  nfs:
    server: <NFS server IP address>
    path: <mount path>
```

Example: Create Persistent volume using Host Path

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <persistent volume name>
  labels:
    app.kubernetes.io/name: <persistent volume name>
    app.kubernetes.io/instance: <release name>
    app.kubernetes.io/managed-by: <service name>
    helm.sh/chart: <chart name>
    release: <release name>
    purpose: cdconfig
spec:
  storageClassName: <storage classname>
  capacity:
    storage: <storage size>
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: <mount path>
```

Invoke the following command to create a Persistent Volume:

Kubernetes:

```
kubectl create -f <peristentVolume yaml file>
```

OpenShift:

```
oc create -f <peristentVolume yaml file>
```

Using pre-created Persistent Volume Claim (PVC)- The existing PVC can also be used for deployment. The PV for PVC should have the certificate files as required for Connect:Direct for UNIX secure plus or LDAP TLS configuration. The parameter for pre-created PVC is `pvClaim.existingClaimName`. One should pass a valid PVC name to this parameter else deployment would fail.

Apart from required Persistent Volume, you can bind extra storage mounts using the parameters provided in `values.yaml`. These parameters are extraVolume and extraVolumeMounts. This can be a host path or a NFS type.

The deployment mounts following configuration/resource directories on the Persistent Volume -

- <install_dir>/work
- <install_dir>/ndm/security
- <install_dir>/ndm/cfg
- <install_dir>/ndm/secure+
- <install_dir>/process
- <install_dir>/file_agent/config
- <install_dir>/file_agent/log

When deployment is upgraded or pod is recreated in Kubernetes based cluster then, only the data of above directories are saved/persisted on Persistent Volume.

Setting permission on storage

When shared storage is mounted on a container, it is mounted with same POSIX ownership and permission present on exported NFS directory. The mounted directories on container may not have correct owner and permission needed to perform execution of scripts/binaries or writing to them. This situation can be handled as below -

- Option A: The easiest and undesirable solution is to have open permissions on the NFS exported directories.

  ```
  chmod -R 777 <path-to-directory>
  ```

- Option B: Alternatively, the permissions can be controlled at group level leveraging the supplementalGroups and fsGroup setting. For example - if we want to add GID to supplementalGroups or fsGroup, it can be done using **storageSecurity.supplementalGroups** or **storageSecurity.fsGroup**.

Apart from above recommendation, during deployment, a default admin user cduser with group cduser is created. The default UID and GID of cduser is 45678.

Example:

If you have a user, host-user having UID and GID set to 2000 and 4000 respectively on the host system. There are 'upload' (for files to be sent) and 'download' (for files to be received) directories created by this user. These directories can be mounted on the CDU container so that these directories are available inside container.

In order to give cduser access to these directories inside the container, the UID and GID of cduser can be set to `2000` and `4000` respectively. Similarly, you can set the UID and GID of `appuser` which is a non-admin user of Connect:Direct for Unix running inside container. The user `appuser` is created only if you have passed its password to be set inside container using cdai_appuser_pwd parameter in cd_param_file.

The UID and GID of cduser can be set to some real user on the host system. By setting the same UID and GID as on the host system, the host system user would have suitable permissions on the files present on the host path so that this user can be used to edit any setting or configuration files related to Connect:Direct for Unix running inside container.

Root Squash NFS support

Root squash NFS is secure NFS share when root privileges are shrinked similar to unprivileged user. Also, this user is mapped to nfsnobody or nobody user on the system. So, you cannot perform operations like changing the ownership of any files/directories.

Connect:Direct for UNIX helm chart can be deployed on root squash NFS. Since, the ownership of files/directories mounted in container would be mounted as nfsnobody or nobody. The POSIX group ID of the root squash NFS share should be added to Supplemental Group list statefulset using storageSecurity.supplementalGroup in values.yam l file. Similarly any extra NFS share is mounted it proper rad/write permission can be provide to container user using supplemental group only.ups

Creating secret

Passwords are used for KeyStore, by Administrator to connect to Connect:Direct server, and to decrypt certificates files.

To separate application secrets from the Helm Release, a Kubernetes secret must be created based on the examples given below and be referenced in the Helm chart as `secret.secretName` value.

To create Secrets using the command line, follow the steps below:

1. Create a template file with Secret defined as described in the example below:

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: <secret name>
   type: Opaque
   data:
     admPwd: <base64 encoded password>
     crtPwd: <base64 encoded password>
     keyPwd: <base64 encoded password>
     appUserPwd: <base64 encoded password>
   ```

   Here:

   - `admPwd` refers to the password that will be set for the Admin user 'cduser' after a successful deployment
   - `crtPwd` refers to the password used to decrypt certificate files
   - `keyPwd` refers to the Key Store password
   - `appUserPwd` refers to password for a non-admin Connect:Direct user. This user will only be created inside container if `appUserPwd` is defined in secret yaml to create Connect:Direct secret object.
   - After the secret is created, delete the yaml file for security reasons. The password entered for `admPwd` is set as password for the user `cduser` at pod initialization.

Note: Base64 encoded passwords need to be generated manually by invoking the below command:

```
echo -n “<password>” | base64
```

Use the output of this command in the <secret yaml file>.

2. Run the following command to create the Secret:

   Kubernetes:

   ```
   kubectl create -f <secret yaml file>
   ```

   OpenShift

   ```
   oc create -f <secret yaml file>
   ```

   To check the secret created invoke the following command:

   ```
   kubectl get secrets
   ```

   For more details see, Secrets.

   Default Kubernetes secrets management has certain security risks as documented here, Kubernetes Security.

   Users should evaluate Kubernetes secrets management based on their enterprise policy requirements and should take steps to harden security.

3. For dynamic provisioning, one more secret resource needs to be created for all certificates (secure plus certificates and LDAP certificates). It can be created using below example as required -

   Kubernetes

   ```
   kubectl create secret generic cd-cert-secret --from-file=certificate_file1=/path/to/
   certificate_file1 --from-file=certificate_file2=/path/to/certificate_file2
   ```

   OpenShift

   ```
   oc create secret generic cd-cert-secret --from-file=certificate_file1=/path/to/
   certificate_file1 --from-file=certificate_file2=/path/to/certificate_file2
   ```

   Note: The secret resource name created above. It should be referenced by Helm chart for dynamic provisioning using parameter `secret.certSecretName'

Configuring - Understanding values.yaml

Following table describes configuration parameters listed in `values.yaml` file in Helm charts and are used to complete installation. Use the following steps to complete this action:

- Specify parameters that need to be overridden using the `--set key=value[,key=value]` argument at Helm install.

  Example:

  helm version 2

  ```

  helm install --name <release-name> \
  --set cdArgs.cport=9898 \
  ...
  ibm-connect-direct-1.1.x.tgz
  ```

helm version 3

```
helm install <release-name> \
--set cdArgs.cport=9898 \
...
ibm-connect-direct-1.1.x.tgz
```

- Alternatively, provide a YAML file with values specified for configurable parameters when you install a Chart. The values.yaml file can be obtained from the helm chart itself using the following command-

  For Online Cluster

  ```
  helm inspect values ibm-helm/ibm-connect-direct > my-values.yaml
  ```

  For Offline Cluster

  ```
  helm inspect values <path to ibm-connect-direct Helm chart> > my-values.yaml
  ```

  Now, edit the parameters in my-values.yaml file and use it for installation.

  Example

  helm version 2

  ```
  helm install --name <release-name> -f my-values.yaml ... ibm-connect-direct-1.1.x.tgz
  ```

  helm version 3

  ```
  helm install <release-name> -f my-values.yaml ... ibm-connect-direct-1.1.x.tgz
  ```

- To mount extra volumes use any of the following templates.

  For Hostpath

  ```
  extraVolumeMounts:
    - name: <name>
      mountPath: <path inside container>
  extraVolume:
    - name: <name same as name in extraVolumeMounts>
      hostPath:
        path: <path on host machine>
        type: DirectoryOrCreate
  ```

  For NFS Server

  ```
  extraVolumeMounts:
    - name: <name>
      mountPath: <path inside container>
  extraVolume:
    - name: <name same as name in extraVolumeMounts>
      nfs:
        path: <nfs data path>
        server: <server ip>
  ```

  Alternatively, this can also be done using --set flag.

  Example

  ```
  helm install --name <release-name> --set
  extraVolume[0].name=<name>,extraVolume[0].hostPath.path=<path on host
  machine>,extraVolume[0].hostPath.type="DirectoryOrCreate",extraVolumeMounts[0].name=<name
  same as name in extraVolume>,extraVolumeMounts[0].mountPath=<path inside container> \
  ...
  ibm-connect-direct-1.1.x.tgz
  ```

OR

```
helm install --name <release-name> --
set extraVolume[0].name=<name>,extraVolume[0].nfs.path=<nfs data
path>,extraVolume[0].nfs.server=<NFS server IP>, extraVolumeMounts[0].name=<name same as name
in extraVolume>,extraVolumeMounts[0].mountPath=<path inside container> \
...
ibm-connect-direct-1.1.x.tgz
```

If extra volume is mounted, please make sure container user (cduser/appuser) has proper read/write permission. The required permissions can be provided to the container user supplemental groups or fs groups as applicable. For example - if an extra NFS share is being mounted where customer user resides and its POSIX group ID 3535, then during deployment add this group ID as supplemental group to ensure container user to be member of this group.

- To use Port Check Ignore List feature, configure as below :

```
service.externalTrafficPolicy: "Local"
```

Use external IP which should be node's IP where pod is deployed as Port Check Ignore List IP addresses in the initparm.cfg after successful deployment.

Parameter	Description	Default Value
licenseType	Specify prod or non-prod for production or non-production license type respectively	prod
license	License agreement. Set true to accept the license.	false
env.timezone	Timezone	UTC
arch	Node Architecture	amd64
replicaCount	Number of deployment replicas	1
image.repository	Image full name including repository	
image.tag	Image tag	
digest.enabled	Enable/Disable digest of image to be used	false
digest.value	The digest value for the image	
image.imageSecrets	Image pull secrets	
image.pullPolicy	Image pull policy	IfNotPresent
cdArgs.nodeName	Node name	cdnode
cdArgs.crtName	Certificate file name	
cdArgs.localCertLabel	Specify certificate import label in keystore	Client-API
cdArgs.cport	Client Port	1363
cdArgs.sport	Server Port	1364
saclConfig	Configuration for SACL	n
cdArgs.configDir	Directory for storing Connect:Direct configuration files	CDFILES

Parameter	**Description**	**Default Value**
appUser.name	Name of Non-Admin Connect:Direct User	appuser
appUser.uid	UID of Non-Admin Connect:Direct User	
appUser.gid	GID of Non-Admin Connect:Direct User	
storageSecurity.fsGroup	Group ID for File System Group	45678
storageSecurity.supplementalGroups	Group ID for Supplemental group	5555
persistence.enabled	To use persistent volume	true
pvClaim.existingClaimName	Provide name of existing PV claim to be used	
persistence.useDynamicProvisioning	To use storage classes to dynamically create PV	false
pvClaim.accessMode	Access mode for PV Claim	ReadWriteOnce
pvClaim.storageClassName	Storage class of the PVC	
pvClaim.selector.label	PV label key to bind this PVC	
pvClaim.selector.value	PV label value to bind this PVC	
pvClaim.size	Size of PVC volume	100Mi
service.type	Kubernetes service type exposing ports	LoadBalancer
service.apiport.name	API port name	api
service.apiport.port	API port number	1363
service.apiport.protocol	Protocol for service	TCP
service.ftport.name	Server (File Transfer) Port name	ft
service.ftport.port	Server (File Transfer) Port number	1364
service.ftport.protocol	Protocol for service	TCP
service.loadBalancerIP	Provide the LoadBalancer IP	
service.loadBalancerSourceRanges	Provide Load Balancer Source IP ranges	[]
service.annotations	Provide the annotations for service	{}
service.externalTrafficPolicy	Specify if external Traffic policy is needed	
service.sessionAffinity	Specify session affinity type	ClientIP
service.externalIP	External IP for service discovery	[]
networkPolicy.from	Provide from specification for network policy for ingress traffic	[]

Parameter	Description	Default Value
networkPolicy.to	Provide to specification for network policy for egress traffic	[]
secret.certSecretName	Name of secret resource of certificate files for dynamic provisioning	
secret.secretName	Secret name for Connect:Direct password store	
resources.limits.cpu	Container CPU limit	500mi
resources.limits.memory	Container memory limit	2000Mi
resources.limits.ephemeral-storage	Specify ephemeral storage limit size for pod's container	"5Gi"
resources.requests.cpu	Container CPU requested	500m
resources.requests.memory	Container Memory requested	2000Mi
resources.requests.ephemeral-storage	Specify ephemeral storage request size for pod's container	"3Gi"
serviceAccount.create	Enable/disable service account creation	true
serviceAccount.name	Name of Service Account to use for container	
extraVolumeMounts	Extra Volume mounts	
extraVolume	Extra volumes	
affinity.nodeAffinity.required DuringSchedulingIgnoredDuring Execution	k8sPodSpec.nodeAffinity.require d DuringSchedulingIgnoredDuring Execution	
affinity.nodeAffinity.preferred DuringSchedulingIgnoredDuring Execution	k8sPodSpec.nodeAffinity.preferr ed DuringSchedulingIgnoredDuring Execution	
affinity.podAffinity.required DuringSchedulingIgnoredDuring Execution	k8s PodSpec.podAntiAffinity. requiredDuringSchedulingIgnore d DuringExecution	
affinity.podAffinity.preferred DuringSchedulingIgnoredDuring Execution	k8sPodSpec.podAntiAffinity. preferredDuringScheduling IgnoredDuringExecution	

Parameter	Description	Default Value
affinity.podAntiAffinity.required DuringSchedulingIgnoredDuring Execution	k8sPodSpec.podAntiAffinity. requiredDuringSchedulingIgnored DuringExecution	
affinity.podAntiAffinity.preferred DuringSchedulingIgnoredDuring Execution	k8sPodSpec.podAntiAffinity. preferredDuringSchedulingIgnored DuringExecution	
livenessProbe.initialDelaySeconds	Initial delay for liveness	45
livenessProbe.timeoutSeconds	Timeout for liveness	5
livenessProbe.periodSeconds	Time period for liveness	15
readinessProbe.initialDelaySeconds	Initial delays for readiness	40
readinessProbe.timeoutSeconds	Timeout for readiness	5
readinessProbe.periodSeconds	Time period for readiness	25
route.enabled	Route for OpenShift Enabled/ Disabled	false
cduser.uid	UID for cduser	45678
cduser.gid	GID for cduser	45678
ldap.enabled	Enable/Disable LDAP configuration	false
ldap.host	LDAP server host	
ldap.port	LDAP port	
ldap.domain	LDAP Domain	
ldap.tls	Enable/Disable LDAP TLS	false
lap.caCert	LDAP CA Certificate name	
ldap.clientValidation	Enable/Disable LDAP Client Validation	false
ldap.clientCert	LDAP Client Certificate name	
ldap.clientKey	LDAP Client Certificate key name	
extraLabels	Provide extra labels for all resources of this chart	{}
cdfa.fileAgentEnable	Specify y/n to Enable/Disable File Agent	n

Affinity

The chart provides ways in form of node affinity, pod affinity and pod anti-affinity to configure advance pod scheduling in Kubernetes. See, Kubernetes documentation for details.

Note: For exact parameters, its value and its description, please refer to values.yaml file present in the helm chart itself. Untar the helm chart package to see this file inside chart directory.

Understanding LDAP deployment parameters

This section demonstrates the steps required to implement the PAM and SSSD configuration with Connect:Direct UNIX to authenticate external user accounts through OpenLDAP.

- Updating `initparam` file: When the LDAP authentication is enabled, the container startup script automatically updates the `initparam` configuration to support the PAM module. The following line is added to `initparam.cfg`:

  ```
  ndm.pam:service=login:
  ```

- The following packages are pre-installed in the container image to enable the LDAP support:

  ```
  openldap-client, sssd, sssd-ldap, openssl-perl, authselect
  ```

- The following default configuration file (`/etc/sssd/sssd.conf`) is added to the image. You must replace the values highlighted in bold with the values of environment variables as explained in next section.

  ```
  [domain/default]
  id_provider = ldap
  autofs_provider = ldap
  auth_provider = ldap
  chpass_provider = ldap
  ldap_uri = LDAP_PROTOCOL://LDAP_HOST:LDAP_PORT
  ldap_search_base = LDAP_DOMAIN
  ldap_id_use_start_tls = True
  ldap_tls_cacertdir = /etc/openldap/certs
  ldap_tls_cert = /etc/openldap/certs/LDAP_TLS_CERT_FILE
  ldap_tls_key = /etc/openldap/certs/LDAP_TLS_KEY_FILE
  cache_credentials = True
  ldap_tls_reqcert = allow
  ```

- Description of the Certificates required for the configuration:
  - Mount certificates inside CDU Container:
    - Copy the certificates needed for LDAP configuration in the mapped directory which is used to share the Connect:Direct Unix secure plus certificates (CDFILES/cdcert directory by default).
  - DNS resolution: If TLS is enabled and hostname of LDAP server is passed as “ldap.host”, then it must be ensured that the hostname is resolved inside the container. It is the responsibility of Cluster Administrator to ensure DNS resolution inside pod's container.
  - Certificates creation and configuration: This section provides a sample way to generate the certificates:
    - LDAP_CACERT - The root and all the intermediate CA certificates needs to be copied in one file.
    - LDAP_CLIENT_CERT – The client certificate which the server must be able to validate.
    - LDAP_CLIENT_KEY – The client certificate key.
  - Use the below new parameters for LDAP configuration:
    - ldap.enabled
    - ldap.host
    - ldap.port
    - ldap.domain
    - ldap.tls
    - ldap.caCert
    - ldap.clientValidation
    - ldap.clientCert

- ldpa.clientKey

Installing IBM Connect:Direct for Unix using Helm chart

After completing all Pre-installation tasks, you can deploy the IBM Certified Container Software for Connect:Direct for UNIX by invoking following command:

Helm version 2

```
helm install --name my-release --set license=true,image.repository=<reponame>
image.tag=<image tag>,cdArgs.crtName=<certificate name>,image.imageSecrets=<image pull
secret>,secret.secretName=<C:D secret name> ibm-connect-direct-1.2.x.tgz
or
helm install --name my-release ibm-connect-direct-1.2.x.tgz -f my-values.yaml
```

Helm version 3

```
helm install my-release --set license=true,image.repository=<reponame>
image.tag=<image tag>,cdArgs.crtName=<certificate name>,image.imageSecrets=<image pull
secret>,secret.secretName=<C:D secret name> ibm-connect-direct-1.2.x.tgz
or
helm install my-release ibm-connect-direct-1.2.x.tgz -f my-values.yaml
```

This command deploys **ibm-connect-direct-1.2.x.tgz** chart on the Kubernetes cluster using the default configuration. "Creating storage for Data Persistence" on page 61 lists parameters that can be configured at deployment.

Mandatory parameters required at the helm install command:

Parameter	Description	Default Value
license	License agreement for IBM Certified Container Software	false
image.repository	Image full name including repository	
image.tag	Image tag	
cdArgs.crtName	Key Certificate file name	
image.imageSecrets	Image pull secrets	
secret.secretName	Secret name for Connect:Direct password store	

Validating the Installation

After the deployment procedure is complete, you should validate the deployment to ensure that everything is working according to your needs. The deployment may take approximately 4-5 minutes to complete.

To validate if the Certified Container Software deployment using Helm charts is successful, invoke the following commands to verify the status (STATUS is DEPLOYED) for a Helm chart with release, **my-release** and namespace, **my-namespace.**

- Check the Helm chart release status by invoking the following command and verify that the STATUS is `DEPLOYED`:

  ```
  helm status my-release
  ```

- Wait for the pod to be ready. To verify the pods status (READY) use the dashboard or through the command line interface by invoking the following command:

  ```
  kubectl get pods -l release my-release -n my-namespace -o wide
  ```

- To view the service and ports exposed to enable communication in a pod invoke the following command:

  ```
  kubectl get svc -l release= my-release -n my-namespace -o wide
  ```

  The screen output displays the external IP and exposed ports under EXTERNAL-IP and PORT(S) column respectively. If external LoadBalancer is not present, refer Master node IP as external IP.

Exposed Services

If required, this chart can create a service of ClusterIP for communication within the cluster. This type can be changed while installing chart using service.type key defined in values.yaml. There are two ports where IBM Connect:Direct processes run. API port (1363) and FT port (1364), whose values can be updated during chart installation using service.apiport.port or service.ftport.port.

IBM Connect:Direct for Unix services for API and file transfer can be accessed using LoadBalancer or external IP and mapped API and FT port. If external LoadBalancer is not present then refer to Master node IP for communication.

Note: NodePort service type is not recommended. It exposes additional security concerns and is hard to manage from both an application and networking infrastructure perspective.

DIME and DARE Security Considerations

This topic provides security recommendations for setting up Data In Motion Encryption (DIME) and Data At Rest Encryption (DARE). It is intended to help you create a secure implementation of the application.

1. All sensitive application data at rest is stored in binary format so user cannot decrypt it. This chart does not support encryption of user data at rest by default. Administrator can configure storage encryption to encrypt all data at rest.
2. Data in motion is encrypted using transport layer security (TLS 1.3). For more information see, Secure Plus.

Post-installation tasks

The post deployment configuration steps can be performed via:

- **Connect Direct Web services**
  - Login to the Connect Direct Web services using the master Load Balancer IP/External IP address and the port to which container API port (1363) is mapped. For configuration steps, see Connect:Direct Web Services Help Videos.
  - Issue the following command to get the external IP address

    ```
    kubectl get svc
    ```

- **Attaching to the container**

  Follow the steps given below:

  - Issue the following command to get the pod name

    ```
    kubectl get pods -n <namespace>
    ```

  - Issue the following command to attach to the container

    ```
    kubectl exec -it <pod name> bash
    ```

- **Restarting Connect:Direct services inside container**

- Sometime, few configurations of IBM Connect:Direct for UNIX needs its services to be restarted. To restart the Connect:Direct service in container, access container terminal either from the command line or OCP dashboard. Inside the container run the below command:

  ```
  touch /cdinstall/.cdrecycle
  ```

- After creating this file in the container, CDPMGR services can be stopped by login to direct prompt using the below command:

  ```
  /opt/cdunix/ndm/bin/direct
  ```

- Then type `stop;` and press **enter** key. Connect:Direct service will be stopped inside container.
- Verify the changes and then restart CDPMGR process using the below command:

  ```
  /opt/cdunix/ndm/bin/cdpmgr -i /opt/cdunix/ndm/cfg/<nodename>/initparm.cfg
  ```

  Note: Pass Connect:Direct Nodename of the container in the above command.
- Consider, liveness and readiness probes configuration before stopping CDPMGR service. If Connect:Direct service remains unavailable beyond liveness and readiness configuration then it would result in pod restart.
- After Connect:Direct service is restarted, delete the created file using the below command:

  ```
  rm -f /cdinstall/.cdrecycle
  ```

Upgrade, Rollback, and Uninstall

After deploying IBM certified container software for Connect:Direct for UNIX, you can perform following actions:

- Upgrade, when you wish to move to a new release
- Rollback, when you wish to recover the previous release version in case of failure
- Uninstall, when you wish to uninstall

Upgrade – Upgrading a Release

To upgrade the chart ensure that the pre-installation tasks requirements are in-place on the cluster (Kubernetes or OpenShift). Following things must be considered while following pre-installation tasks requirement for upgrade:

1. Since, upgrade takes backup of configuration data on the Persistent Volume. Ensure that you have sufficient space available to accommodate the backup and running IBM Connect:Direct for UNIX data. A copy of backup is kept on Persistent Volume to enable rollback in case of upgrade failures.

   Example: The default minimum Persistent Volume size requirement for new deployment is 100Mi. We can just double it for upgrade ie. 200Mi.
2. Re-run the PodSecurityPolicy/SecurityContextConstraints scripts to ensure that the any new changes are in-place in namespace/project on Kubernetes/OpenShift cluster respectively.
3. Check the CD secrets are still valid and available on the cluster.
4. Depending upon the accessibility of the public internet on the cluster. The upgrade procedure can be Online upgrade and Offline upgrade.

Online upgrade

You have access to the public internet on the cluster. Thus, you have access to Entitled registry and IBM public GitHub repository. Follow these steps to upgrade the chart with release name **my-release**:

1. Update the local repo:

```
helm repo update
```

2. Download the newer helm chart:

   ```
   helm pull ibm-helm/ibm-connect-direct
   ```

   The helm chart gets pulled in current directory.
3. Untar the chart and run the PodSecurityPolicy/SecurityContextConstraints scripts to ensure any new required change is in-place on the cluster.

   Refer “Applying Pod Security Standard or Creating Pod Security Policy for Kubernetes Cluster ” on page 58 and “Creating security context constraints for Red Hat OpenShift Cluster ” on page 60 as applicable on the cluster.
4. Upgrade the chart.

   ```
   helm upgrade my-release ibm-connect-direct-1.2.x.tgz --reuse-values -f myvalues.yaml
   ```

Offline upgrade

You don't have access to the public internet on the cluster. Thus, Entitled registry and public IBM GitHub registry cannot be accessed from cluster. So, you need to follow the Offline (Airgap) Cluster procedure to get the installation files. Follow these steps to upgrade the chart with release name my-release:

1. After you have the installation files, go inside the charts directory used for downloading the files.

   ```
   cd <download directory>/charts
   ```

   Download directory is directory where the installation files have been downloaded.
2. Untar the chart and run the PodSecurityPolicy/SecurityContextConstraints scripts to ensure any new required change is in-place on the cluster. Refer “Applying Pod Security Standard or Creating Pod Security Policy for Kubernetes Cluster ” on page 58 and “Creating security context constraints for Red Hat OpenShift Cluster ” on page 60 as applicable for the cluster.
3. Upgrade the chart:

   ```
   helm upgrade my-release ibm-connect-direct-1.2.x.tgz --reuse-values -f myvalues.yaml
   ```

   Refer steps mentioned in “Validating the Installation” on page 72 for validating the upgrade.

Note: For both online and offline processes:

- Do not change/update the values of Connect Direct configuration parameters.
- If any new parameters are introduced in the new chart and you are upgrading using new chart. Then, all those new parameters should be either passed with "--set" option or using a yaml file with "-f" option. The parameter can have default values as specified in the new chart or you can change the values as per your configuration requirement.

Rollback – Recovering on Failure

1. To rollback a chart with release name **my-release** to a previous revision invoke the following command:

   ```
   helm rollback my-release <previous revision number>
   ```

2. To get the revision number execute the following command:

   ```
   helm history my-release
   ```

   Note: The rollback is only supported to a previous release. Subsequent rollbacks are not supported.

Rollback from Connect Direct Unix v6.1 is only supported if it was upgraded from Connect Direct Unix 6.0 iFix026 and later releases.

Uninstall – Uninstalling a Release

To uninstall/delete a Chart with release name **my-release** invoke the following command:

Helm version 2

```
helm delete --purge my-release
```

Helm version 3

```
helm delete my-release
```

Note:

This command removes all the Kubernetes components associated with the Chart and deletes the Release. Certain Kubernetes resources created as an installation prerequisite for the Chart and a helm hook ie ConfigMap will not be deleted using the `helm delete` command. Delete these resources only if they are not required for further deployment of IBM Certified Container Software for Connect:Direct UNIX. If deletion is required, you have to manually delete the following resources:

- The persistent volume
- The secret
- The Config Map

Known Limitations

- Scalability is supported with the conventional Connect:Direct for Unix release with load balancer service.
- High availability can be achieved in an orchestrated environment.
- IBM Connect:Direct for Unix chart is supported with only 1 replica count.
- IBM Connect:Direct for Unix chart supports only the x64 architecture.
- Adding Connect:Direct users at runtime is not supported.
- Interaction with IBM Control Center Director is not supported.
- The container does not include X-Windows support, so SPAdmin cannot run inside the container. SPCli will run in the container and Connect Direct Web Services (CDWS) can be used outside the container in place of using SPAdmin.
- Extension of Connect:Direct containers by customers is not supported. If you want any modifications or updates to the containers you need to raise an enhancement request.

Migrating to Connect:Direct for UNIX using Certified Container Software

Follow the steps given below to create a backup and restore IBM Connect:Direct for Unix using Certified Container Software:

1. Create a backup

   To create a backup of configuration data and other information such as stats and TCQ, present in the persistent volume, follow the steps given below:

   a. Go to mount path of Persistent Volume.
   b. Make copy of the following directories and store them at a secured location:
      - WORK

- SECURITY
- CFG
- SECPLUS
- PROCESS
- FACONFIG
- FALOG

  Note:

  – Update the various values in `initparm.cfg`, `netmap.cfg` and `ndmapi.cfg` for the C:D installation path, hostname/IP and port numbers. The installation path is `` `/opt/cdunix/` `` , hostname would be <helm-release-name>-ibm-connect-direct-0 and client port is 1363 and server port is 1364.

  – The nodename of Connect:Direct for Unix running on traditional system should be same while migrating it inside container.

  – If Connect:Direct for UNIX is installed in a conventional mode, create a backup of the following directories:

    - <install_dir>/work
    - <install_dir>/ndm/security
    - <install_dir>/ndm/cfg
    - <install_dir>/ndm/secure+
    - <install_dir>/process
    - <install_dir>/file_agent/config
    - <install_dir>/file_agent/log

    A file must be created in a work directory before taking backup. The file can be created by invoking the following command:

    ```
    <path to cdunix install directory>/etc/cdver > <path to cdunix install directory>/
    work/saved_cdunix_version
    ```

2. Restore the data in a new deployment

   To restore data in a new deployment, follow the steps given below:

   a. Create a Persistent Volume.

   b. Copy all the backed-up directories to the mount path of Persistent Volume.

3. For other prerequisites such as secrets see, Pre-installation tasks.
4. Upgrade to Certified Container Software

   Create a new instance of chart using the following helm CLI command:

   Helm version 2

   ```
   helm install --name <release-name> --set license=true,image.repository=<reponame>
   image.tag=<image tag>,cdArgs.crtName=<certificate name>,image.imageSecrets=<image pull
   secret>,secret.secretName=<C:D secret name> ibm-connect-direct-1.1.x.tgz
   ```

   Helm version 3

   ```
   helm install <release-name> --set license=true,image.repository=<reponame>
   image.tag=<image tag>,cdArgs.crtName=<certificate name>,image.imageSecrets=<image pull
   secret>,secret.secretName=<C:D secret name> ibm-connect-direct-1.1.x.tgz
   ```

Note: The nodename of Connect:Direct for Unix running on traditional system should be same for migration to IBM Certified Container Software. Configure the nodename using "cdArgs.nodeName" parameter.

Migration from Helm 2 to Helm 3

This migration only applies if you have IBM Certified Container Software instances/releases that are running on Helm 2 and you want to use Helm 3 release. It is the responsibility of the Cluster Administrator to decide on the migration strategy according to deployment requirement and specific use case scenario. Although, there is a detailed and exhaustive documentation provided by the Helm on migration. Please refer the this link, Helm.

Troubleshooting your deployment in an IBM Certified container

The following sections describes troubleshooting tips for issues that most frequently occur when deploying Connect:Direct for UNIX using an IBM Certified Container Software.

Troubleshooting IBM Certified Container Software

The following table describes possible reasons for occurrence and troubleshooting tips for issues that most frequently occur when deploying Connect:Direct for UNIX in an IBM Certified Container Software.

The quickest way to check the reason of a failed installation is to check the container logs or `CDStartup.log` in the `work` directory on Storage Volume mapped path.

Table 4. Troubleshooting deployment and functional issues

Issue	**Possible Reason**	**Solution/Workaround**
Deployment issues		
Connect Direct not running. Exiting.	Installation of Connect:Direct failed with a non-zero rc.	Look for the issue in the container logs. kubectl/oc logs <cdu pod name>
File "/opt/cdfiles/<certificate name>" is not a PEM key certificate.	The certificate in use is not in recommended .PEM format.	Use a certificate in PEM format for installation. See, Generating certificate in PEM format.
'Permission denied' error when deploying C:D using certified container software when SELinux is set to enforced	If SELinux policy is set to enforced, the host mounted path is not writable.	Run the following command to set the proper SELinux policy type label to be attached to the host directory: chcon -Rt svirt_sandbox_file_t <host mounted path>
'helm install' command fails to deploy C:D	Some mandatory parameters are missing in the `helm install` command.	Check that all mandatory parameters are provided when executing the helm install command. For more information see, "Installing IBM Connect:Direct for Unix using Helm chart" on page 72.
Pod recovery fails when persistent volume is mapped to the host path.	Possible reasons could be that the pod recovered on a different worker node and the persistent volume was mapped to a different host path.	NFS storage is the preferred storage volume. If NFS server is not available, set the Pod Affinity such that the pod is always scheduled on the same worker node where persistent volume can be accessed.

Table 4. Troubleshooting deployment and functional issues (continued)

Issue	**Possible Reason**	**Solution/Workaround**
Permission error occurs when trying to create a persistent volume or SecurityContextConstraints in an OpenShift environment	Openshift Admin user was used to create persistent volume or SecurityContextConstraints	Only the cluster Admin has privileges to create persistent volume and SecurityContextConstraints. Attempt creating these as a cluster Admin. For more information see, "Creating security context constraints for Red Hat OpenShift Cluster " on page 60.
Deployment fails with following error: SPCLI> 20200401 17:19:17 8 CDAI007E Secure+ configuration failed. 20200401 17:19:17 0 CDAI010I createExitStatusFile() entered. 20200401 17:19:17 0 CDAI010I createExitStatusFile() exited. command terminated with exit code 137	The certificate file is not valid. Check if conventional Connect:Direct for UNIX can be installed using the automated install procedure using this certificate.	Use the correct certificate file. Check the chain sequence for a chained certificate.
Functional Issues		
Error encountered during Connect:Direct container node recovery.	1. Configurations of the previous container are not mapped to Storage Volume. 2. Hostname provided does not match the hostname of the destroyed/removed container 3. Passwords missing in `cd_param_file`	The configurations of a container should be mapped to the Storage Volume so that the container can be recovered in future. Hostname of the removed/ destroyed container should be provided at container recovery. Also, ensure `cd_param_file` contains all passwords.
Error encountered in connecting to an installed Connect:Direct container node.	Connect:Direct API port is not exposed to the host.	The Connect:Direct API port (default 1363) running inside the container should be mapped to an available host port.
Error encountered in performing a file transfer with other Connect:Direct nodes.	Connect:Direct Server port is not exposed to host or its entry is missing inside the `netmap.cfg` file of the partner node.	• The server port (default 1364) of the Connect:Direct node running inside the container should be mapped to the available host port. • Define exposed port in partner node's `netmap.cfg` file for a successful file transfer.

Table 4. Troubleshooting deployment and functional issues (continued)

Issue	Possible Reason	Solution/Workaround
Error encountered in file transfer from pod to Connect:Direct windows node.	Netmap check on Connect:Direct windows node is enabled and not allowing file transfer.	The IPs of all the worker nodes should be mentioned in the netmap of Connect:Direct windows node using the alternate.comminfo parameter, like below: `Alternate Comminfo : <worker node1 ip>, <worker node2 ip>`
Unable to add certificates for Secure+ using Connect:Direct Web services	This functionality is not present in the current version of Connect:Direct Web services and will be available in the upcoming release.	Import the certificates using Secure+ CLI by attaching to the container.
In case of migration of Connect:Direct to container environment, local users in userfile.cfg not available inside the container	The local users defined in userfile.cfg are not present inside the container environment.	When migrating to container environment from a conventional Connect:Direct environment, the local users defined in `userfile.cfg` should be added inside the container using the `useradd` command.
In case of OpenShift cluster, you get below error while executing helm commands: **Error**: Kubernetes cluster unreachable: the server has asked for the client to provide credentials.	User is logged on the cluster	Login to the cluster `oc login -u <username> -p <password>`

Troubleshooting your IBM certified Containerized environment

This section describes how to troubleshoot in your containerized environment.

The troubleshooting steps assumes that `Kubectl` command line tool is installed and configured in your environment.

When you encounter an issue in your containerized environment, run the following checks to assist you with troubleshooting process.

Note:

You may need to engage your OCP admin and/or solutions team for platform or performance related issues.For OCP cluster issues not related to the IBM Sterling Connect:Direct for UNIX product, contact (RedHat Support).

Generic Checks

- Get nodes information and check the status of nodes is Ready

```
kubectl get nodes -n <namespace> -o wide
```

- Review the Persistent Volume and Persistent Volume Claim information and ensure they match with your deployment YAML files

  ```
  kubectl describe pv <pv name> -n <namespace>
  ```

- Get the status of the IBM Connect:Direct for Unix containers deployment job:

  ```
  kubectl -n <namespace> rollout status deployment/<deployment name>
  kubectl describe deployment <deployment name> -n <namespace>
  ```

- Get information about the pods after the IBM Connect:Direct for Unix containers have been deployed:

  ```
  kubectl get pods -n <namespace> -o wide
  ```

- Retrieve the logs for a particular pod

  ```
  kubectl logs -f <pod name> -c <deployment name> -n <namespace>
  ```

- Get details about a particular pod

  ```
  kubectl describe pods <pod name> -n <namespace>
  ```

Rerun a failed container deployment job

When a container deployment fails investigate the cause and follow the procedure given below over command line:

- Delete the failed deployment using the below command

  ```
  helm delete --purge <release-name>
  ```

- Delete the persistent volume

  ```
  kubectl delete pv <pv-name>
  ```

- Delete the secret

  ```
  kubectl delete secret <secret-name>
  ```

- Delete the configMap

  ```
   kubectl delete configmap <configmap name>
  ```

- Manually delete the data present in persistent volume
- Re-run the deployment steps described in "Installing IBM Connect:Direct for UNIX using an IBM Certified Container Software" on page 39.

View Container logs

- Check the container logs using the command

  ```
  kubectl logs -f <pod name> -c <deployment name> -n <namespace>
  ```

- From the mapped path on Storage Volume, check the silent installation logs `cdaiLog.txt` inside the `work` directory.
- IBM Connect:Direct for UNIX traces for PMGR, SMGR or CMGR can be enabled using the Connect:Direct Web Services. From the mapped path on Storage Volume, check the traces generated inside the `work` directory.
- IBM Connect:Direct for UNIX stats can be checked in the `work` directory inside the mapped path on Storage Volume.

Setting Up Connect:Direct for UNIX Manual Pages

The UNIX operating system organizes all Help into manual (man) pages.

1. For syntax of a UNIX command, type the following where *command* is the UNIX command:

   ```
   % man command
   ```

   Most UNIX systems store online manual pages in /usr/man/man1. IBM Connect:Direct stores its manual pages in d_dir/ndm/man1,where *d_dir* is the IBM Connect:Direct installation directory.
2. Type the following command to copy the IBM Connect:Direct manual pages into the UNIX manual pages directory:

   ```
   % cp d_dir/ndm/man1/*.1 /usr/man/man1
   ```

   You must have write privileges to the directory /usr/man/man1 to perform this command.

   You can also use symbolic links instead of copying the files. Refer to UNIX manual pages.
3. Type the following command to access IBM Connect:Direct manual pages that you combined with UNIX manual pages, where *command* can be cdpmgr, ndmxlt, or ndmmsg:

   ```
   % man command
   ```

Accessing IBM Connect:Direct Manual Pages

To access the IBM Connect:Direct manual pages.

Type the following command to access IBM Connect:Direct manual pages on IBM pSeries, or Sun Sparc running the Solaris operating system, if the system is using the BSD version of the man command (/usr/ucb/man). The command can be cdpmgr, ndmxlt, or ndmmsg.

```
% man -M d_dir/ndm command
```

Virtualization support

IBM cannot maintain all possible combinations of virtualized platforms. However, IBM generally supports all enterprise class virtualization mechanisms, such as VMware ESX, VMware ESXi, VMware vSphere, Citrix Xen Hypervisor, KVM (Kernel-based virtual machine), and Microsoft Hyper-V Server.

IBM investigates and troubleshoots a problem until it is determined that the problem is due to virtualization. The following guidelines apply:

- If a specific issue is happening because the system is virtualized and the problem cannot be reproduced on the non-virtualized environment, you can demonstrate the issue in a live meeting session. IBM can also require that further troubleshooting is done jointly on your test environment, as there is not all types and versions of VM software installed in-house.
- If the issue is not able to be reproduced in-house on a non-virtualized environment, and troubleshooting together on your environment indicates that the issue is with the VM software itself, you can open a support ticket with the VM software provider. IBM is happy to meet with the provider and you to share any information, which would help the provider further troubleshoot the issue on your behalf.
- If you chose to use virtualization, you must balance the virtualization benefits against its performance impacts. IBM does not provide advice that regards configuring, administering, or tuning virtualization platforms.

IBM Control Center Director Support

Control Center Director installs, upgrades and applies maintenance to Connect:Direct through a Connect:Direct Agent instance.

After you have upgraded Connect:Direct for UNIX, to the required maintenance level complete the following procedures to ensure Connect:Direct for UNIX servers are discovered dynamically by Control Center Director.

Control Center Dir uses Certificate-based authentication to authenticate itself to a Connect:Direct® server. For more information on how to configure Connect:Direct and Control Center Director for Certificate-Based Authentication see the following sections:

- Enable Certificate-based authentication on Control Center Director
- Enable Client Authentication on the Connect:Direct Secure Plus
- "Local User Information Record Format" on page 124

Known Restriction

Upgrade on target Connect:Direct nodes via ICC Director Web Console is currently not supported for Connect:Direct servers running on a 32-bit Solaris system. You must upgrade to Connect:Direct for UNIX v6.0.0 and above to support the upgrade process.

Configuring Connect:Direct for UNIX for Server and Upgrade Management

Control Center Director upgrades and applies maintenance to Connect:Direct through a Connect:Direct Agent instance. Agent is included with the Connect:Direct software when it is at a required level of maintenance for Agent inclusion.

To successfully move to a Connect:Direct version that supports a Control Center Director deployment, there are a few scenarios to consider. Review the actions below to optimize your update experience:

- Download the required maintenance version of Connect:Direct software. For more information go to Fix Central.
- Certificate-based authentication is enabled when you install and configure Connect:Direct Secure Plus for UNIX.

1. Configure the Agent listening port that Control Center Director will use to communicate with the Agent.

   ```
   agent.port=<port> [Default:1365]
   ```

   The Agent is now set to automatically listen for incoming connections from Control Center Director.

   Attention: With multiple Connect:Direct instances on the same system you're likely to run into port conflict issues unless you allocate a unique Agent listening port per instance.

   It is also recommended that having upgraded an instance, its unique port number must be applied before upgrading the next instance. This prevents potential errors that you could encounter during an upgrade process due to port conflict.

2. Configure the Control Center Director Open Server Architecture (OSA) URL, the target location where Agent posts all the events to Control Center Director.

   ```
   osa.rest.url=https;//<ip/hostname;port>/osa/events/post:
   [Default:<blank>]
   ```

   Note: Ensure that you insert a '**;**' and not a '**:**' between hostname and port and after https.

3. Set the following property to enable Agent to post all events to Control Center Director .

```
osa.disable=N
```

4. Invoke the following script for changes to take effect. This script is available `atcdinstallation_location/install/agent/bin`.

```
startAgent.sh
```

Configuring Connect:Direct for UNIX for License Governance

Set the following parameters (initparms) to automate license metrics collection from Connect:Direct for UNIX.

Table 5. Initialization Parameters

Parameter (initparm)	Possible Values
`license.edition`	• Premium • Standard • Solo • Default: Blank (undefined)
`license.type`	• Production • Non-Production • Default: Non-Production
`license.pvu`	A non-negative integer • The `license.pvu` parameter is only applicable for Connect:Direct Premium licenses • This value can be calculated using the IBM License Metric Tool (ILMT) or it can be looked up at the IBM Processor Value Unit licensing website. • Default: 0

Note: All three Initparms are required, but can be unset and a user does not have to supply a value.

Solo license edition type constraints:

- A warning message is logged if the number of Netmap entries in `netmap.cfg` exceeds 2.
- A warning message is logged when a transfer is initiated with third or later remote entry, in order of appearance.
- The number of concurrent sessions is restricted to 2 or fewer

Configuring Connect:Direct for Unix for New Install Task

You can initiate fresh installation of Connect:Direct servers from Control Center Director. To automate new installation of Connect:Direct server for UNIX from Control Center Director, set the following parameters (initparms) in the `install.agent` record of the `initparm.cfg` file:

Table 6. Initialization Parameters

Parameter (initparm)	Definition	Possible Values
agent.enable	Use to enable/disable the agent	• y (Default) • n
osa.rest.url	Use to connect Connect:Direct Agent with Control Center Director.	• blank (Default) • URL
osa.disable	Allows disabling OSA without deleting osa.rest.url	• blank (Default) • y • n
agent.installation_id	Identifies the Connect:Direct installation package installed by Control Center Director	Informational only, managed by CCD.

For more information on how to install new Connect: Direct server for UNIX from Control Center Dir , see Installing new Connect:Direct server for UNIX.

Configuring Connect:Direct for Unix for Custom Locations of Connect:Direct Backups and Installation Programs

Use the following parameters to set the Connect:Direct, Agent backup and installer locations to be used before an upgrade.

Table 7. Initialization Parameters

Parameter (initparm)	Definition	Possible Values
cd.backup	Set this parameter in `backup.locations` initparm record to specify the location where backup of Connect:Direct will be stored before an upgrade.	• Any path on the system. **Note:** The path should be an absolute path, not same as installation directory. It should not be a sub directory of <installation directory> or agent.backup path.
agent.backup	Set this parameter in `backup.locations` initparm record to specify the location where backup of Install Agent will be stored before an upgrade.	• Any path on the system. **Note:** The path should be an absolute path, not same as installation directory or cd.backup path. It should not be a sub directory of <installation directory> or cd.backup path.
emergency_restore.installers	Set this parameter in `backup.locations` initparm record to specify the location where installer will be stored, which can be used during emergency restore.	• Any path on the system. **Note:** The path should be an absolute path.

Table 7. Initialization Parameters (continued)

Parameter (initparm)	**Definition**	**Possible Values**
installer.location	Set this parameter in `install.agent` initparm record to specify the location of installer download during upgrade via Control Center Director.	• Any path on the system. **Note:** The path should be an absolute path.

Malware or Virus Scanning Guidelines

This section describes the guidelines for scanning IBM Connect:Direct for UNIX data for viruses, malware, etc.

Malware or Virus Scanning Guidelines

To ensure that Connect:Direct for UNIX file system is free of viruses or malware, the user is responsible for scanning all data transferred by Connect:Direct. Source files need to be scanned prior to sending. Received files must be scanned by SSP or immediately upon receipt by Connect:Direct before being transferred or processed.

The anti-virus or malware scanner can be integrated into the user environment in two ways :

1. **In-house scanner –** The user of Connect:Direct for UNIX uses their own anti-virus or malware scanner before any file is processed by Connect:Direct for UNIX either to receive it or to send it. It is the responsibility of the user to ensure the virus or malware free file system of the Connect:Direct for UNIX.
2. **ICAP Anti-Virus Scanning by IBM Sterling Secure Proxy –** The user of Connect:Direct for UNIX can leverage IBM Sterling Secure Proxy if they have it in their environment. It provides a mechanism to integrate ICAP anti-virus. For more information, refer to https://www.ibm.com/docs/en/secure-proxy/6.0.3?topic=scenarios-icap-anti-virus-scanning.

Chapter 3. Administration Guide

Use the Administration Guide to maintain configuration files, initialization parameters, client configuration files, network map files, access information files, and client and server authentication key files, along with specifying connection information and using IBM Connect:Direct in test mode.

Password Storage

Connect:Direct for UNIX enables you to use any of the following for password storage:

- /etc/passwd file
- /etc/shadow file when supported by the operating system
- HP-UX trusted security
- Network Information Service (NIS), formerly known as Yellow Pages
- Digital UNIX Enhanced Security
- Pluggable Authentication Modules (PAM)

Maintaining configuration files

Configuration files define the operating environment for IBM 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: userfile.cfg and sysacl.cfg

Note: The content of these configuration files is unencrypted. However, passwords are an exception. When Secure Password Encryption (SPE) is enabled, passwords are encrypted.

After the initial customization, you can modify these files, if necessary.

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 sample format below, 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.

Sample format of a configuration file

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

Line	Contents	Notes
1	`#Miscellaneous Parameters`	# indicates a comment
2	`ndm.path:path=/ndm/users/c:`	record name=**`ndm.path`**, parameter=**`path`**, value= `/ndm/users/c`
3	`proc.prio:default=8:`	record name=**`proc.prio`**, parameter=**`default`**, value= 8
6	`#Local IBM Connect:Direct connection information`	# indicates a comment
7	`local.node:\`	record name=**`local.node`**
13	.	
...	.	
21	.	
22	`:tcp.api=rusty;3191:\`	parameter= **`tcp.api`**, value= `rusty;3191`
23	`:tcp.api.bufsize=32768:`	parameter= **`tcp.api.bufsize`**, 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 IBM 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 IBM 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
```

Maintaining the initialization parameters file

Initialization parameters determine various IBM 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 IBM Connect:Direct initialization parameters file using any text editor. Before changing a value in the file, first shut down the IBM 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.

You can also use the Connect:Direct Browser User Interface to perform some of the procedures related to the initialization parameters file. To learn more about the Connect:Direct Browser User Interface, see the documentation related to that product in the IBM Documentation Library. 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.

Contents of 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:

- Path information— The **ndm.path** record contains the location of Connect:Direct for UNIX and the shared work area for SNODE work files.
- Node information— The **ndm.node** record stores the name of the Connect:Direct for UNIX node.
- PAM authentication— The **ndm.pam** record identifies the PAM service configuration file used to authenticate users.
- Process priority information— The **proc.prio** record identifies the default process priority.
- Restrict record — The **restrict** record specifies if some characters are restricted in run task/run job commands when a user's run directory is restricted.
- 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.
- AIX zFBA option— The **zFBA** parameter enhanced performance of large file transfers between z/OS and AIX.
- Secure cdpmgr initialization— Provides the option to prevent **cdpmgr** from sanitizing inherited environment variables.
- License information— The **license** record identifies Connect:Direct license metrics.
- Install Agent record— The **install.agent** record contains parameters to control the Connect:Direct Install Agent.
- Backup locations record— The **backup.locations** record specifies custom Connect:Direct backup paths and installation program paths.
- File Agent parameters— The **cd.file.agent** record contains parameters to enable/disable file agent.
- Port Check parameters— The **port.check** record specifies the list of ignored port check connections.

Sample initialization parameters file

The following example shows how some of these parameters are specified:

```
# Miscellaneous Parameters
ndm.path:path=/sci/users/mscarbro/cd4000:\
        :snode.work.path=/sci/users/mscarbro/cd4000/shared:

ndm.node:name=mws_joshua_4000:
ndm.pam:service=cdlogin:
ndm.quiesce:quiesce.resume=n:

proc.prio:default=10:
restrict:cmd=y:

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

# Global copy parameters.
copy.parms:\
 :ckpt.interval=2M:\
 :ulimit=N:\
 :xlate.dir=/sci/users/mscarbro/cd4000/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:

# Authenticator
authentication:\
 :server.program=/sci/users/mscarbro/cd4000/ndm/bin/ndmauths:\
 :server.keyfile=/sci/users/mscarbro/cd4000/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=rt.sles96440:\
 :comm.info=0.0.0.0;9974:\
 :comm.transport=tcp:

# Secure+ parameters
secure+:\
 :certificate.directory=/home/nis02/jlyon/certs: \
 :s+cmd.enforce.secure.connection=n:
```

Updating records

You can update various parameters in records that IBM Connect:Direct uses. Required parameters are displayed in bold.

Path record

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

Parameter	**Description**	**Value**	**Restart Required**
`path`	The path to all IBM Connect:Direct subdirectories and files.	path specification	Not Applicable **Note:** Not recommended to be changed. If changed, restart is required.

SNODE work path parameter

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. This path must point to a cluster file system, such as IBM's GPFS, or a Network File System version 4 (NFSv4) or greater resource. This includes the Amazon Elastic File System (EFS), as it is mounted via NFSv4 protocol. NFSv3 is not supported. 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**	**Restart Required**
snode.work.path	The path to the shared work area for SNODE work files. **Note:** Specify the same path for all nodes in a cluster.	path specification	No

Process Library location parameter

The default value of this parameter is `<installation directory>/process`, which will be created during installation with the appropriate permissions. A corresponding initparm with the name 'process.dir' is added pointing to this path.

In case the Connect:Direct Admin chooses an alternative location for the process library, this directory must have UNIX permission 1775. Also, the parent directory of the new location should have permissions such that Connect:Direct admin can perform read/write on the new 'process library'. The initparm 'process.dir' pointing to the default process library should also be updated to point to the new location.

The `process.dir` parameter is part of the ndm.path record and identifies the path to the Connect:Direct process library but it is configurable and can be changed. The following table describes the `process.dir` parameter:

Parameter	**Description**	**Value**	**Restart Required**
`process.dir`	The location of process library used by Integrated File Agent	path specification	No

Node name record

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

Parameter	Description	Value	Restart Required
`name`	The name of the node.	The maximum length is 16 bytes. If a node name is longer, the name will be truncated.	Not Applicable **Note:** Not recommended to be changed. If changed, restart is required.
`instance.id`	A unique identifier for Connect:Direct nodes in a load balanced (LB) cluster.	A unique text identifier for each node in an LB cluster. By default, it is initialized with a universally unique identifier (UUID).	Not Applicable. **Note:** Changing this setting is not recommended. If changed, a restart is required.

PAM service record

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

The service name required is typically defined in the `/etc/pam.conf` file for AIX, Solaris and HP operating systems, or defined and named by a file in the `/etc/pam.d` directory for Linux operating systems. Your system might also have a man page for PAM that provides further details.

The following table describes the parameter available for this record:

Parameter	Description	Value	Restart Required
`service`	PAM service configuration file name.	File name	No

Quiesce/resume record

The **ndm.quiesce** record specifies whether IBM Connect:Direct is operating in a “test” mode.

Use this record in conjunction with the NDMPXTBL table to enable the test mode. If you enable the **quiesce.resume** parameter, you must have an NDMPXTBL parameter table updated for your environment in the installation ndm/cfg/*<nodename>* directory. For more information on the test mode and the NDMPXTBL table, see Processing Flow of the Test Mode.

The following table describes the parameter available for this record:

Parameter	Description	Value	Restart Required
`quiesce.resume`	Enables/disables the test mode for IBM Connect:Direct.	y | n y—Enables the test mode. n—Disables the test mode. The default is n.	Yes

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	Restart Required
`default`	The default value of the Process priority.	1–15. The default value is 10. 15 is the highest priority.	No

Restrict record

If a run 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 Local User Information Record Format and Remote User Information Record. The following parameter is available for this record:

Parameter	Description	Value	Restart Required
`cmd`	Determines if commands with certain special characters are allowed.	y | n y—Restricts the ability to use commands with any of the following special characters: ; & ‘ | n—Does not restrict allowed commands.	No

Remote node connection record

The **rnode.listen** record contains parameters used by the local node to monitor inbound connection requests.

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	Restart Required
`recid`	A unique identifier of an **rnode.listen** record.	A text string	Yes

Parameter	Description	Values	Restart Required
`comm.info`	The information needed to monitor connection requests from remote nodes using TCP/IP. This parameter is required. • For TCP/IP connections, specify the host name or the IP address and port number. If specifying an IP address and port, separate parameters with a semicolon (;). Separate multiple addresses/host names with a comma, for example: 10.23.107.5;1364, fe00:0:0:2014::7;1364, msdallas-dt;1364 For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.	For TCP/IP connections, specify the host name or the IP address and port number: 10.23.107.5;1364 Separate multiple IP/host addresses with a comma (,): fe00:0:0:2014::7;1364, msdallas-dt;1364 A space can be added after the comma for readability. Set the IP address to monitor a specific adapter or to 0.0.0.0, to monitor all adapters. The default port is 1364.	Yes
`comm.transport`	The transport protocol for the remote node.	tcp tcp—For TCP/IP connections	Yes

Transmission Control Queue record

The **tcq** record provides information that pertains to the Transmission Control Queue (TCQ).

The following parameters are available for this record:

Parameter	Description	Value	Restart Required
max.age	The maximum number of days a Process with Held-in-Error status remains in the TCQ before it is automatically deleted.	A three-digit decimal number. IBM Connect:Direct does not automatically delete Processes when max.age=0. The default is 8 days.	Yes
ckpt.max.age	The maximum number of days a Process' checkpoint file is stored unused before it is automatically deleted.	An integer in the range 0 - 2147483647, inclusive. IBM Connect:Direct does not automatically delete the checkpoint file when ckpt.max.age= 0. The default is 8 days after a fresh installation. If ckpt.max.age is left unset, it defaults to 2147483647.	Yes

Connect:Direct Secure Plus record

The Connect:Direct Secure Plus record (Secure+ record) provides information pertaining to remote configuration of Connect:Direct Secure Plus from the IBM Connect:Direct client API.

This record is not included in the **initparm.cfg** file by default. You must manually add the Secure+ record to the initparm.cfg file. The following parameters are available for this record:

Parameter	Description	Value	Restart Required
`certificate.directory`	Specifies a default certificate directory for Secure+ commands issued from the IBM Connect:Direct client API. If the certificate directory is not configured, the default directory created during installation is used.	Directory path name	No
`s+cmd.enforce.secure.connection`	Specifies whether Secure+ commands are accepted from the IBM Connect:Direct client API on unsecure connections.	y | n y—Commands from unsecure connections are not accepted. The default is y. n—Commands from unsecure connections are accepted	No

Global Copy record

The Global Copy record called **copy.parms** provides default information for the IBM 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	Restart Required
`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: no—No checkpointing nnnnnnnn—Up to an 8-digit decimal nnnnnnnnK—Up to an 8-digit decimal, where K denotes 1024 bytes nnnnnnnnM—Up to an 7-digit decimal, where M denotes 1048576 bytes nnnnG—Up to an 4-digit decimal, where G denotes 1073741824 bytes	The maximum possible value is 1 terabyte (TB). The normal value is 64KB.	No
`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. y—Recognizes the user file size limit. If this limit is exceeded during a copy operation, the operation fails.	No

Record	Description	Value	Restart Required
`xlate.dir`	The name of the directory containing the translation tables.	Any valid directory. The default path is *d_dir*/ndm/xlate.	No
`xlate.send`	The default translation table used when sending data to a remote node.	Any valid directory. The default file name is def_send.xlt.	No
`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.	No
`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. n—Places a Process in the Hold queue with a value of HE. The default is n.	No
`ecz.compression.level`	Sets the compression level.	1–9. The default is 1. 1—The fastest but offers the least degree of compression. 9—Provides the greatest degree of compression but is the slowest.	No
`ecz.memory.level`	How much virtual memory to allocate to maintaining the internal compression state.	1–9. The default is 4. 1—Uses the least memory and 9 uses the most memory.	No
`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.	No

Record	Description	Value	Restart Required
`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. 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. 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. You can perform retry attempts based on codes only, IDs only, or a combination of the two. When a retry condition is detected, the session is terminated cleanly and the Process is placed in the Timer queue. **Note:** If you are using the file allocation retry function when communicating with a remote node on an operating system that is not UNIX, identify operating system retry codes using formats and code values defined by the remote node.	Any valid error code	No
`retry.msgids`	Identifies the message IDs to use to support a file allocation retry attempt. 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. You can perform retry attempts based on codes only, message IDs only, or a combination of the two. 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.	No

Record	Description	Value	Restart Required
`tcp.crc`	Globally turn on or off the CRC function for TCP/IP Processes.	y | n y—Turns on the CRC function globally. n—Turns off the CRC function globally. The default is n.	No
`tcp.crc.override`	Determines whether netmap remote 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 y—Allows netmap remote node and Process statement overrides for CRC checking. n—Prevents netmap remote node and Process statement overrides for CRC checking. The default is n.	No
`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 y—Strips blanks from the end of a record n—Does not strip blanks from the end of a record i—Setting for strip.blanks is determined by the default value of the remote node type as follows: • z/OS, VM, and i5OS—y • All other platforms—n	No
`insert.newline`	Arbitrarily appends an LF character at the end of each record when receiving a datatype=text file. By default, an LF character is not appended if one already exists at the end of a record.	y | n y—Arbitrarily appends an LF character n—Appends an LF character if needed	No
`recv.file.open.perm`	`recv.file.open.perm`=nnn, where nnn is an octal integer describing the desired default permissions for new files received. It is the same as the value documented for the copy `sysopt permiss`.	Octal Integer	No

Record	Description	Value	Restart Required
`recv.file.open.ovrd`	`recv.file.open.ovrd=x`, where x is one of the following three values: • Y: Allow copy step sysopt `permiss` value to override `recv.file.open.perm` value when receiving a new file. This is the default. • N: Disallow copy step sysopt `permiss` value to override `recv.file.open.perm` value when receiving a new file. • P: Allow copy step sysopt `permiss` value to override `recv.file.open.perm` value when pnode is receiving a new file.	y | n | p	No
`fsync.after.receive`	Files received and closed by C:D to an NFS destination may not be immediately ready for processing due to NFS cached writes. This parameter optionally calls fsync function to attempt to flush all data to disk before closing the file.	[y|n]. Default is y.	No

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, IBM Connect:Direct attempts to synchronize the previous run task step on the SNODE with the current run task step. If synchronization fails, IBM 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 Required
restart	If processing is interrupted when a run task on an SNODE step is executing and if synchronization fails after a restart, IBM 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. **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 y—The run task program runs again. The default is **y**. n—The Process skips the run task step.	No

Statistics file information record

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

Parameter	Description	Value	Restart Required
`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, nnnnnnnM, or nnnnG—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.	No
`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 | <u>n</u> y—Commands are written to the statistics file. n—Commands are not written to the statistics file. The default is n.	No
`log.select`	Specifies whether IBM Connect:Direct creates a statistics record when a select process or select statistics command is executed.	y | <u>n</u> y—A statistics record is created. n—A statistics record is not created. The default is n.	No

Parameter	Description	Value	Restart Required
`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.	Yes

Running a Process generates multiple statistics records. To accommodate the large number of statistics records generated, IBM 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.

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	Restart Required
`server.program`	The name and location of the server program used during the authentication procedure.	The default is ndmauths.	No
`server.keyfile`	The name and location of the key file used during the authentication procedure.	The default is keys.server.	No

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	Restart Required
`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.	No

Parameter	Description	Value	Restart Required
`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. 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.	No
`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, such as 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.	No

Parameter	Description	Value	Restart Required
`security.exit.flag`	Modifies the default behavior of **security.exit.program**. This is an optional parameter.	snode_sec_exit_only | sec_exit_only snode_sec_exit_only—Causes IBM Connect:Direct to use the security exit, when it is acting in the role of the SNODE. After IBM 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 IBM Connect:Direct is the PNODE. sec_exit_only—Causes IBM Connect:Direct to always use the security exit. After IBM Connect:Direct receives a valid message, it evaluates the proxy and the secure point-of-entry to establish the local user.	No

zFBA for large file transfer

The zFBA parameter has functionality that provides enhanced performance of large file transfers between z/OS and AIX; CPU utilization on the z/OS node is greatly reduced and the data throughput is increased by utilizing parallel data paths to the device both from z/OS as well as the AIX system.

The parameter is as follows:

```
# zFBA parameter
zFBA:\
 :on=y:
```

Note: When zFBA is set to on, CRC checking is disabled for sessions using the DS8K device for data transfers. If Secure+ is configured between two nodes, the transfers will use the standard TCP/IP protocol and ignore the zFBA option.

Secure cdpmgr initialization procedure

Secure cdpmgr initialization is used to sanitize inherited environment variables to prevent run task steps from depending on one or more of the inherited environment variables from working properly.

Parameter	Description	Value	Restart Required
`ndm.env_vars:sanitize`	Implementing standard safe initialization procedures, cdpmgr sanitizes environment variables inherited from the user that starts it. This sanitization procedure may prevent some run task steps from working properly if the tasks were designed to rely on these inherited variables. While IBM recommends designing run tasks so that they don't rely on inherited environment variables, this parameter provides the option to prevent cdpmgr from sanitizing inherited environment variables.	[y|n]. Default is y.	Yes

License record

The **license** record identifies Connect:Direct license metrics.

Following table lists the parameters and their possible values:

Parameter	Possible Values	Restart Required
`license.edition`	• Premium • Standard • Solo • Default: Blank (undefined) • Container	Yes
`license.type`	• Production • Non-Production • Default: Non-Production **Note:** For container, the default value is Production	Yes

Parameter	Possible Values	Restart Required
`license.pvu`	A non-negative integer • The `license.pvu` parameter is only applicable for non-container Connect:Direct Premium licenses • This value can be calculated using the IBM License Metric Tool (ILMT) or it can be looked up at the IBM Processor Value Unit licensing website. • Default: 0	Yes
`license.vpc` ⚠ **Attention:** The `license.vpc` parameter is only applicable for Connect:Direct **Container Premium** licenses.	A non-negative integer • This value can be calculated using the IBM License Metric Tool (ILMT) or or for more information about how to determine the usage of VPCs, see Virtual Processor Core (VPC) • Default: 1	Yes

install.agent record

The **install.agent** record contains IBM Sterling Connect:Direct parameters to control the Connect:Direct Install Agent, used by Control Center Director to manage applying upgrades and maintenance.

Parameter	Description	Value	Restart Required
`agent.port`	Configures the Agent listening port that Control Center Director will use to communicate with the Agent.	Port number Default Value: 1365	Yes

Parameter	Description	Value	Restart Required
`osa.rest.url`	Configures the Control Center Director Open Server Architecture (OSA) URL, the target location where Agent posts all the events to Control Center Director.	Valid URL **Example:** `osa.rest.url=https://<ip/hostname:port>/osa/events/post:[Default:<blank>]` **Note:** Ensure that you insert a **';'** and not a '**:**' between hostname and port and after https. Default Value: blank	No
`osa.disable`	Allows disabling OSA without deleting osa.rest.url.	Y | N	No
`agent.enable`	Specifies whether the agent is enabled to run, or disabled and will be stopped.	Y | N	No
`agent.installation_id`	Identifies the Connect:Direct installation package installed by Control Center Director.	Informational only, managed by CCD.	No
`installer.location`	Specifies the location of installer download during upgrade via Control Center Director.	Valid path on system **Note:** The path should be an absolute path.	No

Backup locations record

The **backup.locations** record contains IBM Sterling Connect:Direct parameters to specify custom backup paths and installation program paths, used by Control Center Director during upgrade.

Parameter	Description	Value	Restart Required
`cd.backup`	Specifies the location where backup of Connect:Direct will be stored before an upgrade.	path	No
`agent.backup`	Specifies the location where backup of Install Agent will be stored before an upgrade.	path	No
`emergency_restore.installers`	Specifies the location where Connect:Direct installer will be stored, which can be used during emergency restore.	path	No

Parameter	Description	Value	Restart Required
installer.location	Specifies the location of installer download during upgrade via Control Center Director.	path	No

Parameter	Description	Value

cd.file.agent Record

The **cd.file.agent** record contains IBM Connect:Direct parameters to control Connect:Direct File Agent.

The following table describes the parameter available for this record:

Parameter	Description	Value	Restart Required
`cdfa.enable`	Specifies whether the Integrated File Agent is enabled to run or disabled and will be stopped.	y|n	No

port.check record

The **port.check** record contains IPv4/IPv6 address or hostname from which port check connections are expected and allowed.

The following table describes the parameter available for this record:

Parameter	Description	Value	Restart Required
trusted.addr	Specifies the IPv4/IPv6 address or hostname from which port check connections are expected and allowed.	• list of valid IP addresses or hostnames **separated with commas** • blank (default)	Yes

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 IBM Connect:Direct sessions. These ports also need to be open on the firewall of the trading partner to allow the inbound IBM Connect:Direct sessions. This feature enables controlled access to an IBM Connect:Direct server if it is behind a packet-filtering firewall without compromising security policies.

Before you configure firewalls, review all information regarding firewall navigation and rules beginning with Firewall Navigation.

The following parameters are available for this record:

Parameter	Description	Value
tcp.src.ports	For TCP/IP connections, remote IP addresses and the ports permitted for the addresses when using a packet-filtering firewall. This parameter is required only if the local node acts as a PNODE. Place all values for an address inside parentheses and separate each value for an address with a comma.	Valid IP address with an optional mask for the upper boundary of the IP address range and the associated outgoing port number or range of port numbers for the specified IP address, for example: (199.2.4.*, 1000), (fd00:0:0:2015:*::*, 2000-3000), (199.2.4.0/255.255.255.0, 4000-5000),(fd00:0:0:2015::0/48, 6000, 7000) A wildcard character (*) is supported to define an IP address pattern. If the wildcard character is used, the optional mask is not valid. For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.
tcp.src.ports.list. iterations	The number of times that IBM 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 2.

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:

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

You can modify IBM Connect:Direct configuration files using any text editor. If you want to create a new configuration file, use the cdcust command.

Contents of 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 IBM Connect:Direct is installed.

Sample client configuration file

The following example shows a sample client configuration file:

```
# 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:
```

API configuration record

The IBM 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. For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.
tcp.port	The TCP/IP port number to which the API usually connects.	Port 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.

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 ndmstat and ndmproc.	Directory name. The default directory is ndm/bin/.
prompt.string	Identifies the CLI prompt to display on the command line when the client is started. If the prompt string includes spaces or special characters, enclose it in single or double quotation marks. 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. 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".

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. The default is ndmauthc.
client.keyfile	The key file to use during authentication.	Client key file. The default is keys.client.

Maintaining the network map file

The network map file is created when you install IBM Connect:Direct. 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. For local node connection record (`local.node` record) updates to be effective immediately, the Connect:Direct server must be recycled.

You can also use the Connect:Direct Browser User Interface to perform some of the procedures related to the initialization parameters file. To learn more about the Connect:Direct Browser User Interface, see the documentation related to that product in the IBM Documentation Library.

IBM Connect:Direct Web Services's Partner Card feature simplifies configuring a connection with another Connect:Direct node. For more information, refer to the below links:

- IBM Connect:Direct Web Services Adding Partner Manually From Connect:Direct Windows Node
- IBM Connect:Direct Web Services Accessing Partners
- IBM Connect:Direct Web Services Editing Partner
- IBM Connect:Direct Web Services Importing Partners
- IBM Connect:Direct Web Services Adding Partner Manually From Connect:Direct Unix Node
- IBM Connect:Direct Web Services Adding Partner Manually From Connect:Direct ZOS Node

Contents of 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 IBM Connect:Direct is installed and *cd_node* is the node name.

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.

Sample remote node records in a network map

The following sample shows network map remote node entries for a TCP/IP connection to remote nodes. To insert comments about fields in the network map, be sure to place a # in the first column. If the # is not in the first column, the comment is not ignored and the field is read.

```
# Sample Network Map remote node entry for a TCP/IP connection
 remote.customer.node:\
   :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=180;\
  :runstep.max.time.to.wait=0:\
   :contact.name=:\
   :contact.phone=:\
   :descrip=:\
   :sess.total=255:\
   :sess.pnode.max=255:\
   :sess.snode.max=255:\
   :sess.default=1:\
   :comm.info=10.20.246.49;9974:\
   :comm.transport=tcp:\
   :comm.bufsize=65536:\
   :pacing.send.delay=0:\
   :pacing.send.count=0:
```

Local node connection record

The **`local.node`** record serves two separate purposes:

- Configures settings for the local node
- Provides default configuration values that can be overridden in the remote node entries.

Two sets of connection retry parameters are created:

- Short-term parameters define retry attempts in the event of a short-term connection failure.
- Long-term parameters are used 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**
api.max.connects	The maximum number of concurrent API connections permitted for the local node. The value of **`api.max.connects`** and **`sess.total`** cannot exceed the number of file descriptors available. This value is system dependent. 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.
comm.bufsize	The buffer size for transmitting data to and from a remote node for TCP/IP connections.	The value for TCP/IP has no limit (up to 2,147,483,623). The default is 65536 bytes.

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.	`00.00.00–23.59.59` 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` | `delete` `hold`—Places Processes in the hold queue in "Held in Error" status, after all retry attempts are used. This is the default value. `delete`—Causes the Processes to be deleted from the TCQ.
contact.name	The name of the Connect:Direct administrator or operator.	Name
contact.phone	The phone number of the Connect:Direct administrator or operator.	Phone number
descrip	Comments to include as part of the record.	An unlimited text string
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 an 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 all connections, the remote node name must be in the netmap.cfg.	y | n y—Specifies that the security checks are made to verify that the remote node name is in the netmap.cfg file. Also, checks the network map for all nodes that Connect:Direct will communicate with to validate the node name and the IP address. `l`—Checks the network map only for nodes that the local Connect:Direct will initiate sessions with. `r`—Checks the network map only for remote nodes that will communicate with this node. n—Will not validate any session establishment requests in the network map. The default value is n.

Parameter	Description	Value
outgoing.address	If running in a high availability environment, this parameter enables you to specify the virtual 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.	*nnn.nnn.nnn.nnn* (IPv4) or *nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn* (IPv6) For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.
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.	The format is *nnn*. No limit exists for the size of this value. 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.	No limit exists for the size of this value. The default is `0`, which indicates no pacing of this type.
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 | n y—Specifies that the remote users can specify a dummy user ID in **`snodeid`** parameter. n—Specifies that the remote users cannot specify dummy user ID in **`snodeid`** parameter. The default is n.
	The following code illustrates the logic used to perform a security check for the user ID:	

```
if (snodeid is coded with ID and PSWD)
        attempt OS validation
        if (OS validation succeeds)
                security OK
        else if (proxy.attempt=yes)
                if (ID@PNODE proxy found)
                        security OK
                else
                        security check fails
        else
                security check fails
else if (snodeid is coded with ID only)
        if (proxy.attempt=yes)
                if (ID@PNODE proxy found)
                        security OK
                else
                        security check fails
        else
                security check fails
else if (snodeid is not coded)
        if (submitter&;PNODE proxy found)
                security OK
        else
                security check fails
```

Parameter	Description	Value
rpc.pmgr.port	The information is needed to monitor connection requests from the CLI using TCP/IP. This port identifies the communication port for the PMGR RPC server which uses the TLI/XTI interface to communicate with CLI. This parameter is only applicable for HPUXIT and SOLARIS platforms. It is a mandatory parameter for HPUXIT and SOLARIS-based deployments. **Note:** The Connect:Direct server must be restarted for changes in parameter value to come into effect.	The format is *nnnn*. Where nnnn is a decimal number between 1024 and 65535. The default value is 1367.
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`**.	`0–10000` The value is in seconds. The default value is `0`.
sess.total	The maximum number of concurrent connections between all nodes and the local node. The sum of **`api.max.connects`** and **`sess.total`** cannot exceed the number of file descriptors available. This value is system dependent. You must define enough file descriptors to handle the number of concurrent Connect:Direct sessions allowed, which can be as many as 999.	`0–999` A 1–3 digit number. 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. 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`**.	`0–999` 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. 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`**.	`0–999` 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.	`1–50` The default is `1`.
tcp.api.bufsize	The buffer size for transmitting data to and from an Connect:Direct CLI/API.	This value has no limit. The default is 32768 bytes.

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. Multiple *host name/IP addresses* and *port* combinations can be specified when they are separated by a comma. This parameter is required.	*host name/IP address;nnnn*
		
		*host nam*e—is the name of the Connect:Direct host computer.
		
		IP address—is the IP address of a machine running Connect:Direct:
		
		nnn.nnn.nnn.nnn (IPv4) or *nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn* (IPv6)
		
		port—identifies the communications port for Connect:Direct. The format is *nnnn*, where *nnnn* is a decimal number. A semi-colon separates the *host name/IP address* from the *port*:
		
		`msdallas-dt;1363`
		
		You can specify multiple *address/host name* and *port* combinations (separated with a comma):
		
		`10.23.107.5;1363, fe00:0:0:2014::7;1363, msdallas-dt;1363`
		
		For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.

tcp.api.inactivity. timeout	This is the maximum time a CMGR waits before exiting when it has not received a command from a client program.	`0–86399` (23 hours, 59 minutes, and 59 seconds) The value is in seconds. The default is `0`, which indicates no timeout occurs.

Parameter	Description	Value
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, an appropriate communication error is logged. If the time expires during a session start attempt, the process is moved to the Timer queue and Sterling Connect:Direct attempts to re-establish a session with the remote node. When set to 0, wait time is unlimited.	`0–10000` The value is in seconds. The default value is `180`.

TCP/IP Default Record

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 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**.
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). The default is **65536** bytes.
conn.retry.exhaust. action	Action to take after the specified short and long-term retries have been used.	hold | delete hold—Places Processes in the Hold queue in Held in Error status, after all retry attempts are used. This is the default value. delete—Causes the Processes to be deleted from the TCQ.

Parameter	Description	Value
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, an appropriate communication error is logged. If the time expires during a session start attempt, the process is moved to the Timer queue and Sterling Connect:Direct attempts to re-establish a session with the remote node. When set to 0, wait time is unlimited.	0–10,000 The value in seconds. The default value is **180**. 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.	0–10000 The value in seconds. The default value is **0**.
contact.name	The name of the administrator or operator.	Name
contact.phone	The phone number of the IBM Connect:Direct administrator or operator.	Phone number
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. The sum of api.max.connects and sess.total cannot exceed the number of file descriptors available. This value is system dependent.	0–999 A 1–3 digit number. The default is **255**.
sess.pnode.max	The maximum concurrent connections, where the local node is the initiator of the session.	0–999 The default is **255**.
sess.snode.max	The maximum concurrent connections, where the local node is the secondary node in a session.	0–999 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.	1–50 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.	nnn The size of this number has no limit. The default is **0**, which indicates no pacing of this type.

Parameter	Description	Value
pacing.send.count	The number of send operations to perform before automatically waiting for a pacing response from the remote node.	No limit exists for the size of this value. The default is **0**, which indicates no pacing of this type.

Remote Node Connection Record

The remote node connection record contains information you can use to define default values for a generic remote node connection or customize to for a particular new remote node. Following are the remote node connection parameters.

Parameter	Description	Value
alternate.comminfo	Provides support for establishing netmap-checked sessions with systems with multiple IP addresses, such as IBM Connect:Direct/Plex z/OS. Use this parameter to list all IP addresses or host names that are part of the multiple IP address environment. For IBM Connect:Direct/Plex, this list should include the address of each IBM Connect:Direct/Server with a different IP address from the IBM 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. For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.	host name/IP address or * host name—Host name associated with the IP address, for example: :alternate.comminfo=hops (where hops is a machine on the local domain) :alternate.comminfo=hops.csg.stercomm.com (fully-qualified host name) IP address——the IP address of a machine running IBM Connect:Direct in IPv4 or IPv6 format: nnn.nnn.nnn.nnn (IPv4) or nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn (IPv6) For example: :alternate.comminfo=10.23.107.5 :alternate.comminfo=fe00:0:0:2014::7 Specify multiple addresses/host names by separating them with a comma (,). A space can be added after the comma for readability. For example: 10.23.107.5, fe00:0:0:2014::7, msdallas-dt *—Accepts any IP address. This turns off IP address validation. **Note:** Partial pattern matches are not supported, such as: *.mydomain.com, myplex??.mydomain.com.

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. When the local node is the SNODE, the alternate addresses are used in the Netmap check.	Fully-qualified host name/IP address;nnnn The host name/IP address and port are separated by a semi-colon (;). A comma separates the list of alternate communication paths, and the list is processed from the top down. For example: salmon;9400, 10.20.40.65;9500 For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.
comm.bufsize	The buffer size for transmitting data to and from a remote node on TCP/IP connections.	The value for TCP/IP has no limit (up to 2,147,483,623). The default is **65536** bytes.
comm.info	The information needed to initiate connection requests to remote nodes using TCP/IP. This information refers to the network card that the local IBM Connect:Direct node uses to initiate outbound requests. This value is required. • 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 (;).	For TCP/IP connections, specify the host name or the IP address and port number. The default port is **1364**. For more information on specifying IP addresses and host names, see IP Addresses, Host Names, and Ports.
comm.transport	The transport protocol for the remote node.	tcp tcp—TCP/IP connections
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 **3**.
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 The default is **00.10.00**, or 10 minutes.

Parameter	Description	Value
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 | delete hold—Places Processes in the Hold queue in Held in Error status, after all retry attempts are used. This is the default value. delete—Causes the Processes to be deleted from the TCQ.
contact.name	The name of the IBM Connect:Direct administrator or operator.	Name
contact.phone	The phone number of the IBM Connect:Direct administrator or operator.	Phone number
descrip	Comments to include as part of the record.	An unlimited string
pacing.send.count	The number of send operations to perform before automatically waiting for a pacing response from the remote node.	No limit exists for the size of this value. The default is **0**, which indicates no pacing of this type.
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.	nnn The size of this number has no limit. The default is **0**, which indicates no pacing of this type.
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.	0–10000 The default value is **0**.
sess.total	The maximum number of concurrent connections between all nodes and the local node. The sum of api.max.connects and sess.total cannot exceed the number of file descriptors available. This value is system dependent.	0–999 A 1–3 digit number. The default is **255**.

Parameter	Description	Value
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. 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.	0–999 The default is **255**.
sess.snode.max	The maximum concurrent connections, where the local node is the secondary node in a session. 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.	0–999 The default is **255**.
tcp.crc	Turn on or off the CRC function for TCP/IP Processes on the remote node.	y | n The default is **n**.
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, an appropriate communication error is logged. If the time expires during a session start attempt, the process is moved to the Timer queue and Sterling Connect:Direct attempts to re-establish a session with the remote node. When set to 0, wait time is unlimited.	`0–10000` The value is in seconds. The default value is `180`.

Maintaining access information files

You can control access to IBM Connect:Direct through the following components:

- User authorization information file which contains local and remote user information records
- Strong access control file
- Program directory to limit access
- IBM Connect:Direct's ability to detect shadow passwords
- Security exit

User Authorization Information File

In order for users to have access to IBM Connect:Direct and use IBM Connect:Direct commands and statements, you need to 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. To disable access to the software for a local user, delete or comment 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 IBM Connect:Direct.

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

Sample Mapping of Remote User IDs to Local User IDs

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	@	*	=	test02	Remote user ID “user” on all remote nodes is mapped to local user ID test02.
*	@	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=:\
   :cmd.s+conf=n:
```

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

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.

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

Parameter	Description	Value
admin.auth	Determines if you has administrative authority. If set to y, you can perform all of the commands by default, but the specific command parameters override the default. If set to n, the specific command parameters must be granted individually.	y | <u>n</u> y—User has administrative authority. n—User does not have administrative authority. The default is **n**.
client.cert_auth	Determines if you can perform certificate authentication for client API connections. y—Enables client certificate authentication for you n—Disables client certificate authentication for you	y | <u>n</u>
client.source_ip	Use this parameter to list all of the IP addresses and/or host names that are valid for this user's API connection. If you specify values for this field, the IP address of this user's API connection is validated with the client.source_ip list. If the IP address does not match the one specified on the list, the connection is rejected.	A comma-separated list of client IP addresses or host names associated with client IP addresses. The IP address of the client connection for this user must match the address configured in this field. For example: nnn.nnn.nnn.nnn, localhost
cmd.chgproc	Determines if you can issue the change process command. A “y” value enables you to issue the command to targets owned by that user. Whereas, “a” allows you to issue the command to targets owned by all users.	y | <u>n</u> | a y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**. a—Allows you to issue the command against targets owned by all users.

Parameter	Description	Value
cmd.delproc	Determines if you can issue the delete process command. A “y” value enables you to issue the command to targets owned by that user. Whereas, “a” allows you to issue the command to targets owned by all users.	y | n | a y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**. a—Allows you to issue the command against targets owned by all users.
cmd.flsproc	Determines if you can issue the flush process command. A “y” value enables you to issue the command to targets owned by that user. Whereas, “a” allows you to issue the command to targets owned by all users.	y | n | a y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**. a—Allows you to issue the command against targets owned by all users.
cmd.selproc	Determines if you can issue the select process command. A “y” value enables you to issue the command to targets owned by that user. Whereas, “a” allows you to issue the command to targets owned by all users.	y | n | a y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**. a—Allows you to issue the command against targets owned by all users.
cmd.viewproc	Determines if you can issue the view process command. A “y” value enables you to issue the command to targets owned by that user. Whereas, “a” allows you to issue the command to targets owned by all users.	y | n | a y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**. a—Allows you to issue the command against targets owned by all users.
cmd.selstats	Determines if you can issue the select statistics command. A “y” value enables you to issue the command to targets owned by that user. Whereas, “a” allows you to issue the command to targets owned by all users.	y | n | a y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**. a—Allows you to issue the command against targets owned by all users.

Parameter	Description	Value
cmd.stopndm	Determines if you can issue the stop command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
cmd.s+conf	Determines if you can issue commands from network clients, such as IBM Control Center or Java API, to configure Connect:Direct Secure Plus. **Note:** This parameter has no effect on local tools, such as spadmin.sh and spcli.sh.	y | n y—Allows you to issue commands. The default is **y**. n—Prevents you from issuing commands.
cmd.submit	Determines if you can issue the submit process command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
cmd.trace	Determines if you can issue the trace command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
cmd.rfip	Determines if you can issue refresh initialization parameters (initparm.cfg file) command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
cmd.quiesce	Determines if you can issue the update Quiesce/Resume Record command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
cmd.chg.netmap	Determines if you can issue the update netmap (netmap.cfg file) command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.

Parameter	Description	Value
cmd.chg.users	Determines if you can issue the update local users (userfile.cfg file) command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
cmd.chg.proxies	Determines if you can issue the update proxy (remote) users (userfile.cfg file) command.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
descrip	Permits the administrator to add descriptive notes to the record.	Unlimited text string
name	The name of you.	User name
phone	The phone number of you.	user phone number
pstmt.copy	Determines if you can issue the **copy** statement.	y | n y—Allows you to issue the command. n—Prevents you from issuing the command. The default is **n**.
pstmt.copy.ulimit	The action taken when the limit on you 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 | nnnnnnnM | nnnnG y—Honors you file size limit. If this limit is exceeded during a copy operation, the operation fails. n—Ignores the limit. The default is **n**. nnnnnnnn, nnnnnnnnK, nnnnnnnM, or nnnnG—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 you 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 y—Allows you to send files. The default is **y**. n—Prevents you from sending files.

Parameter	Description	Value
pstmt.upload_dir	The directory from which you can send files. If a value is set for this parameter, then files can only be sent from this directory or subdirectories. The specified restriction is treated as the file system root while processing the send side of copy steps This means that any attempt to copy from a location outside of the specified restriction will fail, such as copying from a symbolic link inside the restriction that points to a location outside the restriction, or using parent directory references (../..) that attempt to navigate above the restriction. The restriction is the default directory for unqualified file specifications. A fully qualified file specification beginning at the actual system root will also succeed if the first part of the specification matches the restriction. For example, assume file /aaa/bbb/ccc.txt exists on the system, and the directory restriction specified is /aaa. Then the following copy step specifications will succeed: • /aaa/bbb/ccc.txt • bbb/ccc.txt • /bbb/ccc.txt **Note:** If a file open exit is in use, this parameter is passed to the exit, but is not enforced.	Directory path name
pstmt.download	Determines if you 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 y—Allows you to receive files. The default is **y**. n—Prevents you from receiving files.

Parameter	Description	Value
pstmt.download_dir	The directory to which you can receive files. If a value is set for this parameter, then files can only be received to this directory or subdirectories. The specified restriction is treated as the file system root while processing the receive side of copy steps. This means that any attempt to copy to a location outside of the specified restriction will fail, such as copying to a symbolic link inside the restriction that points to a location outside the restriction, or using parent directory references (../..) that attempt to navigate above the restriction. The restriction is the default directory for unqualified file specifications. A fully qualified file specification beginning at the actual system root will also succeed if the first part of the specification matches the restriction. For example, assume directory /aaa/bbb exists on the system, and the directory restriction specified is /aaa. Then the following copy step specifications will succeed: • /aaa/bbb/ccc.txt • bbb/ccc.txt • /bbb/ccc.txt **Note:** If a file open exit is in use, this parameter is passed to the exit, but is not enforced.	Directory path name

Parameter	Description	Value
pstmt.run_dir	The directory where IBM Connect:Direct is installed that contains the programs and scripts you executes with run job and run task statements. Any attempt to execute a program or script outside the specified directory fails. The UNIX Restricted Shell provides enhanced security by restricting you to the commands contained in the pstmt.run_dir. If you does not specify pstmt.run_dir, the commands are started with the Bourne shell. 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 Restrict Record.	Directory path name
pstmt.runjob	Specifies whether you can issue the **run job** statement.	y | n y—Allows you to issue the statement. n—Prevents you from issuing the statement. The default is **n**.
pstmt.runtask	Specifies whether you can issue the **run task** statement.	y | n y—Allows you to issue the statement. n—Prevents you from issuing the statement. The default is **n**.
pstmt.submit	Specifies whether you can issue the **submit** statement.	y | n y—Allows you to issue the statement. n—Prevents you from issuing the statement. The default is **n**.
pstmt.submit_dir	The directory from which you can submit Processes. This is for **submit**s within a Process.	Directory path name
snode.ovrd	Specifies whether you can code the **snodeid** parameter on the **submit** command and **process** and **submit statements.**	y | n y—Allows you to code the **snodeid** parameter n—Prevents you from coding the **snodeid** parameter. The default is **n**.

Parameter	Description	Value
pstmt.crc	Gives you the authority to specify the use of CRC checking in a Process statement. Setting this parameter to y enables you to override the initial settings in the initialization parameters or network map settings files.	y | n y—Allows you to specify CRC checking on a Process statement. n—Prevents you from specifying CRCchecking on a Process statement. The default is **n**.
fileagent.auth	Determines if you can issue get/update File Agent JSON configuration command. A “v” value enables you to issue the get File Agent JSON configuration command. Whereas a “y” value enables you to issue both get/update File Agent JSON configuration command. The value "n" prevents you from issuing either commands.	y | n | v y— Allows you to issue both get/update File Agent JSON configuration command. n— Prevents you from issuing File Agent JSON configuration command. v— Allows you to issue only get File Agent JSON configuration.
proclib.auth	Determine if you can issue process library commands like add, delete, rename, list and get. A “v” value enables you to issue the get/list Process library command. Whereas a “y” value enables you to issue all the Process library commands. The value "n" prevents you from issuing any of the commands.	y | n | v y— Allows you to issue all the process library commands. n— Prevents you from issuing process library commands. v— Allows you to only view the process library via get/list commands.
cmd.external.stat.log	Determines if you can log stats in to Connect:Direct for UNIX from Integrated File Agent. A “y” value enables you to log stats from Integrated File Agent into Connect:Direct for Unix. A “n” value prevents you from logging stats from Integrated File Agent into Connect:Direct for Unix.	y | n y— Allows you to log stats from Integrated File Agent into Connect:Direct for Unix stats. n— Prevents you from logging stats from Integrated File Agent into Connect:Direct for Unix stats.

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: To prevent the remote user from using IBM Connect:Direct, delete or comment 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 | n y—Allows a user to issue the statement. n—Prevents a user from issuing the statement. The default is **n**.
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 y—Allows a user to send files. The default is **y**. n—Prevents a user from sending files.
pstmt.upload_dir	The directory from which you can send files. If a value is set for this parameter, then files can only be sent from this directory or subdirectories. The specified restriction is treated as the file system root while processing the send side of copy steps and is the default directory for unqualified file specifications. A fully qualified file specification beginning at the actual system root will also succeed if the first part of the specification matches the restriction. For example, assume file /aaa/bbb/ccc.txt exists on the system, and the directory restriction specified is /aaa. Then the following copy step specifications will succeed: • /aaa/bbb/ccc.txt • bbb/ccc.txt • /bbb/ccc.txt **Note:** If a file open exit is in use, this parameter is passed to the exit, but is not enforced.	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 y—Allows a user to receive files. The default is **y**. n—Prevents a user from receiving files.

Parameter	Description	Value
pstmt.download_dir	The directory to which you can receive files. If a value is set for this parameter, then files can only be received to this directory or subdirectories. The specified restriction is treated as the file system root while processing the receive side of copy steps and is the default directory for unqualified file specifications. A fully qualified file specification beginning at the actual system root will also succeed if the first part of the specification matches the restriction. For example, assume directory /aaa/bbb exists on the system, and the directory restriction specified is /aaa. Then the following copy step specifications will succeed: • /aaa/bbb/ccc.txt • bbb/ccc.txt • /bbb/ccc.txt **Note:** If a file open exit is in use, this parameter is passed to the exit, but is not enforced.	Directory path name
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. 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 Restrict Record.	Directory path name
pstmt.submit_dir	The directory from which the user can submit Processes. This is for **submit**s within a Process.	Directory path name
pstmt.runjob	Specifies whether the user can issue the **run job** statement.	y | n y—Allows a user to issue the statement. n—Prevents a user from issuing the statement. The default is **n**.

Parameter	Description	Value
pstmt.runtask	Specifies whether the user can issue the **run task** statement.	y | n y—Allows a user to issue the statement. n—Prevents a user from issuing the statement. The default is **n**.
pstmt.submit	Specifies whether the user can issue the **submit** statement.	y | n y—Allows a user to issue the statement. n—Prevents a user from issuing the statement. The default is **n**.
descrip	Permits you to add descriptive notes to the record.	Text string

Strong Access Control File

To provide a method of preventing an ordinary user from gaining root access through IBM Connect:Direct, 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 the root through Connect:Direct for UNIX. If you want to give an ordinary user root access through Connect:Direct for UNIX, you may need to access and update the **sysacl.cfg** file.

Note: 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.

If root is defined as a local user in userfile.cfg, then the root:deny.access parameter, which is specified in the sysacl.cfg file, further allows, denies, or limits root access to IBM Connect:Direct. This parameter is required, even if root is not defined as a local user in userfile.cfg. The following values can be specified for the **root:deny.access** parameter:

Parameter	Description	Value
deny.access	Allows, denies, or limits root access to IBM Connect:Direct	y | n | d y—No Processes can acquire root authority n—PNODE Processes can acquire root authority, but SNODE Processes can not. This is the default value. d—Any Process can acquire root authority

For example, given a userfile.cfg with the following entries:

```
remoteUser@remoteNode:\
:local.id=root:
```

```
root:\
 :admin.auth=Y:
```

and sysacl.cfg with:

```
root:\
 :deny.access={x, where x is as described below}:
```

- Incoming process submitted by remoteUser@remoteNode
  - n, connection denied due to security failure
  - y, connection denied due to security failure
  - d, connection allowed to proceed
- Outgoing process submitted on the local node by root
  - n, connection allowed to proceed
  - y, connection denied due to security failure
  - d, connection allowed to proceed

If a user is denied access because of the **user: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.

Limiting Access to 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 >>)

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.

For more information on the Security Exit, see User Exit Record.

Maintaining client and server authentication key files

IBM Connect:Direct client/server security depends on a key, similar to a password, in a IBM 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. You can edit both key files with any text editor installed on your system.

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`.

CAUTION: To mitigate brute force attacks to break the `keys.client` and `keys.server` authentication keys, IBM strongly recommends that customers use authentication key values with a minimum length of 15 characters.

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] ] ]
```

Key File Parameters

The following table describes the available key file parameters:

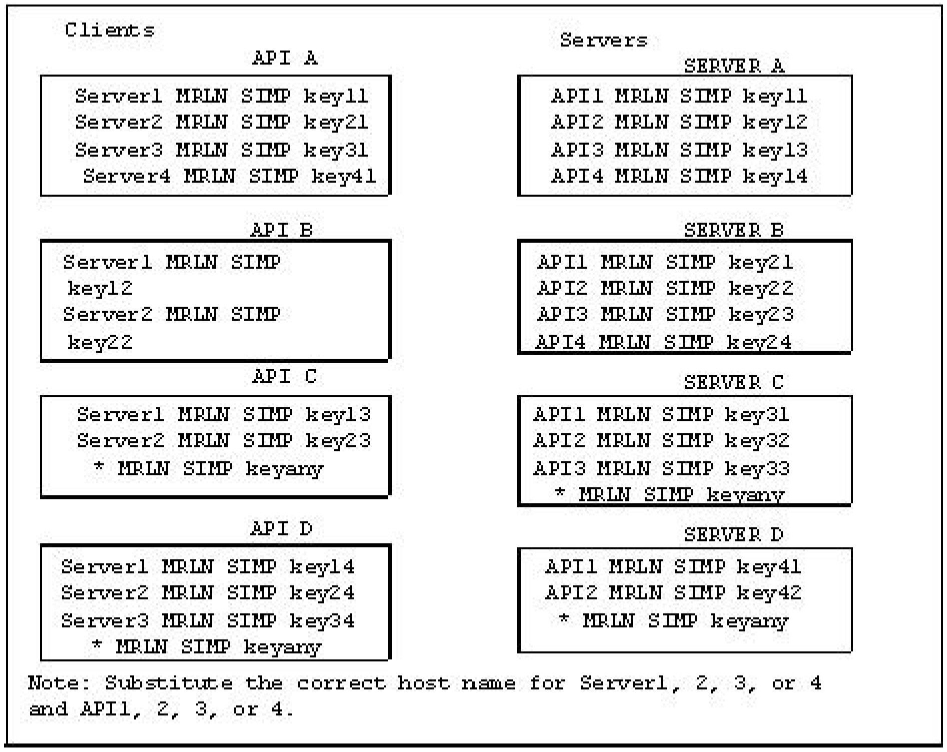
Parameter	Description	Value
hostname	The host name of the server with which you want to communicate or the host name of the API you will allow to communicate with your server. The hostname is followed by one or more space characters. If you replace the host name with an asterisk (*) character in the 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

Parameter	Description	Value
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 (/). **Note:** Key values longer than 22 characters do not generate an error; however, only the last 22 characters entered are used. Any characters prior to the last 22 are ignored.

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.



API A

```
Server1 MRLN SIMP key11
Server2 MRLN SIMP key21
Server3 MRLN SIMP key31
Server4 MRLN SIMP key41
```

SERVER A

```
API1 MRLN SIMP key11
API2 MRLN SIMP key12
API3 MRLN SIMP key13
API4 MRLN SIMP key14
```

API B

```
Server1 MRLN SIMP
key12
Server2 MRLN SIMP
key22
```

SERVER B

```
API1 MRLN SIMP key21
API2 MRLN SIMP key22
API3 MRLN SIMP key23
API4 MRLN SIMP key24
```

API C

```
Server1 MRLN SIMP key13
Server2 MRLN SIMP key23
* MRLN SIMP keyany
```

SERVER C

```
API1 MRLN SIMP key31
API2 MRLN SIMP key32
API3 MRLN SIMP key33
* MRLN SIMP keyany
```

API D

```
Server1 MRLN SIMP key14
Server2 MRLN SIMP key24
Server3 MRLN SIMP key34
* MRLN SIMP keyany
```

SERVER D

```
API1 MRLN SIMP key41
API2 MRLN SIMP key42
* MRLN SIMP keyany
```

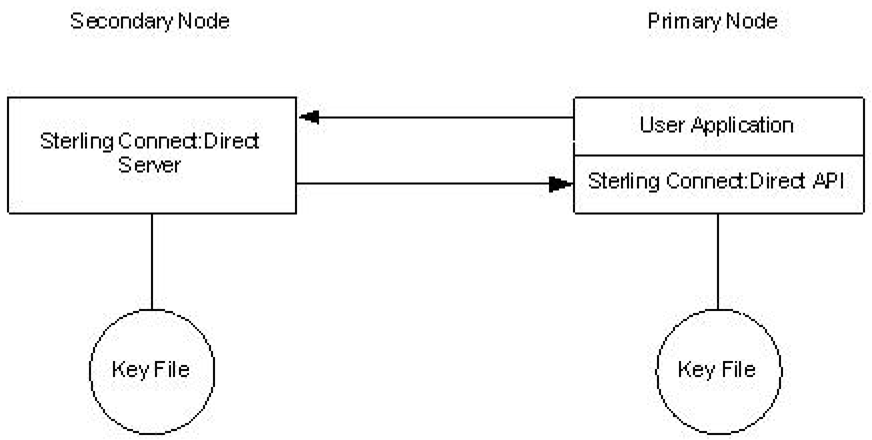
Note: Substitute the correct host name for Server1, 2, 3, or 4 and API1, 2, 3, or 4.

Authentication Process

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

The goal of IBM 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

IBM Connect:Direct API, to ensure that the authentication process 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.

Session Establishment

Session establishment for TCP affects how you set up firewall rules and configure the firewall navigation initialization parameters in IBM Connect:Direct.

TCP Session Establishment

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

Firewall Navigation

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

Before you configure source ports in the IBM Connect:Direct initialization parameters, you need to review all information regarding firewall navigation and rules, especially if you are implementing firewalls.

Implement Firewall Navigation

To implement firewall navigation in IBM Connect:Direct:

1. Coordinate IP address and associated source port assignment with your local firewall administrator before updating the firewall navigation record in the initialization parameters file.
2. Add the following parameters to the IBM Connect:Direct initialization parameters file as needed, based on whether you are using TCP:
   - tcp.src.ports
   - tcp.src.ports.list.iterations
   - udp.src.ports
   - udp.src.ports.list.iterations
3. Coordinate the specified port numbers with the firewall administrator at the remote site.

Firewall Rules

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

TCP Firewall Navigation Rules

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

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

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

TCP Firewall Configuration Example

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

- tcp.src.ports = (333.333.333.333, 2000–2200)
- tcp.src.ports.list.iterations = 1

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

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

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

Specifying connection information

IBM Connect:Direct accepts both Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) versions of the Internet Protocol as well as host names. You can enter IP addresses/host names and ports in several ways, depending on the field you are specifying:

- Address or host name only
- Port number only
- Address/host name with a port number
- Multiple address/host name and port combinations

When specifying IP addresses/host names and ports for IBM Connect:Direct, use the following guidelines.

IP Addresses

IBM Connect:Direct accepts both IPv4 and IPv6 addresses. Wherever an IP address is specified in IBM Connect:Direct, you can use either IPv4 or an IPv6 addresses.

IPv4 Addresses

IPv4 supports 2^{32} addresses written as 4 groups of dot-separated 3 decimal numbers (0 through 9), for example, 10.23.107.5.

IPv6 Addresses

IPv6 supports 2^{128} addresses written as 8 groups of colon-separated 4 hexadecimal digits, for example, 1001:0dc8:0:0:0:ff10:143e:57ab. The following guidelines apply to IPv6 addresses:

- If a four-digit group contains zeros (0000), the zeros may be omitted and replaced with two colons (::), for example:

```
2001:0db8:85a3:0000:1319:8a2e:0370:1337
can be shortened as
2001:0db8:85a3::1319:8a2e:0370:1337
```

- Any number of successive 0000 groups may be replaced with two colons (::), but only one set of double colons (::) can be used in an address, for example:

```
001:0db8:0000:0000:0000:0000:1319:58ab
Can be shortened as:
2001:0db8:0000:0000::1319:58ab
```

- Leading zeros in a four-zero group can be left out (0000 can be shortened to 0), for example:

```
2001:0db8:0000:0000:0000:0000:1319:58ab
Can be shortened as:
2001:0db8:0:0:0:0:1319:58ab
```

- You can write a sequence of 4 bytes that occur at the end of an IPv6 address in decimal format using dots as separators, for example:

```
::ffff:102:304
Can be written as:
::ffff:1.2.3.4
```

This notation is useful for compatibility addresses.

Host Names

When you specify a host name, rather than an IP address, IBM Connect:Direct gets the IP address from the operating system. The first IP address returned by the operating system is used regardless of whether it is in IPv4 or IPv6 format.

A host name (net, host, gateway, or domain name) is a text string of up to 24 characters comprised of the alphabet (A–Z), digits (0–9), minus sign (-), and period (.), for example, msdallas-dt.

The following guidelines also apply:

- No blank or space characters are permitted as part of the name.
- Periods are allowed only when they are used to delimit components of domain-style names.
- Host names are not case sensitive.
- The first and last character must be a letter or digit.
- Single-character names or nicknames are not allowed.

Port Numbers

Port numbers can be appended to the end of IP/host addresses when they are preceded by a semi-colon (;), for example, 10.23.107.5;1364. This convention is specific to IBM Connect:Direct and is not an industry standard.

A port number must be in the range of 0 through 65535. Port numbers lower than 1024 are designated as reserved and should not be used. The following examples show port numbers appended to IP/host addresses using these conventions:

```
10.23.107.5;1364
fe00:0:0:2014::7;1364
msdallas-dt;1364
```

Multiple Addresses, Host Names, and Ports

You can specify multiple IPv4 and IPv6 addresses and host names by separating them with a comma (,). A space can be added after the comma for readability, for example:

```
10.23.107.5, fe00:0:0:2014::7, msdallas-dt
```

You can also specify a port number for each address or host name. The port is separated from its corresponding address/host name with a semi-colon (;), and each address/host name and port combination is separated by a comma (,). A space may be added after the comma for readability. The following example shows multiple address/host name and port combinations:

```
10.23.107.5;1364, fe00:0:0:2014::7;1364, msdallas-dt;1364
```

Multiple address/host names (and combinations with port numbers) are limited to 1024 characters.

About Using Masks for IP Address Ranges

When you specify a value for the **tcp.src.ports** parameter in the initialization parameters file, you can use masks to specify the upper boundary of a range of IP addresses that will use a specific port, multiple ports, or a range of ports. IBM Connect:Direct supports masks for both IPv4 and IPv6 addresses as shown in the following sample entry from the **initparms.cfg** file:

```
tcp.src.ports=(199.2.4.*, 1000), (fd00:0:0:2015:*::*, 2000-3000), (199.2.4.0/
255.255.255.0, 4000-5000), (fd00:0:0:2015::0/48, 6000, 7000)
```

These sample addresses specify the following information:

`(199.2.4.*, 1000)`—Any IPv4 address that falls in the range from 199.2.4.0 through 199.2.4.255 will use only port 1000.

`(fd00:0:0:2015:*::*, 2000-3000)`—Any IPv6 address that falls in the range from fd00:0:0:2015:0:0:0:0 through fd00:0:0:2015:ffff:ffff:ffff:ffff will use a port in the range of 2000 through 3000.

`(199.2.4.0/255.255.255.0, 4000-5000)`—Any IPv4 address that falls in the range from 199.2.4.0 through 199.2.255.255 will use a port in the range of 4000 through 5000.

`(fd00:0:0:2015::0/48, 6000, 7000)`—Any IPv6 address that falls in the range from fd00:0:0:2015:0:0:0:0 through fd00:0:0:ffff:ffff:ffff:ffff:ffff will use port 6000 or port 7000.

As shown in the sample entry above, the wildcard character (*) is supported to define an IP address pattern. You can specify up to 255 unique IP address patterns or up to 1024 characters in length, each with its own list of valid source ports. If the wildcard character is used, the optional mask is not valid.

Using IBM Connect:Direct in a test mode

You can enable a test mode for production instances of IBM Connect:Direct to perform the following functions:

- Test new applications and customer connections
- Prevent future production work from executing until testing is complete after you have terminated all active production work using the Flush Process command
- Resume regular production work after testing
- Control individual file transfers by application
- Enable and disable individual nodes and applications

While testing is being conducted, only Processes, particularly file transfers, involved with the testing activity are executed. No production data is transferred to applications being tested while at the same time no test data is transferred to production applications.

Processing Flow of the Test Mode

You enable the testing mode using the **quiesce.resume** initialization parameter and specify which IBM Connect:Direct Processes to run and not run by storing your preferences as text records in a parameter table named NDMPXTBL. A sample parameters file, NDMPXTBL.sample, is located in the /ndm/src directory. After you have updated the file for your testing environment, place it in the installation ndm/cfg/ <nodename> directory. If you enable the quiesce.resume parameter, you must have an NDMPXTBL table to operate IBM Connect:Direct in a test mode.

You can specify the following criteria that are used to find matches for one or more Processes to include (using the “I” command code) or exclude (“X” command code) from execution:

- A partial or full Process name
- A partial or full remote node name
- A partial or full IBM Connect:Direct submitter ID and submitter node combination

In addition to telling IBM Connect:Direct which Processes to run, you tell the system what to do with the Processes which do not get executed. You can specify the following dispositions for Processes not permitted to run:

- Place the Process in the Hold queue
- Place the Process in the Timer queue for session retry
- Flush the Process from the queue

For more information on how the testing mode can be used, see Sample Test Scenarios.

When the testing mode is enabled, IBM Connect:Direct performs a syntax check on the parameter table and fails initialization if the table is invalid. If the table is valid, IBM Connect:Direct scans it looking for a pattern that matches the Process that is about to execute. If a match is found, the Process is permitted to execute if the “I” (Include) command code is in effect. If command code “X” (Exclude) is in effect, the process is not permitted to execute. If a match is not found in the table, the opposite processing occurs from the case where a match is found, that is, if no match is found and command code “I” is in effect, the Process is not permitted to execute, whereas if command code “X” is in effect, the Process is permitted to execute.

If a Process is not permitted to execute, the disposition specified in the NDMPXTBL parameter table to either hold, retry, or flush the Process is implemented and a non-zero return code is returned. When a Process is prevented from executing in testing mode, appropriate messages are issued and can be viewed in the statistics log.

For Processes initiated on remote nodes, the testing mode functions in the same manner as it does for Processes submitted on the local IBM Connect:Direct node except that the remote node is the PNODE (Process owner) for that Process, and the local node is the SNODE (secondary node). The NDMPXTBL Parameter Table is searched for a matching entry, and the remotely-initiated Process is either permitted to execute or is excluded from execution. Because the local node is the SNODE for this type of transfer, it cannot enforce the Process disposition setting in the NDMPXTBL parameter table. The remote PNODE determines how the Process is handled. Typically, the remote node places the Process in the Hold queue with a status of “HE” (Held in Error).

Preparing the NDMPXTBL Parameter Table

You can use any text editor to modify the sample NDMPXTBL parameter table supplied with IBM Connect:Direct. When you update the parameter table, name it NDMPXTBL and save it to the Server directory of the installation. The parameter table file can be created or updated while the server is active, and any changes made to the file take effect for sessions that begin after the changes are made.

Similarly, the **quiesce.resume** initialization parameter can be modified while the server is active. For more information on the **quiesce.resume** initialization parameter, see Quiesce/Resume Record.

Note: If you enable the quiesce.resume initialization parameter, you must have an NDMPXTBL parameter table.

NDMPXTBL Parameter Table

Each table entry or record consists of a single-character command code in column one. Most command codes have a parameter which begins in column two and varies according to the command code function.

Command Code	Description	Subparameters/Examples
*	Comment line.	* Only run the following Processes.
E	Enables execution of Processes based on table entries. Either “E” or “D” must be the first non-comment entry in the table.	The second column in this entry must contain one of the following values which indicates the disposition of a PNODE Process if it is not allowed to run. H—Places the Process in the Hold queue R—Places the Process in the Timer queue in session retry F—Flushes the Process from the queue
D	Disables the execution of all Processes regardless of the contents of the parameter table and fails Process execution with a non-zero (error) return code and message LPRX003E. Either “E” or “D” must be the first non-comment entry in the table	The parameter for command code “E” can also be specified in column two. This is a convenience to make it easier to change from “E” to “D” and vice versa without having to change column two to a blank for command code “D.”
P	Matches Processes based on a full or partial Process name. Supports the wild card trailing asterisk (*). Can be used to enable or disable Process execution for a particular application by using naming conventions to match an application.	PCOPY—Matches a single Process PACH*—Matches all Processes beginning with “ACH” P*—Matches all Processes
N	Matches Processes based on a full or partial remote node name. Supports the wild card trailing asterisk (*).	NCD.NODE1—Matches a single remote node name NCD.NODEA*—Matches all remote node names beginning with “CD.NODEA” N*—Matches all remote node names
S	Matches Processes based on a full or wild card IBM Connect:Direct submitter ID and submitter node combination. The format is <id>@<node>.	SACTQ0ACD@TPM002—Matches a specific ID and node combination. S*@TPM002—Matches all IDs from node TPM002 SACTQ0ACD@*—Matches ID ACTQ0ACD from all nodes S*@*—Matches all IDs from any node. This is another way to match all Processes.

Command Code	Description	Subparameters/Examples
I	Includes Processes for execution that match the patterns in the table which follow this command code. Either “I” or “X” must be the second non-comment entry in the table. Processes which do not match a pattern in the table are not executed. **Note:** To choose which command code to use to select Processes, determine which group is smaller and use the corresponding command Code. For example, if the number of Processes to be executed is smaller than the number of Processes to exclude from execution, specify “I” as the command code and add patterns to match that group of Processes.	ER I NCD.BOSTON Includes Processes for execution on the CD.BOSTON node only. Processes destined for all other remote nodes are placed in the Timer queue in session retry
X	Excludes from execution those Processes that match the patterns in the table which follow this command code. Either “X” or “I” must be the second non-comment entry in the table. Processes which do not match a pattern in the table are executed.	EH X SDALLASOPS@* Excludes Processes for execution submitted by the ID DALLASOPS from any node
L	Last entry in table.	

Sample Test Scenarios

The following examples show different applications of the test mode using the NDMPXTBL parameter table to define which IBM Connect:Direct Processes to run and not run.

Specifying Which Processes Run

In this example, IBM Connect:Direct executes all Processes that start with ACH or are named DITEST01 or DITEST02. All other Processes are placed in the Hold queue.

```
* Enable processing. Only permit processes matching one of the patterns
* to execute. Hold processes that don't execute.
EH
I
PACH*
PDITEST01
PDITEST02
L
```

Specifying Which Processes to Exclude

In this example, IBM Connect:Direct does not execute any Process that starts with ACH or is named DITEST01 or DITEST02. All other Processes are executed.

```
* Exclude matching processes.  Permit all others to execute.
EH
X
PACH*
PDITEST01
PDITEST02
L
```

Permitting Process Execution by Secondary Node and Submitter User ID/Node

In this example, IBM Connect:Direct executes all Processes that match one of the following criteria:

- The specific secondary node (SNODE) name is DI.NODE1
- An SNODE whose name starts with DI0017
- Any IBM Connect:Direct submitter ID from node DI0049
- The specific IBM Connect:Direct submitter ID ACHAPP from any node

All Processes not matching one of the above criteria are flushed from the queue.

```
* Only permit matching processes to execute.  Flush those that do not.
EF
I
NDI.NODE1
NDI0017*
S*@DI0049
SACHAPP@*
L
```

Stopping the Test Mode

In this example, no Processes will be executed, and a non-zero return code will be displayed, which signifies an error along with message ID LPRX003E. The remainder of the table is ignored (including the "F" code to flush Processes from the queue), and all Processes are placed in the Hold queue.

To resume testing, change the "D" command code to an "E."

```
* Execute no processes at all.  Put them in the hold queue and return.
DF
I
PACH*
PDITEST01
PDITEST02
L
```

Chapter 4. User Guide

The User Guide contains all the information you need in order to use Connect:Direct for UNIXto configure and queue processes, use various system utilities, and create custom programs and user exits.

Controlling and Monitoring Processes

Use the Command Line Interface (CLI) to submit IBM Connect:Direct Processes and commands from a native command line environment. You can also use the Connect:Direct Browser User Interface to perform some of these tasks.

Starting the CLI

1. If you have not defined the NDMAPICFG environment variable, type the following command for the appropriate shell, where *d_dir* is the path to the IBM Connect:Direct subdirectory.

   In the C shell:

   ```
   % setenv NDMAPICFG d_dir/ndm/cfg/cliapi/ndmapi.cfg
   ```

   In the Bourne or Korn shell:

   ```
   $ NDMAPICFG=d_dir/ndm/cfg/cliapi/ndmapi.cfg
   $ export NDMAPICFG
   ```

2. Type the following command to invoke IBM Connect:Direct CLI. Type options as required:

   ```
   $ direct [-P string -s -t n -e nn -n name -p nnnnn -x -r -h -z]
   ```

Stopping the CLI

- Stop the CLI operation by typing **Control-D** or **quit** at the prompt.

CLI Commands

Refer to the following table for a description of the command options and sample command entries:

Option	Description	Value	Sample Command Entry

-P	Identifies the custom string to use at the command line prompt.
	
	If the prompt string includes spaces or special characters, enclose it in single or double quotation marks.
	
	The prompt string can also be specified in the ndmapi.cfg file. If a prompt string is specified on the command line and in the ndmapi.cfg file, **-P** takes precedence.
	
	When the default prompt (“Direct”) is overridden, the new prompt string is shown at the command line prompt and in the welcome banner display.
		text string
		
		Up to 32 characters.
			$ direct -PNewPrompt
			
			$ direct -P”Test CD on Medea”

Option	Description	Value	Sample Command Entry
-s	Suppresses standard output. Use this option to view only the completion status of a command.	none	$ direct -s
-t n	Enables the CLI/API trace option. The level number, **n**, identifies the level of detail in the trace output.	1 | 2 | 4 Specify one of the following level numbers: 1—Provides function entry and function exit. This is the default. 2—Provides function entry and exits and basic diagnostic information, such as displaying values of internal data structures at key points in the execution flow. 4—Enables a full trace. All diagnostic information is displayed.	$ direct -t 4
-e nn	Defines the error level above which the CLI automatically exits. If the returned error code is greater than the error level specified, the CLI automatically exits. Use this command within shell scripts. This parameter prevents unwanted execution of commands following a command that generates an error above the specified level. When the CLI terminates, it returns a UNIX exit code that can be tested by the shell.	0 | 4 | 8 | 16 Valid values in the error level code are: 0—Indicates successful completion. 4—Indicates warning. 8—Indicates error. 16—Indicates catastrophic error.	$ direct -e 16
-n name	Identifies the host name of the computer where the IBM Connect:Direct server (PMGR) is running. **Note:** Invoking direct with **-p** or **-n** overrides the settings in the ndmapi.cfg file.	IBM Connect:Direct host name	$ direct -n hostname
-p nnnnn	Identifies the communications port number for the IBM Connect:Direct node. **Note:** Invoking direct with **-p** or **-n** overrides the settings in the ndmapi.cfg file.	1024–65535. The format is nnnnn.	$ direct -p 2222

Option	Description	Value	Sample Command Entry
-x	Displays command input on standard out. Use this command when debugging scripts.	none	$ direct -x
-r	Makes the Process number available to user-written shell scripts. The CLI displays a special string, _CDPNUM_ followed by a space, followed by the Process number.	none	$direct -r | grep "_CDPNUM_"
-h	Displays command usage information if a IBM Connect:Direct command is typed incorrectly.	none	$ direct -h
-z	Appends a newline character after a prompt.	none	$ direct -z
-a	Dynamically enables client authentication tracing, intended to assist in investigating CLI connection or authorization failures. This trace is captured in the d_dir/ndm/bin/ndmauthc.log file. The trace option `-a` can be used independently of the existing `-t` CLI trace option.	none	$ direct -a

CLI Job Control

IBM Connect:Direct enables you to switch the CLI Process between the foreground and the background in shells that support job control. This capability enables you to edit the text of saved Processes, issue UNIX commands, and resolve Process errors without exiting and reentering the CLI. Use the following commands to switch the CLI Process:

- Press the suspend character (**Control-Z**) to stop or suspend the CLI Process.
- Issue the **`fg`** command to move the CLI Process to the foreground.

Note: If you experience problems with job control, contact your system administrator for suggestions on additional UNIX commands to use.

CLI History Commands

IBM Connect:Direct enables you to use the history commands available with UNIX. History commands do not need the semicolon (;) at the end of the command. The following table lists the available history commands:

Command	Description
!!	Repeat the last command one time.
!#n	Set the number of commands to store in the history buffer. The default history buffer size is 50 commands.
!n	Repeat command number <n> in the history buffer.

Command	Description
!<string>	Repeat command beginning with the string <string>.
!?	List the contents of the history buffer.

Overview of IBM Connect:Direct Commands

You control and monitor IBM Connect:Direct Processes using the following commands:

Note: The CMGR currently limits the size of a Process file to 60K bytes.

Command	Abbreviation	Description
submit	sub	Makes Processes available for execution.
change process	cha pro	Changes the status and modifies specific characteristics, of a nonexecuting Process in the TCQ.
delete process	del pro	Removes a nonexecuting Process from the TCQ.
flush process	flush pro	Removes an executing Process from the TCQ.
stop	stop	Stops Connect:Direct for UNIX and returns control to the operating system.
select process	sel pro	Monitors both executing Processes and Processes waiting for execution. You can specify the search criteria and the form in which the information is presented.
select statistics	sel stat	Retrieves information from the statistics file. You can specify the search criteria and the form in which the information is presented.
view process	view pro	View a Process in the TCQ where the local node is the Pnode. View process can only display Processes running on the local node since only the Pnode has the information required to display a Process.

Abbreviations for Common IBM Connect:Direct Commands

The following table lists valid abbreviations for commonly used parameters for IBM Connect:Direct commands:

Parameter	Abbreviation
detail	det
quit	q
recids	rec
release	rel
pname	pnam, pna
pnumber	pnum
sunday	sun
monday	mon
tuesday	tue

Parameter	Abbreviation
wednesday	wed
thursday	thu
friday	fri
saturday	sat
today	tod
tomorrow	tom

Restricting the Scripts and UNIX Commands Users Can Execute

System administrators and other network operations staff can restrict the scripts and UNIX commands that you can execute with the run task and run job Process statements. System administrators and other network operations staff can enforce the following limits on the capabilities you have with IBM Connect:Direct:

- The capability to send or receive files; you may be limited either to sending files only or to receiving files only.
- The locations to or from which you can send or receive files; you may be limited to specific local or remote nodes.

Check with the system administrator for a list of specific restrictions for your user ID.

IBM Connect:Direct Command Syntax

Use the same command syntax for commands typed at the CLI prompt or used as the command text parameter for an **ndmapi_sendcmd()** function. Refer to "Writing User Exits" on page 212, for details on function calls. The following conventions are used when typing commands:

- When selecting a password or user ID, do not use IBM Connect:Direct keywords.
- Be aware that user names and file names are case sensitive.
- Type an individual command keyword in uppercase, lowercase, or mixed-case characters.
- Terminate all commands with a semicolon (;).
- When typing commands, type the entire command name or type the first three characters or abbreviate specific parameters. Refer to "Abbreviations for Common IBM Connect:Direct Commands" on page 150for a list of abbreviations.
- Do not abbreviate Process statements and parameters.
- File names, group names, user IDs, and passwords are variable length strings and can be any length.
- A IBM Connect:Direct node name is 1–16 characters long. The name of a record in the netmap describing a remote node is typically the remote IBM Connect:Direct node name, but can be any string 1–256 characters long. You can also specify a remote node name as an IP address or hostname and a port number or port name.

"Generic" Parameter Value

When the word generic is specified as a parameter value in a syntax definition, provide a string that can include the asterisk (*) and question mark (?) characters. These characters provide a pattern matching or wildcard facility for parameter values. The asterisk matches zero or more characters, and the question mark matches any single character. The following sample illustrates the use of the asterisk and question mark characters:

```
PNAME = A?PROD5*
```

The generic Process name specified in the previous sample shows a specification that matches all Processes beginning with the letter A, followed by any single character in position two with the string PROD5 in positions three through seven. The asterisk takes the place of zero or more characters beginning in position eight.

“List” Parameter Value

When (list) is a parameter value, you can specify multiple parameter values by enclosing the group in parentheses and separating each value with a comma. A list can also include generic values. The following command illustrates a list:

```
(pnumber1, pnumber2, pnumber3)
```

Submitting a Process

The submit command makes Processes available for execution and enables the software to interpret the Process statements contained in the specified files.

Parameters specified in the submit command override the same parameters specified on the Process statement. There are no required parameters. However, if you do not specify a file name for the file parameter, the text of the IBM Connect:Direct Process must follow the submit command. Following are the parameters for the submit command:

Parameter	Description	Values
file	The name of the Process file. The file name can include a path name indicating the location of the Process. This parameter must be the first parameter.	file name including the path name
class	The node-to-node session on which a Process can execute. A Process can execute on the class specified or any higher session class. The default class is specified as the **sess.default** parameter of the local.node record in the initialization parameters file.	1 | n A numeric value from 1 to the value of maximum concurrent local node connections (sess.pnode.max). The default value is **1**. The value cannot be greater than the maximum number of local sessions with primary control.
crc	Determines if crc checking is performed. This parameter overrides settings in the initialization parameter, the network map, and the Process. **Note:** The user must be assigned authority to change the crc settings in the user authority file.	on | off on—Turns on crc checking. off—Turns off crc checking. The default is **off**.

Parameter	Description	Values
hold	Determines if the Process is placed in the Hold queue. When a Process is submitted with retain=yes or retain=call, IBM Connect:Direct ignores the **hold** parameter.	yes | no | call yes—Specifies the Process is placed in the Hold queue in HI status until it is released by a change process command. A Process submitted with hold=yes is placed on the Hold queue even if you specify a start time. no—Specifies that the Process executes as soon as resources are available. This is the default. call—Specifies that the Process is held until a connection is established between the remote node and the local node. At that time, the Process is released for execution.
maxdelay	How long the submit command waits for the submitted Process to complete execution. This parameter is useful when the command is issued by a shell script. When this parameter is specified, the script waits until the Process completes before it continues execution. The return code of the Process is stored in the $? variable if you are using the Bourne or Korn shell and in $status variable if you are using the C shell, which the shell script can use to test the results of Process execution. If you do not specify maxdelay, no delay occurs. If the time interval expires, the submit command returns a warning status code and message ID to the issuing Process or CLI/ API. The Process is not affected by the time interval expiration and executes normally.	unlimited | *hh:mm:ss* | 0 unlimited—Waits until the Process completes execution. hh:mm:ss—Waits for an interval no longer than the specified hours, minutes, and seconds. 0—Waits until the Process completes execution. If you specify maxdelay=0, you get the same results as when you specify maxdelay=unlimited.
newname	A new Process name that overrides the name in the submitted Process.	A name up to 256 characters long
notify	The user e-mail to receive Process completion messages. This parameter uses the rmail utility available in the UNIX System V mail facility to deliver the completion messages. **Note:** IBM Connect:Direct does not validate the e-mail address or user ID supplied to the notify parameter. Invalid e-mail addresses and failed E-mail attempts are handled according to the local mail facilities configuration.	*username@hostname* or *user@localhost*
pacct	A string containing information about the PNODE. Enclose the string in double quotation marks.	"*pnode accounting data*" up to 256 characters

Parameter	Description	Values
pnodeid	Security user IDs and passwords at the PNODE. The pnodeid subparameters can contain 1–64 alphanumeric characters.	*id , pswd* id—Specifies a user ID on the PNODE. pswd—Specifies a user password on the PNODE. If you specify pnodeid, you must also specify id. Identify the ID first and the pswd last.
prty	The priority of the Process in the Transmission Control Queue (TCQ). A Process with a higher priority is selected for execution before a Process with a lower priority. The prty value does not affect the priority during transmission.	1–15, where fifteen is the highest priority. The default is **10**.
retain	Determines if IBM Connect:Direct retains a copy of the Process in the TCQ. IBM Connect:Direct assigns a Process number to the Process when it is placed in the retain queue. When it's time for a retained process to execute, a new process, which is a copy of the retained process, is created and assigned the next available process number and made available for execution. If you specify a start time and set **retain=yes**, the Process remains in the Timer queue in HR status and is submitted at the appropriate interval. For example, when **startt=(Monday,2:00)**, the Process runs each Monday at 2:00 AM. When **startt=(,1:00)**, the Process runs daily at 1:00 AM. IBM Connect:Direct does not provide a way to run a Process hourly. To do this, you must use the UNIX cron utility. If no start time is identified, you must issue a **change process** command to release the Process for execution. Do not code the **startt** parameter when you specify **retain=initial**.	yes | no | initial yes—Specifies that the system retains the Process in the Hold queue in HR status after execution. no—Specifies that the system deletes the Process from the TCQ after execution. This is the default. initial—Specifies that the system retains the Process in the Hold queue in HR status for automatic execution every time the Process Manager initializes.
sacct	Specifies accounting data for the SNODE. Setting this value in the submit statement overrides any accounting data specified in Process.	"*snode accounting data*" up to 256 characters. Enclose the string in double quotation marks.

Parameter	Description	Values
snode	Identifies the name of the secondary node. Setting this value overrides the snode value in the Process statement. The **snode** parameter is required either on the submit command or Process statement.	*name* | *host name* | *nnn.nnn.nnn.nnn* or nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn[;*port name* | *nnnnn*]] name—Specifies the node name of the remote node. The secondary node name corresponds to an entry in the network map file. host name—Specifies the name of the host computer where the remote IBM Connect:Direct node is running. nnn.nnn.nnn.nnn or nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn —Specifies the IP address of the remote node in IPv4 or IPv6 format: nnn.nnn.nnn.nnn (IPv4) or nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn (IPv6). [;port number |nnnnn]—Identifies the communications port. You can only use this parameter with the host name or IP address parameters. The nnnnn value is a decimal number from 1,024–65,535.

Parameter	Description	Values
snodeid	Specifies security user IDs and security passwords on the SNODE. The snodeid subparameters can contain one or more alphanumeric characters. If IBM Connect:Direct finds that a Process has no **snodeid** parameter or defines a **snodeid** parameter and the initialization parameter **proxy.attempt** is set to **y**, then any password specified on the **snodeid** parameter is ignored. A proxy user record is a remote user record in the userfile.cfg, which corresponds to the user name specified on the snodeid parameter. If no proxy user record exists, the **snodeid** parameter must contain a valid user name and password for a UNIX user who has a corresponding local user record in the userfile.cfg file. When **proxy.attempt=n** and no snodeid is defined, IBM Connect:Direct uses the submitting ID and node to find a Remote User Information record in the User Authorization Information file. If IBM Connect:Direct cannot find a match, then that user cannot send or receive files. If the initialization parameters file parameter **proxy.attempt** is set to **y**, users are not required to specify a password for the **snodeid** parameter. This capability enables the id subparameter to contain a dummy user ID to be used for translation to a local user ID on the remote system. The use of a dummy user ID offers improved security because neither the sender nor the receiver are required to use an actual user ID. Reserved keywords cannot be used in the **snodeid** field.	*id* [,*pswd* [,*newpswd*]] id—Specifies a user ID on the SNODE. pswd—Specifies a user password on the SNODE. If you specify id, you do not have to specify pswd. This capability enables the id parameter to contain a dummy ID to be used for translation to a local ID on the remote system. newpswd—Specifies a new password value. On certain platforms, the user password changes to the new value on the SNODE if the user ID and old password are correct (refer to documentation on the specific platform). If the SNODE is a UNIX node, the password does not change. If you specify pswd, you must also specify id. If you specify newpswd, you must also specify pswd. Type the values in the order of id, pswd, and newpswd.

Parameter	Description	Values
startt	Identifies the date, day, and time to start the Process. IBM Connect:Direct places the Process in the Timer queue in WS (Waiting for Start Time) status. The date, day, and time are positional parameters. If you do not specify date or day, a comma must precede time. Do not code the **startt** parameter when you **specify retain=initial**.	[*date* | *day* | *daily*] [,*hh:mm:ss* [am | pm]] date—Specifies the day (dd), month (mm), and year (yy), which you can code as mm/dd/yyyy or mm-dd-yyyy. If you only specify date, the time defaults to 00:00:00, which indicates midnight. The current date is the default. day—Specifies the day of the week. Values are today, tomorrow, yesterday, monday, tuesday, wednesday, thursday, friday, saturday, and sunday. hh:mm:ss [am | pm]—Specifies the time of day in hours (hh), minutes (mm), and seconds (ss). You can specify the hour in either 12- or 24-hour format. If you use 12-hour format, then you must specify am or pm. The default is the 24-hour format. The default value is 00:00:00, which indicates midnight. If you specify only the day value, the time defaults to 00:00:00. This means that if you submit a Process on Monday, with monday as the only startt parameter, the Process does not run until the following Monday at midnight. daily - Schedules the process for daily execution at the specified time. If the specified time has already elapsed on the day the command is submitted, the process will be scheduled for the next day. If you leave this parameter blank, the process will be scheduled for daily execution. The only difference between the daily specification and leaving the parameter blank is that if the specified time has already elapsed on the day the command is submitted, the process will be submitted immediately, and then daily at the specified time thereafter.
&symbolic name 1 &symbolic name 2 &symbolic name n	Specifies a symbolic parameter assigned a value. The value is substituted within the Process when the symbolic parameter is encountered. The value for the symbolic parameter must be in double quotation marks if it is a keyword or contains special characters. If you want to reserve the double quotation marks when the symbolic name is resolved in the Process, enclose the double-quoted string in single quotes, for example: &filename = "'filename with spaces'" The symbolic name itself must not be a subset of any other symbolic name. (You cannot have, for example, a symbolic name called &param and another symbolic name called &parameter in the same Process.)	variable string 1 variable string 2 variable string n The symbolic name cannot exceed 32 characters.

Parameter	Description	Values
tracel	Specifies the level of trace to perform for a Process. Tracing by Process can be turned on in the submit command or as part of the Process definition.	level = 0 |1 | 2 | 4 file=name level—Specifies the level of detail displayed in the trace output. The default is **4**. • 0—Terminates the trace. • 1—The basic level that provides function entry and function exit. • 2—includes level 1 plus function arguments. • 4—Enables a full trace. Basic diagnostic information, such as values of internal data structures at key points in the execution flow, are displayed. file—Specifies the name of a file where the trace output is directed. If you do not specify a file name, the file is created in the IBM Connect:Direct working directory with the file name CMGR.TRC. The length of the name value is unlimited.

Example - Submit a Process That Runs Every Week

The following command submits the Process named payroll:

```
submit file=payroll retain=yes startt=monday pacct="1959,dept-27";
```

Because **retain=yes** is specified in this sample, the Process is retained in the TCQ after execution. The Process starts next Monday at 00:00:00 and runs every Monday thereafter. Process accounting data is specified for the PNODE.

Example - Submit a Process with a Start Time Specified

The following command submits the Process named copyfil:

```
submit file=copyfil snode=vmcent startt=(01/01/2008, 11:45:00 am);
```

Because **startt** is specified, the Process executes on the first day of January 2008 at 11:45 a.m.

Example - Submit a Process with No File Value

The following command submits a Process without a **file** parameter value, but with the Process statements typed at the CLI command prompt:

```
Direct> sub do_copy process  snode=node1
step01 copy from (
                     file=data.data
                     pnode
                 )
            to   (
                     file=b
                     snode
                 )
      pend ;
Process Submitted, Process Number = 5
```

Example - Submit a Process and Turn On Tracing

The following command submits the Process named copy.cdp:

```
submit file=copy.cdp tracel=4;
```

Because **tracel** is specified, an SMGR and COMM full trace is performed on the Process. Trace information is written to the default file SMGR.TRC.

Changing Process Parameters

The **change process** command modifies specified parameters for a nonexecuting Process.

You specify the Processes to be changed by Process name, Process number, secondary node name, and submitter.

You can change the class, destination node, and priority. You can place a Process on the Hold queue or release a Process from the Hold queue by issuing a change process command with either the **release** or **hold=no** parameter.

If you submit a Process with a **startt** parameter, IBM Connect:Direct places the Process on the Timer queue. If a Process fails, you can move it to the Hold queue by specifying the change process command with hold=yes. IBM Connect:Direct then places the Process in the Hold queue in HO status. You can release the Process for execution at a later time.

You can set tracing for an existing Process by setting the **tracel** parameter to 1, 2, or 4. You can turn off tracing for a Process by setting trace1 to 0.

Specify at least one of the following search criteria parameters:

Parameter	Description	Value
pname	Locate the Process to be changed by Process name. The Process name is limited to 8 characters on Connect:Direct for Microsoft Windows and Connect:Direct for z/OS.	*name* | *generic* | (*list*) name—Specifies the Process name, up to 8 alphanumeric characters. generic—Specifies a nonspecific value for the Process name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more pname strings. list—Specifies a list of Process names. Enclose the list in parentheses, and separate each value with a comma.
pnumber	Locate the Process to be changed by Process number. IBM Connect:Direct assigns the Process number when the Process is submitted.	*number from 1–99,999* | (*list*) number—Specifies the Process number. list—Specifies a list of Process numbers. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
snode	Locate the Process to be changed by the secondary node name. This parameter can be used to specify a specific remote node, a generic value for matching remote node names (using pattern matching), or a list of multiple remote node names. The secondary node name typically contains the 1–16 character remote IBM Connect:Direct node name, but can be any string up to 256 alphanumeric characters long. You can also specify a remote node name as an IP address or hostname and a port number.	*remote node specification* | *generic* | (*list*) remote node specification—Identifies a specific remote node name. generic—Specifies a nonspecific value for the remote node name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more remote node names. list—Specifies a list of remote node specifications. Enclose the list in parentheses, and separate each value with a comma.
submitter	Locate the Processes to be changed by the node specification (the IBM Connect:Direct node name) and user ID of the Process owner. The character length of this parameter is unlimited.	(*node specification, userid*) | *generic* | (*list*) node specification, userid—Specifies the node specification (the IBM Connect:Direct node name) and user ID. generic—Specifies a nonspecific value for node specification and user ID. This generic value, containing pattern-matching characters, evaluates to a list of zero or more node specifications and user IDs. list—Specifies a list of node specification and user ID pairs. Enclose the list in parentheses, and separate each value with a comma.

The optional parameters for the **change process** command are the following:

Parameter	Description	Value
class	Changes the node-to-node session on which a Process can execute. A Process can execute on the class specified or any higher session class. The default class is specified as the **sess.default** parameter of the local.node record in the initialization parameters file.	The default is **1**.

Parameter	**Description**	**Value**
hold	Moves the Process to the Hold or Wait queue.	yes | no | call yes—Places the Process in the Hold queue in HO status until it is released by another **change process** command. no—Places the Process in the Wait queue in WC (Waiting for Connection) status; the Process executes as soon as resources are available. This is the default. call—Places the Process in the Hold queue in HC (Hold for Call) status until the remote node (SNODE) connects to the local node (PNODE) or another Process is submitted. At that time, IBM Connect:Direct releases the Process for execution
newsnode	Specifies a new remote node name to assign to the Process.	new remote node specification
prty	Changes the priority of the Process on the TCQ. IBM Connect:Direct uses the **prty** parameter for Process selection. A Process with a higher priority is selected for execution before a Process with a lower priority. The **prty** value does not affect the priority during transmission.	1–15, where 15 is the highest priority. If you do not specify **prty**, the default is **10**.
release	Releases the Process from a held state. This parameter is equivalent to **hold=no**.	none
tracel	Changes the level of trace to perform for a Process. If you identify the SNODE or PNODE immediately after the trace level definition, the trace level is turned on for all Processes submitted to and from the node identified.	level = 0 | 1 | 2 | 4 level—Specifies the level of detail displayed in the trace output. The default is **4**. 0—Terminates the trace. 1—Is the basic level that provides function entry and function exit. 2 —Includes level 1 plus function arguments. 4—Enables a full trace. Basic diagnostic information, such as values of internal data structures at key points in the execution flow, are displayed.

The following command changes the remote node name for the Process named cdproc to a new remote node, paris:

```
change process pname=cdproc newsnode=paris;
```

Deleting a Process from the TCQ

The **delete** process command removes a nonexecuting Process from the TCQ.

You select the Process to **delete** by Process name, Process number, secondary node name, submitter, or any combination of the search criteria parameters. Specify at least one of the following search criteria parameters:

Parameter	Description	Value
pname	Identify the Process to delete by Process name. The Process name is limited to 8 characters on Connect:Direct for Microsoft Windows and for z/OS.	*name* | *generic* | (*list*) name—Specifies the Process name up to 8 alphanumeric characters long. generic—Specifies a nonspecific value for the Process name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more pname strings. list—Specifies a list of Process names. Enclose the list in parentheses, and separate each value with a comma.
pnumber	Identify the Process to delete by Process number. IBM Connect:Direct assigns the Process number when the Process is submitted. Valid Process numbers range from 1–99,999.	*number* | (*list*) number—Specifies the Process number. list—Specifies a list of Process numbers. Enclose the list in parentheses, and separate each value with a comma (,).
snode	Identify the Process to delete by the secondary node name. This parameter can be used to specify a specific remote node, a generic value for matching remote node names (using pattern matching), or a list of multiple remote node names. The secondary node name typically contains the 1–16 character remote IBM Connect:Direct node name, but can be any string up to 256 alphanumeric characters long. You can also specify a remote node name as an IP address or hostname and a port number.	*remote node specification* | *generic* | (*list*) remote node specification—Identifies a specific remote node name. generic—Specifies a nonspecific value for the remote node name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more remote node names. list—Specifies a list of remote node specifications. Enclose the list in parentheses, and separate each value with a comma.
submitter	Identify Processes to delete by the node specification and user ID of the Process owner. The character length of this parameter is unlimited.	(*node specification, userid*) | *generic* | (*list*) node specification, userid—Specifies the node specification and user ID. generic—Specifies a nonspecific value for node specification and user ID. This generic value, containing pattern-matching characters, evaluates to a list of zero or more node specifications and user IDs. list—Specifies a list of node specification and user ID pairs. Enclose the list in parentheses, and separate each value with a comma.

The following command deletes all nonexecuting Processes submitted by user ID cduser on node dallas:

```
delete process submitter=(dallas, cduser);
```

Removing a Process from the Execution Queue

The **flush process** command removes Processes from the Execution queue. You select the Process to remove by Process name, Process number, secondary node name, submitter, or any combination of the search criteria parameters. Specify at least one of the following search criteria parameters:

Parameter	Description	Value
pname	Locate the Process to remove by Process name. The Process name is limited to 8 characters on Connect:Direct for Microsoft Windows and Connect:Direct for z/OS.	*name* | *generic* | (*list*) name—Specifies the Process name, up to 8 alphanumeric characters. generic—Specifies a nonspecific value for the Process name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more pname strings. list—Specifies a list of Process names. Enclose the list in parentheses, and separate each value with a comma.
pnumber	Locate the Process to remove by Process number. IBM Connect:Direct assigns the Process number when the Process is submitted.	*number from 1–99,999* | (*list*) number—Specifies the Process number. list—Specifies a list of Process numbers. Enclose the list in parentheses, and separate each value with a comma.
snode	Locate the Process to remove by the secondary node name. This parameter can be used to specify a specific remote node, a generic value for matching remote node names (using pattern matching), or a list of multiple remote node names. The secondary node name typically contains the 1–16 character remote IBM Connect:Direct node name, but can be any string up to 256 alphanumeric characters long. You can also specify a remote node name as an IP address or hostname and a port number.	*remote node specification* | *generic* | (*list*) remote node specification—Identifies a specific remote node name. generic—Specifies a nonspecific value for the remote node name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more remote node names. list—Specifies a list of remote node specifications. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
submitter	Locate the Processes to remove by the node specification (the IBM Connect:Direct node name) and user ID of the Process owner.	(*node specification, userid*) | *generic* | (*list*) node specification, userid—Specifies the node specification (the IBM Connect:Direct node name) and user ID. generic—Specifies a nonspecific value for node specification and user ID. This generic value, containing pattern-matching characters, evaluates to a list of zero or more node specifications and user IDs. list—Specifies a list of node specification and user ID pairs. Enclose the list in parentheses, and separate each value with a comma.

The flush process command has the following optional parameters:

Parameter	Description	Value
force	Forcibly terminates an executing Process or terminates a Process in an orderly fashion as the step completes. This parameter is useful if a Process is in the executing state and waiting for unavailable resources.	yes | <u>no</u> yes—Specifies to forcibly and immediately terminate the Process. The SMGR also terminates immediately. no—Specifies to terminate the Process in an orderly fashion as the step completes. The SMGR closes the statistics file and then terminates. This is the default.
hold	Places the terminated Process in the Hold queue where it can be released for re-execution.	yes | <u>no</u> yes—Specifies to place the Process in the Hold queue in HS status after the Process is terminated. no—Specifies to delete the Process from the TCQ after the Process is terminated. This is the default.

The following command flushes all executing Processes named “Rome” from the Execution queue:

```
flush process pname=rome force=yes;
```

The following command flushes all executing Processes on node alma submitted by user ID jones:

```
flush process submitter=(alma, jones);
```

Stopping IBM Connect:Direct

The **stop** command initiates an orderly IBM Connect:Direct shutdown sequence or forcibly terminates the software. After you run the stop command, no new Processes are allowed to run and no new connections with remote systems are established. Commands can be issued and users can sign on until the server terminates.

You can specify the force, immediate, quiesce, or step parameters with the stop command.

Following are the parameters for the stop command:

Parameter	Description
force	Forcibly terminates IBM Connect:Direct and returns control to the operating system.

Parameter	Description
immediate	Begins an immediate, but orderly shutdown of all activity and terminates IBM Connect:Direct. The software terminates connections, writes statistics records, closes files, and shuts down.
quiesce	Runs all executing Processes to completion before shutting down IBM Connect:Direct. No new Processes are started. This is the default value.
step	Shuts down IBM Connect:Direct after all currently executing Process steps are complete. The software writes statistics records, closes files, and shuts down. All active Processes are retained in the TCQ. Processes restart when the software is re-initialized.

The following command forcibly terminates IBM Connect:Direct and returns control to the operating system:

```
stop force;
```

Viewing a Process in the TCQ

The **view process** command is used to view Processes in the TCQ when the local node is the PNODE. You can search by Process name, Process number, queue, secondary node, status, owner of the Process, or any combination of the search criteria parameters.

You also can specify more than one Process in the search criteria.

There are no required parameters for this command. If you do not specify an optional parameter, IBM Connect:Direct selects all Processes executing or waiting for execution. Following are the optional parameters for the view process command:

Parameter	Description	Value
pname	Locate the Process to view by Process name. The Process name is limited to 8 characters on Connect:Direct for Microsoft Windows and Connect:Direct for z/OS.	*name* | *generic* | (*list*) name—Specifies the Process name, up to 8 alphanumeric characters. generic—Specifies a nonspecific value for the Process name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more pname strings. list—Specifies a list of Process names. Enclose the list in parentheses, and separate each value with a comma.
pnumber	Locate the Process to view by Process number. IBM Connect:Direct assigns the Process number when the Process is submitted.	*number from 1–99,999* | (*list*) number—Specifies the Process number. list—Specifies a list of Process numbers. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
queue	Specifies the Processes to be viewed by the specified queue names.	all | exec | hold | wait | timer all—Selects Processes from all queues. This is the default. exec—Selects Processes from the Execution queue. hold—Selects Processes from the Hold queue. timer—Selects Processes from the Timer queue. wait—Selects Processes from the Wait queue.
snode	View the Process by the secondary node name. This parameter can be used to specify a specific remote node, a generic value for matching remote node names (using pattern matching), or a list of multiple remote node names. The secondary node name typically contains the 1–16 character remote IBM Connect:Direct node name, but can be any string up to 256 alphanumeric characters long. You can also specify a remote node name as an IP address or hostname and a port number.	*remote node specification* | *generic* | (*list*) remote node specification—Identifies a specific remote node name. generic—Specifies a nonspecific value for the remote node name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more remote node names. list—Specifies a list of remote node specifications. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
status	Specifies the Processes to be viewed by Process status. If you do not specify a status value, information is generated for all status values.	EX | HC | HE | HI | HO | HR | HS | PE | WC | WR | WS | (list) EX (Execution)—Specifies to select Processes from the Execution queue. HC (Held for Call)—Specifies to select Processes submitted with hold=call. HE (Held due to Error)—Specifies to select Processes held due to a connection error. HI (Held Initially)—Specifies to select Processes submitted with hold=yes. HO (Held by Operator)—Specifies to select Processes held by a **change process** command issued with hold=yes. HR (Held Retain)—Specifies to select Processes submitted with retain=yes or retain=initial. HS (Held Due to Execution Suspension)—Specifies to select Processes suspended by a **flush process** command issued with hold=yes. PE (Pending Execution)—Specifies to select Processes submitted with a **maxdelay** parameter and assigned PE status by the Process Manager just before a Session Manager is created to execute the Process. After the Session Manager initializes, the Process is placed on the Execution queue and the status is changed to EX. WC (Waiting for Connection)—Specifies to select Processes that are ready for execution, but that all available connections to the remote node are in use. WR (Waiting for Restart)—Specifies to select Processes that are waiting for restart after session failure. WS (Waiting for Start Time)—Specifies to select Processes waiting for a start time. These Processes are on the Timer Queue. list—Specifies a list of status values. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
submitter	Locate the Processes to view by the node specification (the IBM Connect:Direct node name) and user ID of the Process owner. The length of this parameter is unlimited.	(*node specification, userid*) | *generic* | (*list*) node specification, userid—Specifies the node specification (the IBM Connect:Direct node name) and user ID. generic—Specifies a nonspecific value for node specification and user ID. This generic value, containing pattern-matching characters, evaluates to a list of zero or more node specifications and user IDs. list—Specifies a list of node specification and user ID pairs. Enclose the list in parentheses, and separate each value with a comma.

The following command displays the specified Process number:

```
view process pnumber=1;
```

Monitoring Process Status in the TCQ

The select process command displays information about Processes in the TCQ.

The search criteria provide flexibility in selecting Processes. You can search for a Process by Process name, Process number, queue, secondary node, status, owner of the Process, or any combination of the search criteria parameters.

You also can specify more than one Process in the search criteria. You can request either a detailed report about the selected Process or a short report.

There are no required parameters for this command. If you do not specify an optional parameter, IBM Connect:Direct selects all Processes executing or waiting for execution. Following are the optional parameters for the **select process** command:

Parameter	Description	Value
pname	Locate the Process to select by Process name. The Process name is limited to 8 characters on Connect:Direct for Microsoft Windows and Connect:Direct for z/OS.	*name* | *generic* | (*list*) name—Specifies the Process name, up to 8 alphanumeric characters. generic—Specifies a nonspecific value for the Process name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more pname strings. list—Specifies a list of Process names. Enclose the list in parentheses, and separate each value with a comma.
pnumber	Locate the Process to select by Process number. IBM Connect:Direct assigns the Process number when the Process is submitted.	*number from 1–99,999* | (*list*) number—Specifies the Process number. list—Specifies a list of Process numbers. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
queue	Specifies the Processes to be selected by the specified queue names. The default is **all**.	<u>all</u> | exec | hold | wait | timer all—Selects Processes from all queues. this is the default. exec—Selects Processes from the Execution queue. hold—Selects Processes from the Hold queue. timer—Selects Processes from the Timer queue. wait—Selects Processes from the Wait queue.
snode	Locate the Process by the secondary node name. This parameter can be used to specify a specific remote node, a generic value for matching remote node names (using pattern matching), or a list of multiple remote node names. The secondary node name typically contains the 1–16 character remote IBM Connect:Direct node name, but can be any string up to 256 alphanumeric characters long. You can also specify a remote node name as an IP address or hostname and a port number.	*remote node specification* | *generic* | (*list*) remote node specification—Identifies a specific remote node name. generic—Specifies a nonspecific value for the remote node name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more remote node names. list—Specifies a list of remote node specifications. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
status	Specifies the Processes to be selected by Process status. If you do not specify a status value, information is generated for all status values.	EX | HC | HE | HI | HO | HR | HS | PE | WC | WR | WS | (*list*) EX (Execution)—Specifies to select Processes from the Execution queue. HC (Held for Call)—Specifies to select Processes submitted with hold=call. HE (Held due to Error)—Specifies to select Processes held due to a connection error. HI (Held Initially)—Specifies to select Processes submitted with hold=yes. HO (Held by Operator)—Specifies to select Processes held by a **change process** command issued with hold=yes. HR (Held Retain)—Specifies to select Processes submitted with retain=yes or retain=initial. HS (Held Due to Execution Suspension)—Specifies to select Processes suspended by a flush process command issued with hold=yes. PE (Pending Execution)—Specifies to select Processes submitted with a **maxdelay** parameter and assigned PE status by the Process Manager just before a Session Manager is created to execute the Process. After the Session Manager initializes, the Process is placed on the Execution queue and the status is changed to EX. WC (Waiting for Connection)—Specifies to select Processes that are ready for execution, but that all available connections to the remote node are in use. WR (Waiting for Restart)—Specifies to select Processes that are waiting for restart after session failure. WS (Waiting for Start Time)—Specifies to select Processes waiting for a start time. These Processes are on the Timer Queue. list—Specifies a list of status values. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
submitter	Locate the Processes to select by the node specification (the IBM Connect:Direct node name) and user ID of the Process owner. The length of this parameter is unlimited.	(*node specification, userid*) | *generic* | (*list*) node specification, userid—Specifies the node specification (the IBM Connect:Direct node name) and user ID. generic—Specifies a nonspecific value for node specification and user ID. This generic value, containing pattern-matching characters, evaluates to a list of zero or more node specifications and user IDs. list—Specifies a list of node specification and user ID pairs. Enclose the list in parentheses, and separate each value with a comma.
detail	Specifies the type of report (short or detailed) that IBM Connect:Direct generates for the selected Processes.	yes | <u>no</u> yes—Generates a detailed report. no—Generates a short report. This is the default.

The following command displays a short report for the specified Process number:

```
select process pnumber=9 detail=no;
```

Output from the command is displayed in the following table:

```
==================================================================================
                            SELECT  PROCESS
==================================================================================
PROCESS NAME    NUMBER        USER    SUBMITTER NODE           QUEUE          STATUS
----------------------------------------------------------------------------------
PR01                   9           root    cd.unix.pj            EXEC                EX
==================================================================================
```

The following command displays a detailed report for the specified Process number:

```
select process pnumber=9 detail=yes;
```

Output from the command is displayed in the following table:

```
==================================================================================
                                                    SELECT PROCESS
==================================================================================
Process Name    => pr01           Class         => 9
Process Number  => 9              Priority      => 8
Submitter Node  => cd.unix.pj     PNODE         => cd.unix.pj
Submitter       => sub1           SNODE         => cd.unix.pj
Retain Process  => no             Header Type   => p

Submit Time     => 19:52:35       Schedule Time =>
Submit Date     => 05/22/1996     Schedule Date =>

Queue           => EXEC
Process Status  => EX
Message Text    =>
----------------------------------------------------------------------------------
```

Determining the Outcome of a Process

The **select statistics** command is used to examine Process statistics from the Connect:Direct statistics file. The type of information in the statistics report includes copy status and execution events.

The search criteria provide flexibility in selecting information you want to display. The parameters used with the select statistics command determine search criteria and the form in which the information is presented. You can specify records to select by condition code, Process name, Process number, identification type, category, secondary node, start time, stop time, and submitter node specification and user ID.

There are no required parameters for this command. If you do not indicate a search requirement with an optional parameter, Connect:Direct selects all statistics records; however, the volume of records can be excessive. Following are parameters for the select statistics command:

Parameter	**Description**	**Value**
ccode	Selects statistics records based on the completion code operator and return code values associated with Step Termination. You must specify the return code.	operator, *nn* operator—Specifies the completion code operator. Following are the valid completion code operators: <u>eq</u> or = or == (equal) This is the default. ge or >= or => (greater than or equal) gt or > (greater than) le or <= or =< (less than or equal) lt or < (less than) ne or != (not equal) The return code is the exit status of the UNIX command or the Connect:Direct Process or command. nn—Specifies the return code value associated with Step Termination.
destfile	Selects statistics based on a destination file name. This parameter can be abbreviated as dest.	dest=*/path/file name* For example: sel stat dest=/sci/payroll/june.payroll; This parameter can be used in combination with the srcfile parameter to select statistics based on a source file name and a destination file name, for example: sel stat srcf=/sci/accounting/june.payroll dest=/sci/payroll/june.payroll
pname	Locate the statistics to select by Process name. The Process name is limited to 8 characters on Connect:Direct for Microsoft Windows and Connect:Direct for z/OS.	*name* | *generic* | (*list*) name—Specifies the Process name, up to 8 alphanumeric characters. generic—Specifies a nonspecific value for the Process name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more pname strings. list—Specifies a list of Process names. Enclose the list in parentheses, and separate each value with a comma.
pnumber	Locate the statistics to select by Process number. Connect:Direct assigns the Process number when the Process is submitted.	*number from 1–99,999* | (*list*) number—Specifies the Process number. list—Specifies a list of Process numbers. Enclose the list in parentheses, and separate each value with a comma.

Parameter	Description	Value
reccat	Specifies whether the selection of statistics file records is based on events or related to a Process or is from an external source.	CAEV | CAPR | CAEX (CAEV, CAPR, CAEX) CAEV—Specifies that the selection of statistics file records is related to an event, such as a IBM Connect:Direct shutdown. CAPR—Specifies that the selection of statistics file records is related to one or more IBM Connect:Direct Processes. CAEX—Specifies that the selection of statistics file records is related to an external source like Integrated File Agent.

Parameter	Description	Value
recids	Specifies the statistics file records to be selected by record ID. This parameter identifies particular types of statistics records, such as a copy termination records or Connect:Direct initialization event records.	*record id* | (*list*) record id—Selects statistics file records for the specified record ID. list—Specifies a list of Process names. Enclose the list in parentheses, and separate each value with a comma. Some of the most common Rec ID (Identifiers) are listed below. Other Rec IDs are typically derived from the first 4 letters of a message ID. The message text will provide the necessary details.

Record ID	Category	Description
CMOT	Event	Client manager comm termination
CSPA	Process	Secure+
CSPE	Event	Strong Password Encryption
CTRC	Process	Copy Termination Record
EXFA	External Source	EXFA- External Integrated File Agent
FMSD, FMRV	Process	FMH sent / received
IFED, RJED, RTED, SBED, PSED	Process	Step ended for IF / RUN TASK / RUN JOB / SUBMIT / other
LSMG	Process	Session Manager
LSST, RSST	Process	Local / remote step started
NUIC, NUIS	Event	Server Startup
NUTR, NUT1, NUT2, NUTC, NUIS	Event	Server Shutdown
PMIP, PMST, PMED	Event	Process manager initializing / started / ended
PMMX	Event	TCQ max age processing
PSTR, PRED, PERR, PRIN, PFLS, PSAV	Event	Process started / ended / error / interrupted / flushed / saved
QCxx	Event	TCQ change (xx typically identifies the new queue)
SCNT	Event	Concurrent session count

Parameter	Description	Value
snode	Locate the statistics file record by the secondary node name. This parameter can be used to specify a specific remote node, a generic value for matching remote node names (using pattern matching), or a list of multiple remote node names. The secondary node name typically contains the 1–16 character remote Connect:Direct node name, but can be any string up to 256 alphanumeric characters long. You can also specify a remote node name as an IP address or hostname and a port number.	*remote node specification* | *generic* | (*list*) remote node specification—Identifies a specific remote node name. generic—Specifies a nonspecific value for the remote node name. This generic value, containing pattern-matching characters, evaluates to a list of zero or more remote node names. list—Specifies a list of remote node specifications. Enclose the list in parentheses, and separate each value with a comma.
srcfile	Selects statistics based on a source file name. This parameter can be abbreviated as srcf.	srcf=*/path/file name* For example: sel stat srcf=/sci/accounting/june.payroll; This parameter can be used in combination with the destfile parameter to select statistics based on a source file name and a destination file name, for example: sel stat srcf=/sci/accounting/june.payroll dest=/sci/payroll/june.payroll
startt	Selects records produced both at and since the specified time. The date or day and time are positional. If you do not specify date or day, a comma must precede time.	[*date* | *day*] [, *hh:mm:ss* [am|pm]] date—Specifies the day (dd), month (mm), and year (yy), which you can code as mm/dd/yyyy or mm-dd-yyyy. If you only specify date, the time defaults to 00:00:00. The current date is the default. day—Specifies the day of the week. Values are today, monday, tuesday, wednesday, thursday, friday, saturday, and sunday. hh:mm:ss [am | pm]—Specifies the time of day in hours (hh), minutes (mm), and seconds (ss). You can specify the hour in either 12- or 24-hour format. If you use 12-hour format, then you must specify am or pm. The default is the 24-hour format. The default value is **00:00:00**, which indicates midnight. If you specify only the day value, the time defaults to 00:00:00.

Parameter	Description	Value
stopt	Specifies that Connect:Direct searches for statistics records up to and including the designated date, day, and time positional parameters. If you do not specify date or day, a comma must precede time.	[*date* | *day*] [, hh:mm:ss [am|pm]] date—Specifies the day (dd), month (mm), and year (yy), which you can code as mm/dd/yyyy or mm-dd-yyyy. If you only specify date, the time defaults to 00:00:00. The current date is the default. day—Specifies the day of the week. Values are today, monday, tuesday, wednesday, thursday, friday, saturday, and sunday. hh:mm:ss [am | pm]—Specifies the time of day in hours (hh), minutes (mm), and seconds (ss). You can specify the hour in either 12- or 24-hour format. If you use 12-hour format, then you must specify am or pm. The default is the 24-hour format. The default value is **00:00:00**, which indicates midnight. If you specify only the day value, the time defaults to 00:00:00.
submitter	Locate the statistics records to select by the node specification (the Connect:Direct node name) and user ID of the Process owner. The character length of the parameter is unlimited.	(*node specification, userid*) | *generic* | (*list*) node specification, userid—Specifies the node specification (the Connect:Direct node name) and user ID. generic—Specifies a nonspecific value for node specification and user ID. This generic value, containing pattern-matching characters, evaluates to a list of zero or more node specifications and user IDs. list—Specifies a list of node specification and user ID pairs. Enclose the list in parentheses, and separate each value with a comma.
detail	Specifies the type of report (short or detailed) that Connect:Direct generates for the selected Processes.	yes | <u>no</u> yes—Generates a detailed report. no—Generates a short report. This is the default.

Generating a Detailed Output Report for a Process

You can use the **select statistics** command to generate a detailed report for a Process. The following command generates a detailed report for Process number 9:

```
select statistics pnumber=9 detail=yes startt=(08/10/2008);
```

The report consists of all records from August 10, 2008.

A sample statistics output for two steps only is listed in the following section. Use the table of recids in "Determining the Outcome of a Process" on page 171 to interpret the Record ID. The Record ID can change for each Process step displayed. The completion code indicates whether the Process executed successfully or produced an error condition.

To display the long text of the message, issue the **ndmmsg** command.

Generating a Summary Report for a Process

You can use the **select statistics** command to generate a summary report for a Process. The following command generates summary statistics for Process number 9:

```
sel stat pnumber=9 detail=no startt=(08/10/2008);
```

The report consists of all records from August 10, 2008.

Sample output that describes all Process steps in summary form is displayed in the following table:

```
===================================================================================
                          SELECT STATISTICS
===================================================================================
P RECID LOG TIME             PNAME PNUMBER STEPNAME CCOD FDBK MSGID
-----------------------------------------------------------------------------------
P PSTR 08/10/2008 09:10:39   PR01  9                0         XSMG200I
P IFED 08/10/2008 09:10:44   PR01  9                0         XSMG405I
P CTRC 08/10/2008 09:10:44   PR01  9                0         XSMG405I
P IFED 08/10/2008 09:10:45   PR01  9                4         XSMG400I
P RTED 08/10/2008 09:10:45   PR01  9                0         XSMG400I
P IFED 08/10/2008 09:10:45   PR01  9                4         XSMG400I
P CTRC 08/10/2008 09:10:45   PR01  9                0         XSMG405I
P CTRC 08/10/2008 09:10:45   PR01  9                8         XSMG405I
P CTRC 08/10/2008 09:10:45   PR01  9                8         XSMG405I
===================================================================================
```

To avoid lengthy search times when issuing the **select statistics** command, archive or delete statistics files regularly. Also, use the **startt** and **stopt** parameters to bracket the desired stats as closely as possible. Execution of a Process generates multiple statistics records. IBM Connect:Direct closes the current statistics file and creates a new statistics file every midnight. 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 in the *d_dir/work/cd_node* directory. Names of the statistics file 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.

Running System Diagnostics

The diagnostic command, trace, enables you to run system diagnostics and troubleshoot operational problems. Use the **trace** command with the appropriate parameter listed in the following table to enable and disable runtime traces within the Process Manager, Command Manager, and Session Manager components of the software. For Session Manager traces, you can run a trace for a specific node.

The Command Manager trace is turned on immediately for the client that issued the trace command. After the trace command is issued, all clients that make connections are also traced. Session Manager traces go into effect immediately.

The **trace** command has the following parameters:

Parameter	Description	Value
cmgr	To trace the Command Manager.	level=0 |1 | 2 | 4 file=name level—Specifies the level of detail displayed in the trace output. The default is **4**. 0—Terminates the trace. 1—Is the basic level that provides function entry and function exit. 2—Includes level 1 plus function arguments. 4—Enables a full trace. Basic diagnostic information, such as values of internal data structures at key points in the execution flow, are displayed. file—Specifies the name of a file where the trace output is directed. If you do not specify a file name, the file is created in the IBM Connect:Direct working directory with the file name **CMGR.TRC**. The length of the name value is unlimited.
comm	To trace the data sent to and received from a remote IBM Connect:Direct system within the Session Manager. You can set this trace independently from or in conjunction with the smgr trace. If you run both the comm and smgr traces, trace output for both traces is directed to the file name of the trace last specified.	level=0 |1 | 2 | 4 file=name level—Specifies the level of detail displayed in the trace output. The default is **4**. 0—Terminates the trace. 1—Is the basic level that provides function entry and function exit. 2—Includes level 1 plus function arguments. 4—Enables a full trace that provides basic diagnostic information, such as values of internal data structures at key points in the execution flow. file—Specifies the name of a file where the trace output is directed. If you do not specify a file name, the file is created in the IBM Connect:Direct working directory with the file name COMM.TRC. The length of the name value is unlimited. The default file name is **COMM.TRC**.

Parameter	Description	Value
pmgr	To trace the Process Manager.	level=0 |1 | 2 | 4 file=name level—Specifies the level of detail displayed in the trace output. The default is **4**. 0—Terminates the trace. 1—Is the basic level that provides function entry and function exit. 2—Includes level 1 plus function arguments. 4—Enables a full trace that provides basic diagnostic information, such as values of internal data structures at key points in the execution flow. file—Specifies the name of a file where the trace output is directed. If you do not specify a file name, the file is created in the IBM Connect:Direct working directory with the file name **PMGR.TRC**. The length of the name value is unlimited.
smgr	To run the trace for Session Managers created after issuing the trace command. Currently executing Session Managers are unaffected. If you run both the comm and smgr traces, trace output for both traces is directed to the file name of the trace last specified.	level=0 |1 | 2 | 4 snode | pnode | tnode file=name level—Specifies the level of detail displayed in the trace output. The default is **4**. 0—Terminates the trace. 1—Is the basic level that provides function entry and function exit. 2—Includes level 1 plus function arguments. 4—Enables a full trace that provides basic diagnostic information, such as values of internal data structures at key points in the execution flow. snode—Specifies to trace only the SNODE SMGR. pnode—Specifies to trace only the PNODE SMGR. tnode—Identifies the node on which to perform the trace. If you want to gather trace information for more than one node, identify more than one node in this parameter. file—Specifies the name of a file where the trace output is directed. If you do not specify a file name, the file is created in the IBM Connect:Direct working directory with the file name SMGR.TRC. The length of the name value is unlimited. The default file name is **SMGR.TRC**.

The following sample trace command performs a level 2 trace on the Session Manager for the node called ath3500ry and writes the output to the file Smgp.trc:

```
trace smgr pnode tnode=ath3500ry level=2 file=Smgp.trc;
```

In case there is an issue in startup of cdpmgr itself, the diagnostics can be captured by enabling cdpmgr startup traces using the following command. Here d_dir is the installation directory where Connect:Direct for UNIX is installed, cd_node is the node name and output will be captured in cdpmgr_startup.trc file.

```
$ d_dir/ndm/bin/cdpmgr -i d_dir/ndm/cfg/cd_node/initparm.cfg -t 4 >> cdpmgr_startup.trc 2>&1
```

A partial sample trace output is illustrated in the following section. A trace identifies the Process ID and the function, the month and day, and the time in microseconds. The first column contains the Process ID. Column two indicates the month and day in the form of MM/DD. Column three indicates the time in the form of HH:MM:SSSS. The last column indicates the function. An arrow pointing to the right indicates the function was entered. An arrow pointing to the left indicates the function was exited. Some of the functions are indented, which indicates nesting. An indented arrow indicates that the function was called by the preceding function.

```
indicates that the function was called by the preceding function.
==================================================================================
498    05/18 15:13:0104   cm_sendcmd_1 entered.
498    05/18 15:13:0206    -> ndm_error_destroy
                         <- ndm_error_destroy: ok
498    05/18 15:13:0506    -> ndm_error_create
                         <- ndm_error_create: ok
498    05/18 15:13:0708    ndm_cmds_free entered.
                         ndm_cmds_free exited.
498    05/18 15:13:0801    ->ndm_parser_jdi
498    05/18 15:13:0806    -> ndm_error_create
                          <- ndm_error_create: ok
498    05/18 15:13:0916     ->Parser: SELPRO
498    05/18 15:13:0926      ->bldexp
                           <-bldexp: Null argument value,
                                don&apos;t add.
498    05/18 15:13:1116      ->bldexp
498    05/18 15:13:1136       -> ndm_crit_comp
498    05/18 15:13:1155        ->compile
                             <-compile
                            <- ndm_crit_comp: Handle
                           <-bldexp: ok
                        .
                        .
                        .
==================================================================================
```

Process Queuing

The TCQ controls Process execution as IBM Connect:Direct operates. After you submit a Process, it is stored in the TCQ. The TCQ consists of four queues: Execution, Wait, Timer, and Hold.

After you submit a Process, you can monitor the status, modify specific characteristics, and stop execution by using the appropriate commands. The commands listed in the following table allow you to perform these tasks:

Command	Definition
change process	Change the status and modify specific characteristics of a nonexecuting Process in the TCQ.
delete process	Remove a nonexecuting Process from the Wait, Timer, and Hold queues.
flush process	Remove an executing Process from the Execution queue.
select process	Monitor Processes in the TCQ, including those Processes that are executing.
view process	View Processes in the TCQ.

Scheduling IBM Connect:Direct Activity

IBM Connect:Direct places a Process in a queue based on the parameters that affect scheduling. You can specify scheduling parameters in the **Process** statement or the **submit** command.

Scheduling parameters are listed in the following section:

- retain=yes|no|initial
- hold=yes|no|call
- startt=[([date|day] [, hh:mm:ss | [am | pm]])

The following table shows how scheduling parameters affect the logical queues.

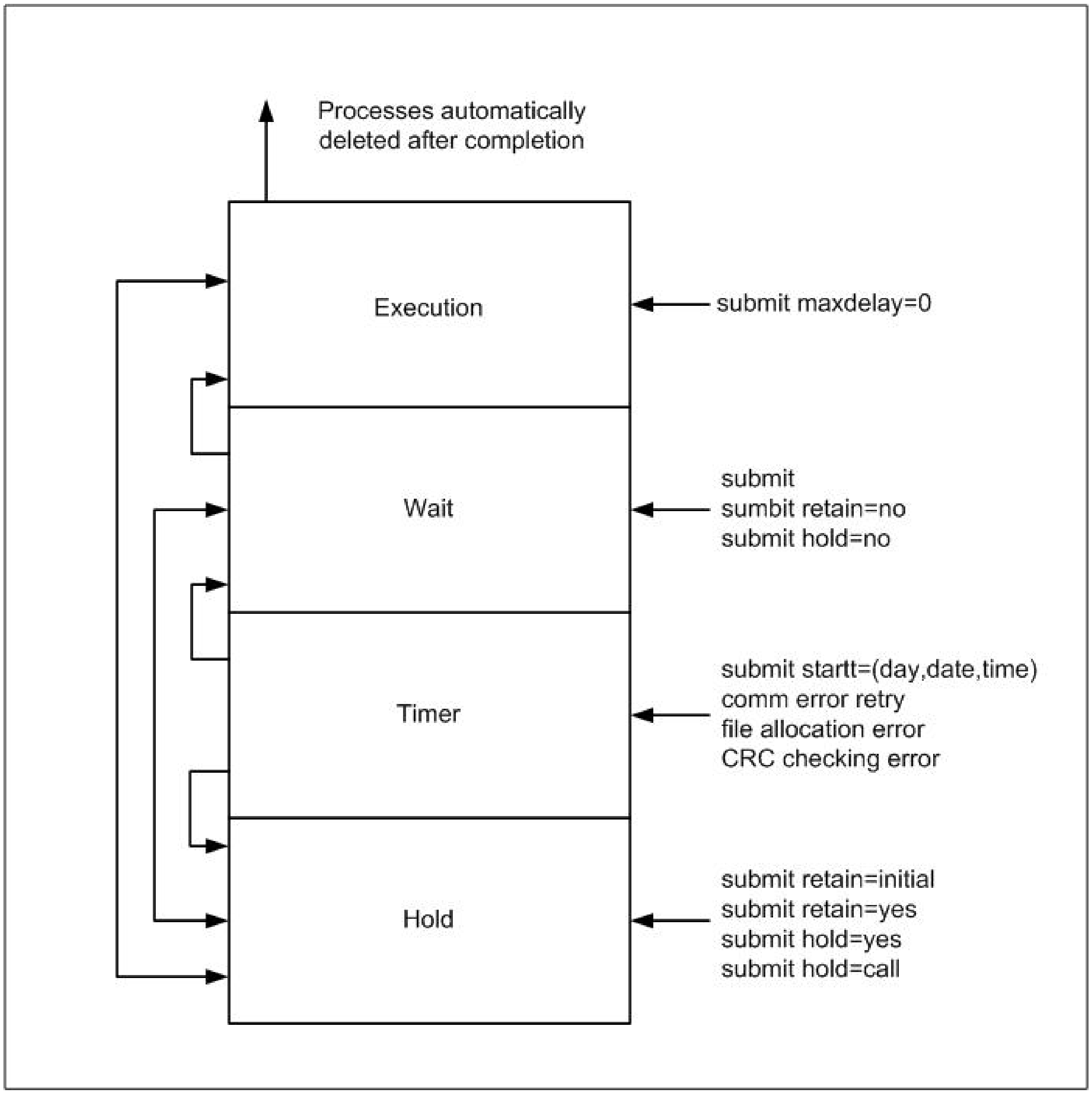
Scheduling Parameter	**Queue**	**Comments**
None of the scheduling parameters specified	Wait	The Process remains in the Wait queue until IBM Connect:Direct establishes a session with the remote node. After a session is established, the Process moves to the Execution queue.
retain=yes	Hold	A copy of the Process executes once, unless you specify a startt parameter value. Specify a day or time or both for the Process to start.
retain=no	Wait (if no other parameters are specified)	The Process remains in the Wait queue until IBM Connect:Direct establishes a session with the remote node. The default is **no**.
retain=initial	Hold	A copy of the Process remains in the Hold queue and executes every time the Process Manager is initiated.
retain=yes and hold=no or hold=call	Hold	A copy of the Process remains in the Hold queue to be executed when released.
hold=yes	Hold	You can execute the Process by specifying the **change process** command with the release parameter.
hold=no	Wait (if no other parameters are specified)	The default for hold is **no**.
hold=call	Hold	The Process remains in the queue until the remote node starts a session with the local node or another Process starts a session with that remote node.
startt	Timer	When the scheduled day or time occur, the Process is moved to the Wait queue.

Each Process in the TCQ has an associated status value. Each status value has a unique meaning that is affected by the logical queue in which the Process is placed. Status values for each queue are shown in the tables in the following sections. You can use the **select process** command to examine that status of Processes in the TCQ. For example, the following command displays all Processes in the TCQ with execution status:

```
select process status=EX;
```

Progression of a Process Through the TCQ

This section describes each logical queue of the TCQ and the progression of a Process through these queues. The following figure illustrates the four logical queues and their associated parameter values:



The Execution Queue

Processes are placed in the Execution queue after IBM Connect:Direct connects to the remote node. Processes normally come from the Wait queue, but also can be placed in the Execution queue by a submit command with maxdelay=0 specified.

Processes in the Execution queue can be in execution (EX) status or pending execution (PE) status. Processes with EX status are exchanging data between two IBM Connect:Direct nodes. Processes with PE status are waiting for Process start messages to be exchanged between the local node and the remote node. Processes usually have PE status assigned for a very short period of time.

After a Process successfully completes, it is automatically deleted from the Execution queue. A flush process command with hold=yes moves a Process from the Execution queue and places it in the Hold queue. When a session is interrupted, the Process moves from the Execution queue to the Timer queue if retry values are specified. If connection is not made before the retry values are exhausted or if retry values are not specified, the action taken depends on the **conn.retry.exhaust.action** parameter. By default, the Process moves to the Hold queue.

The following table shows the status values for the Execution queue:

Element	Comment
PE	Pending Execution is the initial queue status when a Process is submitted with maxdelay=0.
EX	Execution status indicates that the Process is executing.

The Wait Queue

Processes in the Wait queue are waiting for a new or existing connection to become available between the local node and the remote node.

Processes can come from the Hold queue or the Timer queue. Processes also can be placed in the Wait queue by a submit command with no parameters specified, submit with retain=no, or submit with hold=no.

After the connection is made, Processes automatically move to the Execution queue.

The following table shows the status values for the Wait queue:

Status	Comment
WC	This status indicates the Process is ready to execute as soon as possible, but no session is available. Other Processes may be executing with the SNODE, and no other sessions are available. This Process runs as soon as a new session is created or an existing session becomes available.
WR	This status indicates that the Process is in retry status. The number of retries and intervals between retries is specified in the network map.
WA	This status indicates the initial queue status when a Process is submitted without a hold or retain value. This Process is ready to execute as soon as possible.
WS	This status indicates that the Process is waiting for its scheduled time to execute.

The Timer Queue

Processes are placed in the Timer queue by a submit command with the startt parameter specified. Processes in the Wait for Start Time (WS) status are waiting for the start time to arrive before moving to the Wait queue. Processes also are placed in the Timer queue in Retry (WC) status if one of the following error conditions occur:

- If a file allocation error occurs when a Process is executing on either the local or the remote node, and the file allocation error is identified as a condition to retry, the Process is placed in the Timer queue. The Process is then retried using the short-term and long-term retry parameter definitions. This capability enables a Process that was unable to execute because a file that it called was unavailable to be retried at a later time.

- If a connection error occurs while a Process is executing, the intelligent session retry facility places all Processes scheduled for the node, including the executing Process, in the Timer queue. This capability eliminates the overhead required to retry each of the Processes on the node even though the connection is lost.
- If CRC checking is activated, a Process that generates a CRC error is placed in the Timer queue.

IBM Connect:Direct automatically tries to execute the Process again based on the number of times to retry and the delay between retries as specified in the network map parameters.

Processes move from the Timer queue to the Wait queue. A **change process** command with hold=yes specified moves the specified Process from the Timer queue to the Hold queue. The following table shows the status values for the Timer queue:

Status	Comment
WR	This status indicates that the Process is in retry status. The number of retries and intervals between retries is specified in the network map.
WS	This status indicates that the Process is waiting for its scheduled time to execute.
HR	Held Retain indicates that the Process was submitted with retain=yes or retain=initial specified and has already executed. The Process can be released later by a **change process** command with release specified.
WC	This status indicates the Process is ready to execute as soon as possible, but no session is available. Other Processes may be executing with the SNODE, and no other sessions are available. This Process runs as soon as a new session is created or an existing session becomes available.

The Hold Queue

Processes in the Hold queue are waiting for operator intervention before they progress to the Wait queue. This queue enables operators of the local node and remote node to coordinate and control Process execution.

Processes are placed in the Hold queue by a submit command with retain=initial, retain=yes, or hold=yes parameters specified. Processes submitted with hold=call also are placed in the Hold queue. Processes are moved from the Timer queue to the Hold queue by a **change process** command with hold=yes specified. Additionally, Processes are moved from the Execution queue to the Hold queue by a f**lush process** command with hold=yes specified.

Processes are moved from the Hold queue to the Execution queue by a **change process** command with the release parameter specified.

The following table shows the status values for the Hold queue:

Status	Comment
HC	Held for Call indicates that the Process was submitted with hold=call specified. A session started from either node causes the Process to be moved to the Wait queue in WC status. The Process is placed in the Execution queue when the Process is selected for execution.

Status	Comment
HI	Held Initially indicates that the Process was submitted with hold=yes specified. The Process can be released later by a **change process** command with release or hold=no specified.
HE	Held due to error specifies that a session error or other abnormal condition occurred.
HO	Held by Operator indicates that a change process hold=yes was specified.
HR	Held Retain indicates that the Process was submitted with retain=yes or retain=initial specified and has already executed. The Process can be released later by a **change process** command with release specified.
HS	Held for Suspension indicates that the operator issued a **flush process** command with hold=yes specified. The Process can be released later by a **change process** command with release specified.

IBM Connect:Direct Utilities

IBM Connect:Direct translates data from one character set code to a different character set code, such as from ASCII to EBCDIC, based on a character translation table in the *d_dir*/ndm/xlate directory. IBM Connect:Direct provides a default character translation table for use during file transfer operations or you can modify this table using the utility program called ndmxlt.

Creating a Translation Table

1. To create a translation table, either copy the file called /*cd_dir*/cdunix/ndm/src/def_send.sxlt or /*cd_dir*/cdunix/ndm/src/def_recv.sxlt, where *cd_dir* is the directory where IBM Connect:Direct is installed, and rename it or modify this file.
2. Use a text editor to add the new values to the table in the file you created.
3. Compile the updated file with the ndmxlt utility.
4. Replace the default translation table in the *d_dir*/ndm/xlate with the updated table. Each table is 256 bytes long.

```
Following is a sample translation table:
# This file contains an example of defining an ASCII-to-EBCDIC translation table and
# then changing it to translate lowercase to uppercase.
#
# Define the ASCII-to-EBCDIC table.
offset=0
00 01 02 03     04 05 06 07 08 05 15 0B 0C 0D 0E 0F
10 11 12 13     3C 15 16 17 18 19 1A 1B 1C 1D 1E 1F
40 5A 7F 7B     5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61
F0 F1 F2 F3     F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F
7C C1 C2 C3     C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6
D7 D8 D9 E2     E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D
79 81 82 83     84 85 86 87 88 89 91 92 93 94 95 96
97 98 99 A2     A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 7F
80 81 82 83     84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93     94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3     A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3     B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3     C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3     D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3     E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3     F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
#
# Change the lowercase characters to uppercase.
```

```
offset=61
C1 C2 C3 C4     C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6 D7
D8 D9 E2 E3     E4 E5 E6 E7 E8 E9
```

Each byte stores the character value for the target character set. The source character set is used as an index into the table. For example, an ASCII blank (Hex 20) would locate the byte at offset Hex 20 in the translation table. If the byte at location Hex 20 contains Hex code 40, that would translate to an EBCDIC code indicating a blank character.

Compiling a Translation Table Using the ndmxlt Utility

You can create or modify a translation table tailored to your requirements with the ndmxlt utility program.

To invoke the **ndmxlt** utility, type the following command at the UNIX prompt:

```
$ ndmxlt -ssourcefile -ooutputfile [ -rradix] [ -ffiller] -mxlatefile
```

The parameters for the ndmxlt command are listed in the following table:

Parameter	Description	Values
-ssourcefile	The path and file name of the translation table source file. If no value is specified, input is read from STDIN.	Path and name of translation table
-ooutputfile	The path and file name of the translation table output file.	Path and name of translation output file
-rradix	The radix or base of the source file input data. All numeric values whether from command line options or input data are interpreted based on the radix setting.	x | d | o x—Hexadecimal. This is the default. d—Decimal o—Octal The default is x.
-ffiller	A filler byte value. The entire table is initialized to this value before the input data is scanned and applied to the table.	Any keyboard character, number, or special character, plus control characters entered using a preceding slash. For example, "\0" is null.
-m	The path and file name of a model translation table. If specified, the model table is read in and then the input data is scanned and applied to the table. This capability permits creating a number of different tables that are variations from a single base table without having to specify all 256 bytes of input data for each table.	Path and file name of the model translation table

Example—Creating a Translation Table

Perform the following steps to create a sample translation table that changes lowercase characters to uppercase characters:

1. Make a copy of the sample translation table located at cd_dir/ndm/src/def_send.sxlt.
2. Open the new translation table with a text editor.

3. Add the following lines to the bottom of the table. It should look like the table in "Creating a Translation Table" on page 185when you have added this information.

```
#
# Change the lowercase characters to uppercase.
offset=61
C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6 D7
D8 D9 E2 E3 E4 E5 E6 E7 E8 E9
```

4. Copy the modified file to *cd_dir*/ndm/src and name it UpperCaseEBC.sxlt.
5. Compile the new translation table using the following syntax:

```
ndmxlt -s../src/UpperCaseEBC.sxlt -oUpperCaseEBC.xlt
```

6. To use this translation table, add the following sysopts parameter to the copy statement:

```
copy from file=filename
     to   file=filename
          sysopts=":xlate.tbl=pathname/UpperCaseEBC.xlt:"
```

Example—Modifying a Model Translation Table

Perform the following steps to modify a model translation table. This method, when implemented, reads the model table and writes it to a new file. It then reads the input data and makes changes to the table created.

1. Create a file called FourLinesUpperCase.sxlt and add the following lines to the file:

```
#
# Change the lowercase characters to uppercase.
offset=61
C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6 D7
D8 D9 E2 E3 E4 E5 E6 E7 E8 E9
```

2. Copy the modified file to *cd_dir*/ndm/src.
3. Type the following command to compile this file and create a translation table called fourLineUpperCase.xlt:

```
ndmxlt -s../src/FourLineUpperCase.sxlt -oFourLineUpperCase.xlt -mdef_send.xlt
```

4. To use this translation table, add the following sysopts parameter to the copy statement:

```
copy from file=filename
     to   file=filename
          sysopts=":xlate.tbl=pathname/FourLineUpperCase.xlt:"
```

Using Translation During File Transfer Operations

Translation is specified in the copy statement of a IBM Connect:Direct Process. You can use the default translation table or create a new table.

Translation is specified in the copy statement of a IBM Connect:Direct Process. You can use the default translation table or create a new table.

To use the default translation table, type the following copy statement:

```
copy from file=abc
     to   file=xyz
          sysopts=":xlate.tbl=yes:"
```

To specify a customized table for data translation, include the following sysopts subparameter in the copy statement, where *pathname/filename* identifies the translation table:

```
copy from file=filename
     to   file=filename
          sysopts=":xlate.tbl=pathname/filename:"
```

Refer to the UNIX section of the IBM Connect:Direct Processes Web site at https://www.ibm.com/support/knowledgecenter/CD_PROC_LANG/com.ibm.help.cdprocoverview.doc/cdprc_over_what_is_a_cd_process.html for additional details concerning translation table specification with a **copy** statement.

Translation Table Error Messages

The following table displays the error messages that are generated by **ndmxlt**:

Diagnostic Number	Description
XXLT001I	Invalid directive
XXLT002I	Input file open error
XXLT003I	Model file open error
XXLT004I	Invalid filler value
XXLT005I	Invalid offset value
XXLT006I	Invalid radix value
XXLT007I	Invalid table value
XXLT008I	Table data out of bounds

Accessing IBM Connect:Direct Messages

The IBM Connect:Direct message file contains records with text for all messages, including errors and messages from IBM Connect:Direct servers other than the host server. You can add and delete message records with a text editor. The message file resides in *d_dir*/ndm/cfg/cd_node/msgfile.cfg. You can display message text with the **ndmmsg** command.

Message File Content

The message file is structured much the same way as other IBM Connect:Direct configuration files. Each record is a logical line in a text file that consists of one or more physical lines. Each record has a unique name, a message ID, and fields that make up the message text.

The message record definitions provide for symbolic substitution, which permits including actual file names or other variable information within the text to more specifically identify a problem. Symbolic variables begin with the ampersand character (&).

The format of IBM Connect:Direct message IDs is listed in the following table:

```
XxxxnnnI

Where:

X Indicates IBM Connect:Direct
xxx is a 3-character IBM Connect:Direct component identifier
nnn is a 3-digit decimal number
I is the standard, though not required, suffix
```

Message File Record Format

The following example shows the format of the message file record. Each record can be up to 4K bytes long. Optional parameters and values are in brackets.

```
message id [long.text detailed message explanation] [mod.name issuing module name] short.text
message summary
```

Following are the parameters for the message file record:

Parameter	Description	Values
long.text	A string that explains the message in detail.	A text string
mod.name	The name of the source module issuing the message ID.	Source module name
short.text	A summary of the message. This field is required.	Summary message, up to 72 characters

The following example illustrates a sample message record for XCPS008I:

```
XCPS008I:\ :mod.name=NUSMCP00.C:\
:short.text=File is not VB datatype.:\
:long.text=File is not variable block. Change sysopts datatype to\
either binary or text to transfer this file.\
\nSYSTEM ACTION-> the copy step failed and CD processing\
continued with the next process step.\
\nRESPONSE-> change the sysopts datatype to either\
binary or text.:\
```

Displaying Message Text

Use the ndmmsg command to display text in the message file. You can display both short and long text.

The following command illustrates the format for ndmmsg:

ndmmsg -f msgfname [-l | -s] msgid1 [msgid2 [msgid3 [...]]]

Following are the parameters for the ndmmsg command. If you do not specify an **l** or **s** parameter, both short and long text are displayed.

Parameter	Description
-f	Specifies the name of the message file.
-l	Displays the long text of a message.
-s	Displays the short text of a message.

Following is a sample **ndmmsg** command:

```
ndmmsg -f /usr/ndmunix/msgfile.cfg XCMG000I
```

Output from the command is displayed in the following example:

```
rc=&rc
fdbk=&fdbk
mod.name=NUCMRG00.C
func.name=ndmapi_sendcmd
short.text=CMGR RPC call returns NULL
long.text=The ndmapi_sendcmd RPC call made by the API to the CMGR returns a
NULL pointer.There is probably an RPC error.ndm.action=None
user.action=First, check if the ndmcmgr is still running; it could have
been killed accidently.If so, then abort the current CLI and restart the
```

```
CLI.  If the same problem occurs again, try to increase the value of wait
time (if set) in the API configuration file (ndmapi.cfg).
```

Precompressing/Decompressing Files Using the Standalone Batch Compression Utility

The Standalone Batch Compression Utility (cdsacomp) enables you to precompress files and then transfer the precompressed files to remote IBM Connect:Direct nodes using IBM Connect:Direct Processes. You have the following options for decompressing the files. A file can either be:

- Decompressed as it is received by the remote node (available on all IBM Connect:Direct platforms)
- Stored on the remote node and later decompressed offline using cdsacomp (available only on IBM Connect:Direct and Connect:Direct for z/OS).

Because cdsacomp can be used offline, it allows you to allocate some of the overhead associated with compression to non-peak times. For example, if you need to send the same file to several remote nodes, use this utility so that the file is precompressed only one time. You can also use cdsacomp to determine how much compression can be achieved for a file without having to transmit the file.

The cdsacomp utility is located in the IBM Connect:Direct /bin directory.

Special Considerations for Using the Standalone Batch Compression Utility

Consider the following when you are using cdsacomp to precompress files:

- If you precompress a file with the cdsacomp utility, then you cannot specify any compression options in your IBM Connect:Direct Process when you copy that file.
- You cannot specify data transformations (xlate, codepage, strip blanks, and so on) when sending a precompressed file with :precompress=yes: sysopts (for on-the-fly decompression). The following transformation options are available:
  - -x
  - -p
  - -s
  - -a
- If you precompress a file with the cdsacomp utility on a IBM Connect:Direct node, then you cannot specify a checkpoint interval in your IBM Connect:Direct Process if you decompress the file as it is received by the remote node.
- When you are copying a precompressed file to z/OS without :precomp=yes: (for deferred decompression):
  - The Copy operation must specify DCB information for the destination file. The physical block size of the destination file on Connect:Direct for z/OS must match the logical block size of the precompressed source file on Connect:Direct for UNIX.
  - The logical block size of the source file defaults to 27920 unless overridden by the -b parameter.

Using the Standalone Batch Compression Utility

To invoke the standalone batch compression utility (cdsacomp), type the following command at a UNIX prompt:

```
cdsacomp
```

Following are the parameters for the cdsacomp utility:

Parameter	Description	Values
-m	Specify which mode to use: precompress or decompress. This argument is required.	compress | decompress The default is **compress**.
-i	Specify the input file to precompress or decompress. This argument is required.	*full or relative path of input file*
-o	Specify the output file to save. If the output file already exists, it is overwritten. This argument is required.	*full or relative path of output file*
-z	Use this option with “-m compress” to override default compression values. This argument is optional. When decompressing, the values used during compression are used.	*level, window, memory* level—**Compression level**. The range is 1–9. The default is **1**. 1—Provides the least compression, but is the fastest. 9—Provides the most compression, but is the slowest. window—The size of the compression window and history buffer. Increasing the window increases the compression, but uses more virtual memory. The range is 9–15. The default is **13**. memory—**The amount of virtual memory to allocate.** The range is 1–9. The default is **4.** 1—Uses the least virtual memory. 9—Uses the most virtual memory.
-x	Use this option to translate the file. If this parameter is not specified, the file is not translated. This parameter is mutually exclusive with -codepage.	*full path to translate table file | relative path to translate table file*
-p	Use this option to specify codepages for file conversion. Default is no codepage translation. This parameter is mutually exclusive with -xlate.	*source codepage, destination codepage*

Parameter	Description	Values
-d	Specify the datatype of the file. When you use “-m compress”, the datatype values result in the following: • text Strips newline characters from each record Supports -s and -a parameters Uses data attributes of blocksize=23040, recfm=vb, lrecl=23036, dsorg=ps • binary Uses data attributes of blocksize=23040, recfm=u, lrecl=0, dsorg=ps Does not support -s and -a parameters • VB Does not support -x, -p, -s, and -a parameters Uses data attributes of blocksize=23040, recfm=vb, lrecl=23036, dsorg=ps When you use “-m decompress”, the datatype values result in the following: • text Inserts newline characters into each record Supports the -s parameter • binary Does not support the -s parameter • VB Does not support -x, -p, and -s parameters	text | binary | VB The default is **text.**
-b	Specify the block size of the output file. This parameter is valid only when you specify “-m compress” for the compression option.	*nnnnn* The range is 4096–32760. The default is **27920**.
-s	Use this option to strip trailing blanks. This parameter is valid only when you specify “-d text” for the datatype of the file.	y | n y—yes n—no The default is **y**.
-a	Use this option to replace zero-length records with a single, blank character. This parameter is valid only when you specify the following: “-d text” and “-m compress”.	y | n y—yes n—no The default is **y**. Specify **n** if the data is copied to an i5OS or mainframe node.

Parameter	Description	Values
-h	Use this option to display online help for the utility.	No values are available for this parameter.

Example—Precompress a Text File

In this example, the source file is a text file named source.file which is precompressed into a destination file named compressed.file. The file is translated using the default translation table, /home/cd/ndm/xlate/def_send.xlt. Trailing blanks are stripped. Default settings for ZLIB tuning, checkpoint interval and block size are used.

```
cdsacomp -m compress
         -d text
         -i source.file
         -o compressed.file
         -x /home/cd/ndm/xlate/def_send.xlt
         -s y
```

Example—Precompress a Text File With Codepage Conversion

In this example, the source file is a text file named zzz.sac which is precompressed into a file named zzz.txt. The file is converted from EBCDIC-US to ASCII using the codepage option. Default settings are used for parameters that are not specified.

```
cdsacomp -m compress
         -d text
         -i zzz.txt
         -o zzz.sac
         -p EBCDIC-US,ASCII
```

Example—Precompress a Binary File

In this example, the source file is a binary file named source.file which is precompressed into a destination file named compressed.file. Default settings are used for parameters that are not specified.

```
cdsacomp -m compress
         -d binary
         -infile source.file
         -outfile compressed.file
```

Example—Decompress a Text File

In this example, the source file is a precompressed text file named compressed.file which is decompressed into a destination file named dest.file. The file is translated using the default translation table, /home/cd/ndm/xlate/def_recv.xlt. Default settings are used for parameters that are not specified.

```
cdsacomp -m decompress
         -d text
         -i compressed.file
         -o dest.file
         -x /home/cd/ndm/xlate/def_recv.xlt
```

Examples—csdacomp Command Help

Requesting a summary of cdsacomp command parameters and help options:

```
cdsacomp -h
```

Example—Decompress a File on the Remote Node During the Copy Step

The “precomp=yes” parameter is used when the file was compressed by the cdsacomp utility prior to this Process. The file is transferred by this Process as a pre-compressed file. It is then decompressed by special processing as it is received on the remote node.

```
sample process snode=cdunix1
step01 copy
from
(
file=/home/cd/upload/compressed.file
sysopts=”:precomp=yes:”
pnode
)
to
(
file=/home/cd/download/decompressed.file
snode
disp=rpl
)
pend;
```

Example—Send Precompressed File to z/OS and Storing It as Precompressed

The precompressed file is copied to the z/OS node with PNODE sysopts of “datatype=binary”. The destination file is not decompressed. The DCB settings of the original precompressed file are preserved on the z/OS node. The specified checkpoint interval will be used during the file transfer. The file can be decompressed with the z/OS cdsacomp utility.

```
sample process snode=cdunix1
step01 copy
from
(
file=/home/cd/upload/compressed.file
sysopts=”:datatype=binary:”
pnode
)
chkpt=2M
to
(
file=upload.compressed.file
dcb=(blksize=27920, lrecl=0, dsorg=ps, recfm=u)
snode
disp=(new,catlg)
)


        pend;
```

Validate Configuration Files

When you manually edit any of the five text-based IBM Connect:Direct configuration files, the Configuration Checking Utility (cfgcheck) enables you to validate these files offline. The following files can be validated using this utility: userfile.cfg, initparm.cfg, netmap.cfg, ndmapi.cfg, and sysacl.cfg.

Note: The Strong Access Control File (sysacl.cfg) will be validated only when the user running the Configuration Checking Utility is a root user.

By default, cfgcheck is run with no arguments and attempts to find all five of the configuration files in the current working directory. If all of the IBM Connect:Direct components are not installed, then some of the files will not be found. For example, if the Command Line Interface (CLI) is installed but the IBM Connect:Direct server is not installed, only the ndmapi.cfg file will exist in the installation directory. Therefore, only the ndmapi.cfg file will be validated. When cfgcheck is run with no arguments, the utility will report that the other configuration files were not found.

Note: Before you can execute cfgcheck, you must set the NDMAPICFG environment variable. For more information, see “Controlling and Monitoring Processes” on page 147.

To invoke cfgcheck, type the following command at the UNIX prompt:

```
$ cfgcheck -t -h -f filename.cfg
```

The **cfgcheck** command has the following arguments:

Argument	Description
No arguments (default)	When no arguments are specified and the cfgcheck utility is run by a non-root user, it searches the `cfg/ directory` for the following configuration files: initparm.cfg, netmap.cfg, userfile.cfg, and ndmapi.cfg. When a root user runs cfgcheck, the utility also searches the SACL/ `directory` to locate the sysacl.cfg file.
-h	Prints the help screen and exits.
-t	Turns on tracing and prints verbose debug information.
-f filename.cfg	Specifies a configuration file name to validate, where filename is the name of one of the configuration files. You can specify multiple -f arguments. When the -f argument is used, cfgcheck will not automatically search for other configuration files from the file specified.

Configuration Reports

You can generate a report of your system information and IBM Connect:Direct configuration information using the Configuration Reporting Utility (cdcustrpt). Configuration reports can be generated for the following IBM Connect:Direct components:

- Base installation of IBM Connect:Direct
- Connect:Direct Secure Plus for UNIX

During the IBM Connect:Direct installation, cdcustrpt is installed in the *<installation>*/etc/ directory.

Generating a Configuration Report on the Base Installation

When you use cdcustrpt to generate a report on the base IBM Connect:Direct installation, it reports the following types of system information:

- Name and other information of the operating system
- Space on file systems
- Virtual memory statistics
- Contents of the IBM Connect:Direct installation directory

In addition to reporting system information, cdcustrpt invokes the Configuration Checking Utility (cfgcheck) to validate the syntax of the five text-based configuration files (if they are available and if the user has access to the files) and to report on the contents of the configuration files. For more information on cfgcheck, see "Validate Configuration Files" on page 194.

In this procedure, default values are computed by the utility based on the location and name of the installed IBM Connect:Direct and are provided in brackets "[ ]". Press **Enter** to accept the default values.

To invoke cducustrpt and generate a report of the base installation:

1. Type the following command at a UNIX prompt:

```
% cdcustrpt
```

2. Type the full path where IBM Connect:Direct is installed and press **Enter**.
3. Type the full path and name for the report that will be generated and press **Enter**.

   The report is generated in the location you specified, and any error messages are displayed as shown in the following example:

```
% cdcustrpt

Enter full path of Connect:Direct destination directory:[/sci/users/jbrown1/cd40]:

Enter full path and name for this support report file:[/sci/users/jbrown1/cd40/etc/
cd.support.rpt]:

ls: /sci/users/jbrown1/cd40/ndm/SACL: Permission denied

cdcustrpt ended
```

   In this example, the user does not have root access, so the Strong Access Control File (sysacl.cfg) can not be accessed. The following example shows an excerpt from a sample report:

```
###########################################################################
#######     Connect:Direct for UNIX 4.0.00 configuration report     #######
###########################################################################
Connect:Direct for UNIX Version 4000, Build 00, IBM/RS6000 AIX, Fix date:
01OCT2007

Install directory: /sci/users/jbrown1/cd40

Local Node name: jb_aix40

Report for: jbrown1

=========================================================
=====   Begin: Environment and system information   =====
=========================================================

System: AIX skyglass 3 5 00CE208E4C00


Disk usage:
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on

/dev/hd4          262144     64216   76%     2479     4% /

/dev/hd2         8126464   2708688   67%    37802     4% /usr

/dev/hd9var       262144     18448   93%      613     2% /var

/dev/hd3          786432    363600   54%      424     1% /tmp

/dev/fwdump       524288    507752    4%       17     1% /var/adm/ras/platform

/dev/hd1          262144    216520   18%      167     1% /localhome

/proc                  -         -    -         -     -  /proc

/dev/hd10opt      524288     52168   91%     3688     6% /opt

/dev/fslv00    121634816  13629040   89%   264984    15% /sci

scidalnis01:/export/nis01 1677670392 512499192   70%        0    -1% /home/nis01


Memory statistics:
System Configuration: lcpu=4 mem=3824MB

kthr    memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  1 400072 232777   0   0   0   0    0   0   4 1805 197  0  1 99  0
```

Generating a Configuration Report on Connect:Direct Secure Plus for UNIX

If cdcustrpt detects the Connect:Direct Secure Plus directory in the installation directory, <*installation*>/ndm/secure+/, it invokes the Connect:Direct Secure Plus Command Line Utility (splicli.sh) to report on Secure+ parameters. If Connect:Direct Secure Plus is detected, you are prompted to enter the path to the Connect:Direct Secure Plus parameters file (the default location is provided in brackets "[ ]"), for example:

```
Enter full path of Secure+ parmfile
directory: [/sci/users/jbrown1/cd40/ndm/secure+/nodes]:
```

The following example shows an excerpt from a sample report:

```
  ===== Begin: Secure+ parameters   =====
  ========================================
  All secure+ nodes in /data/cd4204sp/ndm/secure+/nodes:
    ************************************************************************************
    *          Secure+ Command Line Interface                                          *
    *          IBM(R) Connect:Direct(R) Secure Plus v6.0.0                             *
    *----------------------------------------------------------------------------------*
    *   Licensed Materials - Property of IBM                                           *
    *   (C) Copyright IBM Corp. 1999, 2018 All Rights Reserved.                        *
    *   US Government Users Restricted Rights - Use, duplication                       *
    *   or disclosure restricted by GSA ADP Schedule Contract                          *
    *   with IBM Corp.                                                                 *
    ************************************************************************************

   SPCLI> display all;
     Name=.Client
     BaseName=.Client
     Type=R
     Protocol=DefaultToLN
     Override=Y
     SecurityMode=DefaultToLN
     AuthTimeout=120
     KeyCertLabel=
     ClientAuth=N
     CertCommonName=
     CipherSuites=()


     Name=.Keystore
     File=/data/cd4204sp/ndm/secure+/certificates/cdkeystore.kdb


     Name=.Local
     BaseName=cd4204sp
     Type=L
     Protocol=TLS1.2
     Override=Y
     SecurityMode=Disable
     AuthTimeout=120
     SeaEnable=N
     SeaCertValDef=
     KeyCertLabel=MYSHA2562048ID
     EncryptData=Y
     ClientAuth=N
     CipherSuites=(TLS_RSA_WITH_AES_256_GCM_SHA384,
       TLS_RSA_WITH_AES_256_CBC_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA,
       TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,
       TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_RC4_128_SHA,
       TLS_RSA_WITH_3DES_EDE_CBC_SHA,TLS_RSA_WITH_NULL_SHA256,TLS_RSA_WITH_NULL_SHA)

     2016/03/11 11:58:07 cdtest
```

Writing Custom Programs

The IBM Connect:Direct Application Programming Interface (API) allows you to write custom programs in either C or C++ to use with IBM Connect:Direct. With the C functions or the C++ classes, you can create programs to perform the following tasks:

- Establish a connection to the IBM Connect:Direct server

- Disconnect from the server
- Receive command responses from the server
- Send commands to the server

This topic describes the format of the IBM Connect:Direct API functions and classes and provides samples of their use. Sample programs are provided that use the IBM Connect:Direct API functions and classes to issue commands and receive responses from the IBM Connect:Direct server.

Compiling Custom Programs

After you write a custom program, you must compile it, using a C or C++ compiler. Refer to the following information to determine what minimum C++ compiler version to use for each platform:

Platform	C++ Compiler
AIX	IBM XL C/C++ for AIX, V11.1
Sun Solaris	Sun C++ 5.12
HP-Itanium	aCC: HP ANSI C++ B3910B A.06.07
Linux x86	g++ (GCC) 6.3.1
Linux S390	g++ (GCC) 4.4.7
Linux ppc64le	g++ 4.8.5

Use the commands defined in the following table to compile a custom C++ program using the C++ API calls:

Platform	C++ Compile Command
AIX	
64-bit	/usr/vacpp/bin/xlC -q64 -qinline -I../include -+ -o sdksample sdksample.C ../lib/ndmapi64.a -lbsd -ldl -lsrc -lpthreads
Sun	
64-bit	/opt/SUNWspro/bin/CC -m64 -DBSD_COMP -I../include -o sdksample sdksample.C ../lib/ndmapi64.a -L/usr/ucblib/sparcv9 -L/usr/lib/sparcv9 -lsocket -lrpcsoc -lnsl -lelf -ldl -R/usr/ucblib/sparcv9
HP-Itanium	
64-bit	/opt/aCC/bin/aCC +DD64 -I../include -o sdksample sdksample.C ../lib/ndmapi64.a -L/usr/lib/hpux64 -lrpcsvc -lnsl -ldld -Wl,+s -lunwind
Linux x86	
64-bit	g++ -I/usr/include/tirpc -I../include -O -DLINUX -o sdksample sdksample.C ../lib/ndmapi64.a -ldl -lstdc++ -ltirpc **Note:** The `libtirpc-devel` package is required for compiling.
LinuxS390	

Platfo rm	C++ Compile Command
64-bit	g++ -I/usr/include/tirpc -I../include -O -DLINUX -o sdksample sdksample.C ../lib/ndmapi64.a -ldl -lstdc++ -ltirpc **Note:** The `libtirpc-devel` package is required for compiling.
Linux ppc64le	
64-bit	g++ -I/usr/include/tirpc -I../include -O -DLINUX -o sdksample sdksample.C ../lib/ndmapi64.a -ldl -lstdc++ -ltirpc **Note:** The `libtirpc-devel` package is required for compiling.

To build a C++ program using the C API calls, such as the apicheck.C sample program, replace the sdksample.C parameter with the name of the C++ program and rename the output file parameter, -o sdksample, to the name of the output file you want to create such as apicheck.

Use the commands defined in the following table to compile a C program:

Platfo rm	C Compile Command
AIX	
64-bit	/usr/vacpp/bin/xlc -q64 -I../include -+ -o apicheck apicheck.c ../lib/ndmapi64.a -lbsd -ldl -lsrc -lC -lpthreads
Sun	
64-bit	/opt/SUNWspro/bin/cc -m64 -DBSD_COMP -I../include -o apicheck apicheck.c ../lib/ndmapi64.a -L/usr/ucblib/sparcv9 -L/usr/lib/sparcv9 -lsocket -lCstd -lCrun -lrpcsoc -lnsl -lelf -ldl -lCrun -R/usr/ucblib/sparcv9
HP-Itanium	
64-bit	/opt/ansic/bin/cc +DD64 -I../include -o apicheck apicheck.c ../lib/ndmapi64.a -L/usr/lib/hpux64 -lrpcsvc -lnsl -ldld -Wl,+s -lcl -lstd_v2 -lCsup -lunwind
Linux x86	
64-bit	gcc -I/usr/include/tirpc -I../include -O -DLINUX -o apicheck apicheck.c ../lib/ndmapi64.a -ldl -lstdc++ -ltirpc **Note:** The `libtirpc-devel` package is required for compiling.
LinuxS390	
64-bit	gcc -I/usr/include/tirpc -I../include -O -DLINUX -o apicheck apicheck.c ../lib/ndmapi64.a -ldl -lstdc++ -ltirpc **Note:** The `libtirpc-devel` package is required for compiling.
Linux ppc64le	
64-bit	gcc -I/usr/include/tirpc -I../include -O -DLINUX -o apicheck apicheck.c ../lib/ndmapi64.a -ldl -lstdc++ -ltirpc **Note:** The `libtirpc-devel` package is required for compiling.

Writing Custom C Programs

If you write a custom program using the C API calls, you must include the header file ndmapi.h and link it with ndmapi.a. A sample program called apicheck.c is provided.

For Java programming, you can call the C API functions by using the JNI and the libndmapi shared objects: libndmapi.sl for HP and libndmapi.so for the other supported platforms. Although the JNI is supported, the IBM Connect:Direct Java Application Interface is recommended for Java programs that invoke the services of IBM Connect:Direct.

Note: The environment variable NDMAPICFG must be set to the pathname of the client configuration file. Refer to "Controlling and Monitoring Processes" on page 147 for instructions on setting the environment variable.

Use the following IBM Connect:Direct API functions for C and C++ programs:

C++ Function	C Function	Description
ndmapi_connect()	ndmapi_connect_c()	Establishes a connection with the server. Specify the node to connect to in the ndm_nodespec pointer or in the CLI/API Configuration Information file. If the call is successful, NDM_NO_ERROR is returned. Control returns to the application when the connection is established and is ready for the first API request.
ndmapi_sendcmd()	ndmapi_sendcmd_c()	Sends commands to IBM Connect:Direct. You must provide the command text. The resp_moreflag is a pointer to the flag indicating that more responses are pending for the executed command. Invoke ndmapi_recvresp_c() for C programs or ndmapi_recvresp() for C++ programs to retrieve the extra responses. Only the select process and select statistics commands require the use of ndmapi_recvresp_c() for use with C and ndmapi_recvresp() for use with C++.
ndmapi_recvresp()	ndmapi_recvresp_c()	Receives responses to commands sent to IBM Connect:Direct. The contents of the response buffer are returned.
ndmapi_disconnect()	ndmapi_disconnect_c()	Terminates the API connection.

Three types of IBM Connect:Direct command responses are returned by these functions.

- Informational responses return information about the submitted command.
- Data responses, stored in the resp_buffer, contain data records.
- Error responses return ERROR_H, a pointer to a linked list of all errors found. The ID field values are fixed for use when debugging. The msgid, feedback, and rc fields are specified by IBM Connect:Direct and are referred to in message text. The subst field points to a string that contains substitution variable information to be inserted appropriately in the message text. The error control structure keeps track of the current and total number of errors. You can move through the errors by using the next pointer in error entry blocks.

The following code defines the ERROR_H structure:

```
#define NDM_ERR_ENT_T struct NDM_ERR_ENT_S
#define NDM_ERR_ENT_H NDM_ERR_ENT_T *
#define NDM_ERR_CTL_T struct NDM_ERR_CTL_S
#define ERROR_H       NDM_ERR_CTL_T *
struct NDM_ERR_ENT_S
{
        int32               id;
        char               msgid[MSGIDLEN];
        int32               feedback;
        int32               rc;
        char               *subst;
        NDM_ERR_ENT_H                next;
};
struct NDM_ERR_CTL_S
{
        int32               id;
        int32               cur_entry;
        int32               num_entries;
        NDM_ERR_ENT_H                next;
};
```

Creating a Connection to IBM Connect:Direct Using ndmapi_connect() or ndmapi_connect_c()

Use ndmapi_connect() or ndmapi_connect_c() to create a connection to IBM Connect:Direct so that an application can send commands and receive responses from the commands. Control returns to the application when the connection is established and IBM Connect:Direct is ready for the first API request or when an error condition is set.

Following is the format for the **ndmapi_connect()** or **ndmapi_connect_c()** function:

```
int32 ndmapi_connect ERROR_H error, char *  ndm_hostname, char *  ndm_portname
```

The following table describes the parameters for the **ndmapi_connect()** or **ndmapi_connect_c()** function:

Parameter	Description	Value
error	A pointer to a IBM Connect:Direct-defined structure that contains error information or status information.	Pointer
ndm_hostname	A pointer to the text specification of the IBM Connect:Direct host to which the connection is made. If this parameter is not specified, the host name is determined by first checking the environment variable TCPHOST. If no value is specified, the tcp.host.name field in the CLI/API configuration file is checked. If no value is specified, the gethostbyname() command is invoked and the default value of ndmhost is used.	Null-terminated string
ndm_portname	A pointer to the host port number. If this parameter is not specified, the environment variable TCPPORT is checked. If no value is specified, the value of the tcp.port in the CLI/API configuration file is checked. If no value is specified, the default value **1363** is used.	Pointer

The **ndmapi_connect()** or **ndmapi_connect_c()** function has the following return codes:

Parameter	Description
NDM_NO_ERROR	A session was established with the server.
NDM_ERROR	A session was not established with the server. Consult the error structure for detailed error status.

The following sample function illustrates the use of **ndmapi_connect()** to connect to the sun1 host:

```
rc=ndmapi_connect (error, "sun1", "3122");
```

Terminating a Connection Using ndmapi_disconnect() or ndmapi_disconnect_c()

Use **ndmapi_disconnect()** or **ndmapi_disconnect_c()** to terminate a connection to IBM Connect:Direct that was established by a call to **ndmapi_connect()** or **ndmapi_connect_c()**. The **ndmapi_disconnect()** or **ndmapi_disconnect_c()**function call is the following:

```
void ndmapi_disconnect
```

There are no parameters and no return codes for **ndmapi_disconnect()** or **ndmapi_disconnect_c()**. Following is a sample ndmapi_disconnect() function:

```
ndmapi_disconnect ();
```

Receiving Responses Using ndmapi_recvresp() or ndmapi_recvresp_c()

Use **ndmapi_recvresp()** or **ndmapi_recvresp_c()** to receive responses that are associated with a previous command sent from the application. Following is the format for **ndmapi_recvresp()** or **ndmapi_recvresp_c()**:

```
int32 ndmapi_recvresp ERROR_H  error int32 *  resp_length, char *   resp_buffer,
int32 * resp_moreflag
```

Following are the parameters for ndmapi_recvresp() or **ndmapi_recvresp_c()**:

Parameter	Description	Value
error	A pointer to a IBM Connect:Direct-defined structure that contains error information or status information.	Pointer
resp_length	A pointer to the length, in bytes, of the application buffer to receive the response. The API sets this parameter to the number of bytes returned.	Pointer to number of bytes returned or 0 if you no longer want to receive responses. Setting this field to zero purges all queued responses.
resp_buffer	A pointer to the application buffer that receives the command or submit response. This buffer should allocate 4096 bytes. The format of **resp_buffer** is a free-form text record structure. Field names are four characters long and all uppercase. The data can be any length and can include blanks. The structure is: field name=data | field name=data |... For example: SUBM = username | PNUM = 12 | PNAM = proc1 |...	A local buffer, with a size greater than or equal to that set by **r**esp_length and filled in by **ndmapi_recvresp()** or **ndmapi_recvresp_c()**. The CLI passes the resp_buffer to AWK for parsing. Valid values include: ADMN—IBM Connect:Direct administrator name ADPH—IBM Connect:Direct administrator phone number CCOD—Completion code CKPT—Checkpoint CLAS—Class CSDS—Copy step start timestamp CSPE—Secure+ enabled indicator CSPP—Secure+ protocol CSPS—Secure+ cipher suite

Parameter	Description	Value
		DBLW—Destination file blocks received DBYW—Bytes written DBYX—Bytes received DCOD—Destination completion code DDAY—Submit date DDS1—Destination disposition 1 DDS2—Destination disposition 2 DDS3—Destination disposition 3 DESC—IBM Connect:Direct administrator description DFIL—Destination file DLDR—Download directory restriction DMSG—Destination message ID DNVL—Destination number of volumes DRCW—Records written DRUX—RUs received DVCN—Destination file volume count DVOL—Destination volume array DVSQ—Destination file volume sequence number ECMP—Extended compression ON or OFF ECPR—Extended compression percentage ECTP—Extended compression type ETMC—Ela[sed clock time ETMK—Elapsed kernal time ETMU—Elapsed user time FROM—Copy sending node

Parameter	Description	Value
		ICRC—CRC indicator LCCD—Local completion code LCLP—Local IP address and port number LKFL—Link fail LMSG—Local message ID LNOD—Local node MSGI—Message ID MSGT—Message text MSST—Short text OCCD—Other completion code OERR—Other node in error OMSG—Other message ID PACC—PNODE account PCRC—CRC indicator PFIL—Process file PNAM—Process name PNOD—PNODE PNUM—Process number PPMN—PDS member name PRTY—Priority QUEU—Queue RECC—Record category RECI—Record ID RETA—Retain Process RMTP—Remote IP address and port number RSTR—Process restarted RUSZ—RU Size SACC—SNODE account SBID—Submitter node ID SBLR—Source file blocks sent

Parameter	Description	Value
		SBND—Submitter node name SBYR—Bytes read SBYX—Bytes sent SCMP—Standard compression SCOD—Source completion code SCPR—Standard compression percentage SDDY—Schedule date SDS1—Source disposition 1 SDS2—Source disposition 1 SDS2—Source disposition 2 SDS3—Source disposition 3 SELA—Elapsed time of the event SFIL—Source file SFSZ—source file size SMSG—Source message ID SNAM—Step name SNOD—SNODE SNVL—Source number of volumes SOPT—SYSOPTS record SRCR—Records read SRUX—RUs sent SSTA—Start time of the event **Note:** SSTA is deprecated and shall be removed in the future release. Use use STRT instead. STAR—Start log date/time for record STAT—Process status STDL—Copy termination record (CTRC) log time STIM—Schedule time STOP—Stop time of the event **Note:** STOP is deprecated and shall be removed in the future release. Use STPT instead. STPT—Stop time of the event STRT—Start time of the event

Parameter	Description	Value
		SUBM—Submitter ID SUBN—Submitter node SUMM—Summary output selector SVCN—Source file volume count SVOL—Source volume array SVSQ—Source file volume sequence number TIME—Submit time TZDI—Local time zone delta from GMT XLAT—Translation ZLVL—Zlib compress level Zlib—memory level Zlib—window size
resp_moreflag	Indicates that more ndmapi_recvresp() or ndmapi_recvresp_c() calls must be issued for more information. This flag occurs only on select process and select statistics commands.	None

The **ndmapi_recvresp()** or **ndmapi_recvresp_c()** function has the folliowing return codes:

Return Code	Description
NDM_NO_ERROR	The function completed successfully.
NDM_ERROR	An error occurred. Consult the error structure for detailed error status.
TRUNCATED	Data is truncated because the receiving buffer is too small.

Following is a sample **ndmapi_recvresp()** function:

```
int32 rc, resp_length;
int32 resp_moreflag;
char resp_buffer[makbuf];

rc= ndmapi_recvresp (error,
                      &resp_length,
                                    resp_buffer,
                                    &resp_moreflag
                                    );
```

Sending a Command to IBM Connect:Direct Using ndmapi_sendcmd() or ndmapi_sendcmd_c()

Use **ndmapi_sendcmd()** or **ndmapi_sendcmd_c()** to allow a command to be sent to a IBM Connect:Direct application. Following is the format of ndmapi_sendcmd() or **ndmapi_sendcmd_c()**:

```
int32  rc, resp_moreflag;
struct sendcmd_data ret_data;
rc=ndmapi_sendcmd (error,
                   "select process pnumber=2,",
                   &resp_moreflag,
                                 &ret_data
                                );
```

Following are the parameters for **ndmapi_sendcmd()** or **ndmapi_sendcmd_c()**:

Parameter	Description	Value
error	A pointer to a IBM Connect:Direct-defined structure that contains error information or status information.	Pointer
cmd_text	A pointer to the null-terminated text string that specifies the command to send to IBM Connect:Direct. The command text must be followed by a semicolon and terminated with a null. When you use the **submit=filename** command from the API, ensure that you allocate enough storage for the Process text. The text of the Process submitted is returned in the text string associated with this parameter when the function completes. If you do not allocate enough storage for the Process text, a core dump can result.	Pointer to a text string
resp_moreflag	A pointer to the flag that indicates that more responses are pending for the command just executed. Invoke **ndmapi_recvresp()** or **ndmapi_recvresp_c()** to retrieve the extra responses.	Pointer to a flag

ret_data	A pointer to a structure containing internal response information for a command. The structure is:
	
	`struct sendcmd_data  {`
	`        char  * cmd_name;`
	`        ulong cmd_id;`
	`        long data1;`
	`        long data2;`
	`        long data3;`
	`};`
		Pointer to a structure

sendcmd_data	Provides the caller with some information about the user request. Because parsing of command text occurs at the CMGR, the End User Application (EUA) has no way to identify the command that was submitted, unless it generated the text.	Information about the user request
cmd_name	A pointer to a string with the name of the command submitted. The CLI uses this pointer to display completion messages. This field enables you to display unique completion messages without any knowledge of a specific command in the EUA.	Pointer to name of command

Parameter	Description	Value

cmd_id	A four-byte identifier of the command that was found in the command text. Following are the four-byte identifiers:
	`/**************Command IDs*******************/`
	`#define CHANGE_PROCESS 0x43484750  /* "CHGP" */`
	`#define DELETE_PROCESS     0x44454c50  /* "DELP"*/`
	`#define FLUSH_PROCESS 0x464c5350  /* "FLSP" */`
	`#define SELECT_PROCESS     0x53454c50  /* "SELP"*/`
	`#define SELECT_STATISTICS 0x53454c53  /* "SELS" */`
	`#define SUBMIT 0x5355424d  /* "SUBM" */`
	`#define TRACE_API 0x41504920  /* "API " */`
	`#define TRACE_CMGR 0x434d4752  /* "CMGR" */`
	`#define TRACE_SMGR 0x534d4752  /* "SMGR" */`
	`#define TRACE_PMGR 0x504d4752  /* "PMGR" */`
	`#define TRACE_COM 0x434f4d4d  /* "COMM"*/`
	`#define TRACE     0x54524143  /* "TRAC" */`
	`#define STOPNDM 0x53544F50   /* "STOP" */`
	The CLI uses these identifiers to ensure that rules are being followed. For instance, if an **ndmapi_sendcmd** returns with the **resp_moreflag** set and the **cmd_id** is not **SELECT_STATISTICS** or **SELECT_PROCESS**, the CLI generates an error.
		Four-byte identifier

data1, data2, and data3	For future expansion. data1 is used with the **submit** command to return the Process number. data2 is used with the submit command to return the result of the Process (0, 4, 8, or 16)	

The **ndmapi_sendcmd_c()** function call has the following return codes:

Return Code	Description
NDM_NO_ERROR or Process Number	The function completed successfully.
NDM_ERROR	An error occurred. Consult the error structure for detailed error status.

Following is a sample ndmapi_sendcmd() function:

```
int32  rc, resp_moreflag;
struct sendcmd_data ret_data;
rc=ndmapi_sendcmd (error,
                   "select process pnumber=2 ;",
                   &resp_moreflag,
                                 &ret_data
                                 );
```

Writing Custom C++ Programs

If you write a custom program using C++ API calls, you must include the class called ConnectDirectSession. The calling program must instantiate ConnectDirectSession and call the send and receive functions. A sample program called sdksample.C is provided. To write a custom C++ program, create a ConnectDirectSession class. The class contains the ConnectDirectSession interface and a constructor and destructor call to allocate and release the storage associated with the class. This class is the interface to the IBM Connect:Direct methods and provides connection, command, data retrieval, and error services. Each method returns either CD_SUCCESS or CD_FAILURE.

Note: The environment variable NDMAPICFG must be set to the pathname of the client configuration file. Refer to "Starting the CLI" on page 147for instructions on setting the environment variable.

To use the ConnectDirectSession class, your application must include the cdunxsdk.h header file provided in the installation and must link with the ndmapi.a file. Following is a sample ConnectDirectSession class program:

```
#include "cdunxsdk.h"
#include <iostream.h>
#include <string.h>
void getError(ConnectDirectSession& cdSess);
main()
{
    ConnectDirectSession cdSess;
    char processText[16384];
    if (cdSess.SessionINF->Connect() == CD_SUCCESS)
    {
        strcpy(processText,"submit maxdelay=unlimited sdksample process snode=SNODENAME ");
        strcat(processText,"copy00 copy from (file=sample.txt pnode)");
        strcat(processText,"              to (file=sample.000 snode disp=rpl) ;");
        if (cdSess.SessionINF->SendCommand(processText) == CD_SUCCESS)
        {
            printf("%s completed, pnumber = %ld.\n",
                 cdSess.SessionINF->GetCommandName(),
           cdSess.SessionINF->GetProcessNumber());
sprintf(processText, "SELECT STATISTICS PNUMBER=%ld DETAIL=YES ;", cdSess.SessionINF-
>GetProcessNumber());
(cdSess.SessionINF->SendCommand(processText) == CD_SUCCESS)
            {
 }
        else
          {
            getError(cdSess);
        }
      }
        else
        {
        getError(cdSess);
      }
 cdSess.SessionINF->DisConnect();
    }
    else
    {
        getError(cdSess);
    }
}
void getError(ConnectDirectSession& cdSess)
{
    if (cdSess.SessionINF->GetFirstError())
    {
        printf("\nError Message:  %s",   cdSess.SessionINF->GetMsgID());
        printf("\nError Feedback: %d",   cdSess.SessionINF->GetFeedBackCode());
        printf("\nError RC:       %d",   cdSess.SessionINF->GetReturnCode());
        printf("\nError SUBST:    %s\n", cdSess.SessionINF->GetSubstitute());    }
 while(cdSess.SessionINF->GetNextError())
    {
        printf("\nError Message:  %s",   cdSess.SessionINF->GetMsgID());
        printf("\nError Feedback: %d",   cdSess.SessionINF->GetFeedBackCode());
        printf("\nError RC:       %d",   cdSess.SessionINF->GetReturnCode());
        printf("\nError SUBST:    %s\n", cdSess.SessionINF->GetSubstitute());
        }
}
```

The ConnectDirectSession class methods are described in the following table:

Method	Description	Parameter	Return Values
Connect	Provides a connection to the IBM Connect:Direct server. Connect() with a void parameter connects to the hostname and port specified in the client configuration file.	void or a pointer to an IP address and port.	CD_SUCCESS or CD_FAILURE
DisConnect	Disconnects the current session.	void	CD_SUCCESS or CD_FAILURE

Method	Description	Parameter	Return Values
SendCommand	Sends a IBM Connect:Direct command to the server for processing.	Pointer to a command text buffer.	CD_SUCCESS or CD_FAILURE
ReceiveResponse	Receives the response from a previously issued command, such as the select statistics command.	void	CD_SUCCESS or CD_FAILURE
GetResponse	Retrieves the response from the ReceiveResponse call.	void	Pointer to a response buffer.
GetResponseLength	Returns the length of the previously received response buffer.	void	Length of the response buffer from the previously issued call.
MoreData	Returns a value indicating if outstanding data from the previously issued send command call is available. If the return value is TRUE, call ReceiveResponse again to retrieve more data.	void	TRUE—If more data is outstanding. FALSE—If no data is outstanding.
GetCommandName	Returns the command name of the previously issued send command, such as the submit command.	void	Pointer to a command name buffer.
GetProcessNumber	Returns the Process number of a previously issued submit command.	void	Process number of a submit command. -1—If no submit command can be found.
GetProcessCount	Returns the number of Processes affected by the last send command that issued a delete, change, or flush process.	void	Process number of a submit command that issued a delete, change or flush process. -1—If no submit command can be found.
GetCurrentError	Moves the error data pointer to the current error in the list.	void	TRUE—If successful FALSE—If no current error exists.
GetNextError	Moves the error data pointer to the next error in the list.	void	TRUE—If successful FALSE—If no more errors are found.
GetPreviousError	Moves the error data pointer to the previous error in the list.	void	TRUE—If successful FALSE—If no previous error exists.
GetFirstError	Moves the error data pointer to the first error in the list.	void	TRUE—If successful FALSE—If no error is found.

Method	**Description**	**Parameter**	**Return Values**
GetLastError	Moves the error data pointer to the last error in the list.	void	TRUE—If successful, otherwise FALSE.
GetMsgID	Retrieves the message of the current error data block. You must call one of the GetXXXXError methods before calling this method in order to retrieve the proper results.	void	Return Value: Pointer to a message ID if data block is value.
GetFeedBackCode	Returns the feedback code of the current error data block.	void	Feedback code.
GetReturnCode	Returns the IBM Connect:Direct return code.	void	One of the valid IBM Connect:Direct return code: 1,4,8,16.
GetStatus	Returns the status.	void	IBM Connect:Direct status code.
GetSubstitute	Returns the current substitution buffer associated with the error.	void	Pointer to a substitution buffer.
DisplayError	Displays the current error chain to an output location.	Parameters: Pointer to a file I/O structure.	Return Value: Returns the highest error found in the error chain or -1 on error.

Following is the ConnectDirectSession class header:

```
#include <stdio.h>

// Error enumeration.
typedef enum CDErrorCode
{
    CD_SUCCESS =  0,
    CD_FAILURE = -1

} CDErrorCode;

// <<Interface>>
class CDSession
{
public:
    // Communication methods...
    virtual CDErrorCode Connect(void) = 0;
    virtual CDErrorCode Connect(char *IpAddress, char *IpPort) = 0;
    virtual CDErrorCode DisConnect(void) = 0;
    virtual CDErrorCode SendCommand(char *CmdText) = 0;
    virtual CDErrorCode ReceiveResponse(void) = 0;

    // Methods for retrieving ReceiveResponse data...
    virtual const char *GetResponse(void) = 0;
    virtual int         GetResponseLength(void) = 0;
    virtual bool        MoreData(void) = 0;

    // Methods for retrieving SendCommand return data...
    virtual const char *GetCommandName(void) = 0;
    virtual long        GetProcessNumber(void) = 0;
    virtual long        GetProcessCount(void) = 0;

// Methods to iterate over error collection ...
    virtual bool        GetCurrentError(void) = 0;
    virtual bool        GetNextError(void) = 0;
    virtual bool        GetPreviousError(void) = 0;
    virtual bool        GetFirstError(void) = 0;
    virtual bool        GetLastError(void) = 0;

    // Methods to retrieve error data...
    virtual const char *GetMsgID(void) = 0;
    virtual int         GetFeedBackCode(void) = 0;
    virtual int         GetReturnCode(void) = 0;
    virtual int         GetStatus(void) = 0;
    virtual const char *GetSubstitute(void) = 0;

    // Method to display error collection...
    virtual int   DisplayError(FILE *Output) = 0;
};

class ConnectDirectSession
{
public:
    // Interface classes
    CDSession *SessionINF;

    ConnectDirectSession();
    ~ConnectDirectSession();
};
```

Writing User Exits

The user exit API functions allow you to write custom programs to use with IBM Connect:Direct. The user exit programs are used by IBM Connect:Direct to invoke user-specific logic at strategic points within IBM Connect:Direct execution. User exit programs must be C or C++ language programs and cannot be shell scripts. The PMGR invokes the Statistics user exit program when you start IBM Connect:Direct and the exit runs as long as IBM Connect:Direct runs. The SMGR invokes the File Open and Security user exits for each session and stops them when the particular session terminates.

Note: exit_skeleton.c and exit_skeleton.C contain working examples of all three exits and can be made with the make_exit_c and make_exit_C make files.

The user exit programs are described in the following:

Program	Description

File Open Exit	IBM Connect:Direct sends a message to this user exit program to open the source or destination file during processing of the copy statement. The File Open Exit opens the source file and identifies the file descriptor. This exit can perform any sort of processing to file names or directory names. It can also redirect the open request to other files as needed.
	
	The File Open Exit program (named “exit_skeleton” in this example) must be owned by root and the setuid bit must be set. Use the following commands:
	
	`% chown root exit_skeleton`
	
	`% chmod u+s exit_skeleton`

Security Exit	The Security Exit enables you to implement your own security system or provide access to a third-party security system.
Statistics Exit	The Statistics Exit is a notification to the user exit program after any record is written to the statistics file. Whenever a statistics record is written to the statistics file, an exact copy is passed to this exit.

User Exit Functions

A connection between the user exit and IBM Connect:Direct is established when the user exit program calls the **exit_child_init()** or **exit_child_init_c()** function. The connection is terminated through a specially designated stop message. The types of messages are defined in the include file user_exit.h. The following functions facilitate communications between the user exit and IBM Connect:Direct:

C++ Function	C Function	Description
exit_child_init()	exit_child_init_c()	Use this function as the first line in a user exit program to initialize communications between IBM Connect:Direct and the user exit program.
recv_exit_msg()	recv_exit_msg_c()	Used by both IBM Connect:Direct and the user exit program to receive a message from the other Process. The receive exit messages wait for a response from the other Process.
send_exit_file()	send_exit_file_c()	The user exit program uses this function when it has opened a file for IBM Connect:Direct. This function uses underlying UNIX methods to pass an open file descriptor. from one Process to another.
send_exit_msg()	send_exit_msg_c()	Both IBM Connect:Direct and the user exit program use this function to send a message to the other Process. Send messages are followed with a receive message to get the response from the other Process.

Initializing Communications with exit_child_init() or exit_child_init_c()

Use the **exit_child_init()** or **exit_child_init_c()** function as the first line of code of the user exit program to initialize communications. This function performs a check to verify that each side is ready to communicate. Following is the format of the **exit_child_init()** function:

```
int exit_child_init( char * logfile )
```

The exit_child_init() or exit_child_init_c() function has the following parameter:

Parameter	Description	Value
logfile	The name of the log or trace file that is opened for use by the user exit programs. Because the file open and security exit are started by SMGR, which is running as root, the exits also run as root. Running the exits as root can cause problems with file permissions of the log file, so **logfile** enables you to easily change owner or permissions on the file. See the sample exit in *d_dir*/ndm/src/exit_skeleton.c for more details.	Name of log file or trace file

The **exit_child_init()** or **exit_child_init_c()** function have the following return codes. Return codes for the function are defined in ndmapi.h.

Return Code	Description
GOOD_RC	Communications between IBM Connect:Direct and the user exit program were successfully initialized.
ERROR_RC	Communications between IBM Connect:Direct and the user exit program could not be initialized.

Waiting for a Message Using recv_exit_msg() or recv_exit_msg_c()

The recv_exit_msg() or recv_exit_msg_c() function waits until it receives a message from IBM Connect:Direct. Control is suspended until a message is received or an error occurs. The recv_exit_msg() has the following format:

```
int recv_exit_msg( int exit_flag )
    int * msg_type,
    char * recv_buf,
    int * recv_buf_len
```

The recv_exit_msg() or recv_exit_msg_c() functions have the following parameters:

Parameter	Description	Value
exit_flag	A flag to specify the recipient ID. The only valid value a user exit program can use is EXIT_PROGRAM.	EXIT_PROGRAM
msg_type	A pointer to the name of the received message. Messages are requests from IBM Connect:Direct and the associated response from the user exit program.	Pointer to message
recv_buf	A pointer to the memory location of the message.	Pointer to message
recv_buf_len	The length in bytes of the message to be received.	Length of message

The recv_exit_msg()or recv_exit_msg_c() functions have the following return codes. Return codes for the function are defined in ndmapi.h.

Return Code	Description
GOOD_RC	The message was received successfully.
ERROR_RC	An error occurred and the message was not received successfully. Possible causes include: IBM Connect:Direct terminated, an invalid value used for the exit_flag parameter, or the receiving buffer not large enough to hold the message received.

Passing a File Descriptor Using send_exit_file() or send_exit_file_c()

Use the send_exit_file() or send_exit_file_c() function to pass a file descriptor from one Process to another Process. Following is the format of send_exit_file():

```
int send_exit_file int  exit_flag
    int fd
```

Following are the parameters for send_exit_file() or send_exit_file_c():

Parameter	Description	Value
exit_flag	A flag to specify the sender ID. The only valid value a user exit program can use is EXIT_PROGRAM.	EXIT_PROGRAM
fd	The file descriptor of a file that the user exit program opened in the place of IBM Connect:Direct, similar to one returned by the open(2) function.	File descriptor

The send_exit_file() or send_exit_file_c() function calls have the following return codes. Return codes for the function are defined in ndmapi.h.

Header	Header
GOOD_RC	The file descriptor was received successfully.
ERROR_RC	An error occurred and the file descriptor was not sent successfully. Possible causes include: IBM Connect:Direct terminated, an invalid value used for the exit_flag or fd parameters, or the last message sent was not send_exit_msg.

Sending a Message to IBM Connect:Direct Using send_exit_msg() or send_exit_msg_c()

The send_exit_msg() or send_exit_msgc() function enables the user exit program to send a message to IBM Connect:Direct. This function returns control to the caller immediately after the message is queued.

Following is the format of the send_exit_msg() function:

```
int send_exit_msg int exit_flag
    int msg_type,
    char * send_buf,
    int send_buf_len
```

Following are the parameters for send_exit_msg() or send_exit_msg_c():

Parameter	Description	Value
exit_flag	A flag to specify the sender ID. The only valid value a user exit program can use is EXIT_PROGRAM.	EXIT_PROGRAM
msg_type	A message name. Messages are requests from IBM Connect:Direct and the associated response from the user exit program.	Pointer to message
send_buf	A pointer to the memory location of the message to be sent.	Pointer to message
send_buf_len	The length in bytes of the message to be sent.	Length of message

Following are the return codes for **send_exit_msg()** or **send_exit_msg_c()**. Return codes for the function are defined in ndmapi.h.

Return Code	Description
GOOD_RC	The message was sent successfully.
ERROR_RC	An error occurred and the message was not sent successfully. Possible causes include: IBM Connect:Direct terminated or an invalid value is used for the exit_flag or msg_type parameters.

Overview of User Exit Messages

IBM Connect:Direct sends and receives messages, using the send_exit_msg() and the **recv_exit_msg()** functions for a C++ program or the **send_exit_msg_c()** and the **recv_exit_msg_c()** functions for a C program. For the exact definition of the data sent in each message, see the files located in *d_dir*/ndm/include/user_exit.h and *d_dir*/ndm/include/user_exit2.h.

Note: The copy control block is defined in user_exit2.h.

Statistics Exit Message

The statistics exit has only one type of message, the STATISTICS_LOG_MSG.

IBM Connect:Direct sends a STATISTICS_LOG_MSG to the user exit program. Every time IBM Connect:Direct writes a statistic record, this message provides an exact copy of the character string. The STATISTICS_LOG_MSG contains the IBM Connect:Direct statistics record.

File Open Exit Messages

The file open exit has four types of messages:

- FILE_OPEN_OUTPUT_MSG
- FILE_OPEN_OUTPUT_REPLY_MSG
- FILE_OPEN_INPUT_MSG
- FILE_OPEN_INPUT_REPLY_MSG

The file open exit has the following limitations:

- The oflag parameter passed to the user exit is already calculated based on the file disposition, as explicitly specified on the copy statement or using the default value. If the user exit changes the oflag to truncate and the original disposition is mod meaning the copy will append to the end of file if the file already exists, then the user exit causes the Process to behave differently from how the Process language is documented.
- If the oflag specifies opening a file with write access and the user exit changes access to read-only, IBM Connect:Direct will fail when it attempts to write to a read-only file.
- The upload and download parameters that restrict directory access are not enforced by Connect:Direct for UNIX when a file open exit is in use. These restrictions are passed into the file open exit and any enforcement action is the responsibility of the file open exit coder.

FILE_OPEN_OUTPUT_MSG

During the copy statement process, IBM Connect:Direct sends a FILE_OPEN_OUTPUT_MSG to the user exit program to open the destination file. The FILE_OPEN_OUTPUT_MSG contains:

- The open function oflag parameter (for example, O_CREAT|O_RDWR|O_TRUNC)
- The open function mode parameter, which controls file permissions
- UNIX user ID that will own the file
- UNIX group ID that will own the file
- UNIX user name
- A copy of the IBM Connect:Direct copy control block
- A copy of the IBM Connect:Direct parsed sysopts structure (the copy control block contains the actual raw version from the process)

FILE_OPEN_OUTPUT_REPLY_MSG

The user exit program sends a reply message to the IBM Connect:Direct FILE_OPEN_OUTPUT_MSG. The FILE_OPEN_OUTPUT_REPLY_MSG contains:

- Status value of zero for successful or non zero for failure
- Status text message (if status value is failure, status text message is included in the error message)
- Pipe pid (for pipe I/O, the UNIX process ID of the shell process that is performing the shell command for pipe I/O)

- Actual file name opened (to be used in statistics log messages)

If the status value is zero for successful, the user exit program must immediately call **send_exit_file()** or **send_exit_file_c()** to send the file descriptor of the opened file to IBM Connect:Direct.

FILE_OPEN_INPUT_MSG

During the copy statement Process, IBM Connect:Direct sends a FILE_OPEN_INPUT_MSG to the user exit program to open the source file. The FILE_OPEN_INPUT_MSG contains:

- The open function oflag parameter (for example, O_RDONLY)
- The open function mode parameter, which controls file permissions
- UNIX user ID that will own the file
- UNIX group ID that will own the file
- UNIX user name
- A copy of the IBM Connect:Direct copy control block
- A copy of the IBM Connect:Direct parsed sysopts structure (the copy control block contains the actual raw version from the Process)

FILE_OPEN_INPUT_REPLY_MSG

This message type is used when the user exit program sends a reply message to the IBM Connect:Direct FILE_OPEN_INPUT_MSG. The FILE_OPEN_INPUT_REPLY_MSG contains:

- Status value of zero for success or non zero for failure
- Status text message (if status value is failure, status text message is included in the error message)
- Pipe pid (for pipe I/O, the UNIX process ID of the shell process that is performing the shell command for pipe I/O)
- Actual file name opened (used in statistics log messages)

Security Exit Messages

The security exit contains four types of messages:

- GENERATE_MSG
- GENERATE_REPLY_MSG
- VALIDATE_MSG
- VALIDATE_REPLY_MSG

⚠ **CAUTION:** If the security exit is used, IBM Connect:Direct relies on it for user ID authentication. If the security exit is not implemented correctly, security can be compromised.

GENERATE_MSG

IBM Connect:Direct sends a generate message to the user exit program at the start of a session to establish a security environment. The PNODE sends the GENERATE_MSG to the security exit to determine a user ID and security token to use for authentication on the SNODE. The GENERATE_MSG contains:

- Submitter ID
- PNODE ID
- PNODE ID password, if user specified one

- SNODE ID
- SNODE ID password, if user specified one
- PNODE name
- SNODE name

GENERATE_REPLY_MSG

The user exit program sends a reply message to IBM Connect:Direct. The GENERATE_REPLY_MSG contains:

- Status value of zero for success or non zero for failure
- Status text message (if status value is failure, status text message is included in the error message)
- ID to use for security context on the SNODE side (may or may not be the same ID as in the generate message)
- Security token used in conjunction with ID for security context on the SNODE side

VALIDATE_MSG

IBM Connect:Direct sends a validate message to the user exit program. The SNODE sends the VALIDATE_MSG to the security exit to validate the user ID and security token received from the PNODE. The VALIDATE_MSG contains:

- Submitter ID
- PNODE ID
- PNODE ID password, if user specified one
- SNODE ID
- SNODE ID password, if user specified one
- PNODE name
- SNODE name
- ID to use with security token
- Security token (password, PASSTICKET, or other security token)

VALIDATE_REPLY_MSG

The user exit program sends a reply message to the IBM Connect:Direct VALIDATE_MSG. The VALIDATE_REPLY_MSG contains:

- Status value of zero for success or non zero for failure
- Status text message (if status value is failure, status text message is included in the error message)
- ID used for security context
- Security token to use in conjunction with ID for security context

User Exit Stop Message

IBM Connect:Direct sends the stop message, STOP_MSG, when all useful work for the user exit is complete and to notify the user exit to terminate. A user exit should terminate only when a stop message is received or if one of the above listed user exit functions returns an error code.

Copy Control Block

The copy control block structure contains the fields, which control how IBM Connect:Direct Processes the copy statement Process file.

Exit Log Files

If user exit programs are specified in the initparm.cfg, IBM Connect:Direct creates exit logs. Exit log files are provided specifically for the user exit programs and are used for debug and trace type messages. The user exit program is started with the log file already opened on STDOUT and STDERR. The exit log files are:

- stat_exit.log
- file_exit.log
- security_exit.log

Note: You can access the log files through the normal **printf()** and **fprintf (stderr,...)** functions.

The log files are located in the installed (d_dir) working directory:

.../*d_dir*/work/cd_node

Chapter 5. Using FASP

IBM Aspera High-Speed Add-on for Connect:Direct for UNIX uses FASP (Fast and Secure Protocol) network transport to transfer files over high bandwidth and high latency network connections.

At low latency it performs similarly to TCP/IP. However, as latency and packet loss increase, unlike TCP/IP, its performance does not degrade, and FASP continues to take advantage of all the available bandwidth.

IBM Aspera High-Speed Add-on for Connect:Direct for UNIX supports interoperability with Connect:Direct for Microsoft Windows and Secure Proxy. Fore more information, refer to High-Speed Add-On Getting Started Guide.

Note: Secure+ is used to secure FASP transfers exactly the same way it is used for TCP/IP transfers.

Related concepts

"Using Connect:Direct for UNIX with IBM Aspera High-Speed Add-on and Secure Proxy (V4.2.0.4 or later)" on page 223
You can send files using IBM Aspera High-Speed Add-on through Secure Proxy using Connect:Direct for UNIX.

Activating FASP

By default, IBM Aspera High-Speed Add-on for Connect:Direct is not enabled. To enable it, you must download a license key and install Connect:Direct for UNIX V4.2.0.4 iFix 13 or later.

1. Download the IBM Aspera High-Speed Add-on for Connect:Direct license key for your Connect:Direct node from Passport Advantage.
2. Rename the file *aspera-license.*
3. Save the renamed file to the <*cd_dir*>/ndm/cfg/<*nodename*> directory.
4. Download and install Connect:Direct for UNIX V4.2.0, Fix Pack 3 (or later) from Fix Central.

   Important: The Connect:Direct install package includes the IBM Aspera High-Speed Add-on for Connect:Direct configuration file (aspera.conf). It contains the minimum necessary basic configuration statements to use FASP on Connect:Direct. It is always installed even if you do not purchase IBM Aspera High-Speed Add-on for Connect:Direct. Do NOT make any changes to this file.

Licensed bandwidth for FASP transactions

The bandwidth available to a file transfer is limited by, among other things, the bandwidths specified in the sender's and receiver's Aspera license keys.

There are two types of available license keys:

- Datacenter licenses (available in 10gbps, 1gbps, 300mbps and 100mbps) - can send and receive files using FASP when connected to a node that has an Endpoint or DataCenter license.
- Endpoint license - can send and receive files using FASP when connected to a node that has a DataCenter license.

Note: Aspera FASP bandwidth setting should not exceed the network capability. As per the available network bandwidth in your network, start with a reasonable setting e.g. 100 mbps. FASP bandwidth setting is likely to max out the resources especially CPU. FASP requires more CPU than normal TCP/IP transfer and secure+ requires even more CPU than Non-Secure+ transfer. In case of high CPU usage (reaching approx. 100%), either add more CPU resources or **reduce the FASP bandwidth setting**.

When both sender and receiver only have Endpoint licenses, file transfer over FASP is not supported. When either the sender or receiver has an Endpoint license and the other has a Datacenter license, the available bandwidth is limited to the value in the Datacenter license. When both sender and receiver have Datacenter licenses, the bandwidth is limited to the smaller of the two values in the Datacenter licenses.

FASP Process Language

Once the FASP parameters for both trading partners have been configured, you can override the default settings on a process by process basis to perform exception processing.

Optional Parameters

FASP Parameters:

- FASP (Yes | No)
- FASP POLICY (Values are the same as the FASP Local and Remote node record parameters)
- FASP.FILESIZE.THRESHOLD (Values are the same as the FASP Local and Remote node record parameters)
- FASP.BANDWIDTH (Values are the same as the FASP Local and Remote node record parameters)

FASP Parameters are applicable in three different contexts:

- COPY statement - The four FASP parameters may be used individually or as a group within a COPY statement. This will set FASP values for the duration of that COPY statement and will not have any effect on statements within the submitted Process
- PROCESS statement - The four FASP parameters may be used individually or as a group at the end of a PROCESS statement. This will set the FASP parameters for all of the COPY statements in the process
- SUBMIT command - The four FASP parameters may be set individually or as a group at the end of a SUBMIT command. This will set the FASP parameters for all COPY statements in the process being submitted These settings will set FASP information for their relevant part of the scope, potentially overriding the Local Node settings, Remote Node settings and each other.

Examples

Copy statement example:

```
step01 copy
from
(
file = /tmp/exampleout
pnode
)
ckpt = 2M
compress extended
fasp=yes
fasp.policy=fixed
fasp.bandwidth=500M
fasp.filesize.threshold=10G
to
(
file = /tmp/examplein
snode
disp = rpl
)
```

Process statement example:

```
SAMPLE PROCESS    SNODE=WINVM-470
fasp=yes
fasp.policy=fixed
fasp.bandwidth=500M
fasp.filesize.threshold=10G
step01 copy
from
(
file = /tmp/exampleout
pnode
)
ckpt = 2M
compress extended
to
(
file = /tmp/examplein
snode
disp = rpl
)
PEND
```

Hierarchy Settings

The system uses the following hierarchy to process overrides:

1. Remote node record overrides local node record.
2. Process parameters override remote node record.
3. Submit statement overrides the process parameters.
4. Each Copy statement overrides the effective settings of the session established by the node settings, Process or Submit statements. The Copy statement override is effective only for the duration of the Copy step.

Using Connect:Direct for UNIX with IBM Aspera High-Speed Add-on and Secure Proxy (V4.2.0.4 or later)

You can send files using IBM Aspera High-Speed Add-on through Secure Proxy using Connect:Direct for UNIX.

FASP is supported in Secure Proxy V3.4.3 or later. If you send a file from your local Connect:Direct for UNIX node configured for FASP, it passes through your Secure Proxy instance using FASP, and is sent to the remote node.

In addition to the FASP parameter values outlined in Configuring FASP, the following parameter should be used when using Secure Proxy between Connect:Direct nodes:

```
fasp=(yes|no|ssp,yes|no|ssp)
```

The first parameter is the default for Connect:Direct as the PNODE. The second parameter is the default for Connect:Direct as the SNODE.

This parameter can now be used in the netmap local node record and remote node trading partner record in Connect:Direct for UNIX.

The following table shows results when Connect:Direct FASP protocol is used between two Connect:Direct nodes with no Sterling Secure Proxy involved.

PNODE fasp=	Protocol	SNODE fasp=
N	TCP	N
N	TCP	Y
N	TCP	SSP

Y	TCP	N
Y	C:D FASP	Y
Y	TCP	SSP
SSP	TCP	N
SSP	TCP	Y
SSP	TCP	SSP

The following table shows results when Connect:Direct FASP protocol is used with two Connect:Direct nodes going through a single instance of Sterling Secure Proxy.

PNODE fasp=	**Protocol**	**SSP**	**Protocol**	**SNODE fasp=**
N	TCP	SSP	TCP	N
N	TCP	SSP	TCP	Y
N	TCP	SSP	TCP	SSP
Y	TCP	SSP	TCP	N
Y	C:D FASP	SSP	C:D FASP	Y
Y	C:D FASP	SSP	TCP	SSP
SSP	TCP	SSP	TCP	N
SSP	TCP	SSP	C:D FASP	Y
SSP	TCP	SSP	TCP	SSP

The following table shows results when Connect:Direct FASP protocol is used with two Connect:Direct nodes going through two instances of Sterling Secure Proxy.

PNODE fasp=	**Protocol**	**SSP**	**Protocol**	**SSP**	**Protocol**	**SNODE fasp=**
N	TCP	SSP	TCP	SSP	TCP	N
N	TCP	SSP	TCP	SSP	TCP	Y
N	TCP	SSP	TCP	SSP	TCP	SSP
Y	TCP	TCP	TCP	SSP	TCP	N
Y	C:D FASP	SSP	C:D FASP	SSP	C:D FASP	Y
Y	C:D FASP	SSP	C:D FASP	SSP	TCP	SSP
SSP	TCP	SSP	TCP	SSP	TCP	N
SSP	TCP	SSP	C:D FASP	SSP	C:D FASP	Y
SSP	TCP	SSP	C:D FASP	SSP	TCP	SSP

For more information on using Sterling Secure Proxy with FASP, see *Using FASP with Sterling Secure Proxy (V3.4.3 or later)*.

Configuring FASP

FASP configuration settings are not added to the Connect:Direct configuration files during install. To enable IBM Aspera High-Speed Add-on for Connect:Direct, you must manually configure the initparm.cfg and netmap.cfg files to run FASP.

1. Do one of the following steps:
   - If you installed Connect:Direct for UNIX V4.2.0.4 as a new installation (you did not upgrade from a previous version), go to Step 2. The initparm.cfg file is already configured for FASP listen ports.
   - If you upgraded from a previous version of Connect:Direct for UNIX to V4.2.0.4, you must configure the initparm.cfg file by specifying a FASP listen port or port ranges.

   Format is `listen.ports=(nnnnn, nnnnn-nnnnn)`.

   Example:

   ```
   # FASP listen ports
   fasp:\
   :listen.ports=(44001, 33002-33005):
   ```

   Note: The number of concurrent FASP processes is limited to the number of ports designated in this file. If you attempt to use more concurrent FASP processes than there are ports available fails, FASP fails.

2. Configure the netmap.cfg file by specifying FASP values for the local node record. Use the following chart.

   Example:

   ```
   local.node:\

   …
   :fasp=yes:\
   :fasp.policy=fair:\
   :fasp.bandwidth=500MB:\
   :fasp.filesize.threshold=2GB:\
   ```

Parameter	Value
fasp	Optional. Default is *no* if the parameter is not present. Enables FASP. • If set to *no*, FASP is disabled. • If set to *yes, yes*, FASP is enabled. This sets the default for all Connect:Direct file transfers. fasp=(pnode value, snode value), for example, fasp=(yes, ssp) • This setting can be overridden by the remote node record or process parameters. • The remote server must have FASP enabled.
fasp.filesize.threshhold	Optional. Used to restrict small files from being sent using FASP. • If the file is greater than or equal to the stated value, the Connect:Direct server sends the file using FASP. Otherwise, it is sent using TCP/IP. • Default is 1GB. • You can use KB, MB, or GB designators. If no designator is included, the system uses bits.

Parameter	Value
	• This setting can be overridden by the remote node record or process parameters.
fasp.bandwidth	Optional. Default is as stipulated in the FASP license key. Specifies how much bandwidth each transfer can use. • Default value can be changed, but cannot exceed the bandwidth specified in the license key. • You can use KB, MB, or GB designators. If no designator is included, the system uses bits per second. • This setting can be overridden by the remote node record or process parameters, but cannot exceed the bandwidth specified in the license key.
fasp.policy	Optional. Specifies the fairness of each transfer. Default is *fair*. • This setting can be overridden by the remote node record or process parameters. • Valid values are: – Fixed - FASP attempts to transfer at the specified target rate, regardless of the actual network capacity. This policy transfers at a constant rate and finishes in a guaranteed amount of time. This policy typically occupies a majority of the network's bandwidth, and is not recommended in most file transfer scenarios. – Fair - FASP monitors the network and adjusts the transfer rate to fully utilize the available bandwidth up to the maximum rate. When other types of traffic build up and congestion occurs, FASP shares bandwidth with other traffic fairly by transferring at an even rate. This is the best option for most file transfer scenarios. – High - FASP monitors the network and adjusts the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, a FASP session with high policy transfers at a rate twice of a session with fair policy. – Low - Similar to Fair mode, the Low (or Trickle) policy uses the available bandwidth up to the maximum rate as set in the Aspera license file. When congestion occurs, the transfer rate is decreased all the way down to the minimum rate as set in the Aspera license file.

3. (Optional) Configure the netmap.cfg file by specifying FASP values for the remote node record. Use the following chart. Configure the remote node if you need to override your local node settings. For example, if you want to exclude a trading partner from using FASP. You can also configure the remote node record later.

   Example:

```
myRmtNodePartner:\

     …
          :fasp=yes:\
          :fasp.policy=fair:\
          :fasp.bandwidth=1GB:\
          :fasp.filesize.threshold=1GB:\
```

Parameter	**Value**
fasp	Optional. Valid values are *yes* and *no*. Enables FASP. • If set to *no*, files sent to this remote node will not use FASP. • If set to *yes*, files sent to this remote node will default to use FASP instead of TCP/IP. • This setting can be overridden by the process parameters. • The remote server must have FASP enabled.
fasp.filesize.threshhold	Optional. Used to restrict small files from being sent using FASP. • If the file is greater than or equal to the stated value, the Connect:Direct server sends the file using FASP. Otherwise, it is sent using TCP/IP. • Default is 1GB. • You can use KB, MB, or GB designators. If no designator is included, the system uses bits. • This setting can be overridden by the process parameters.
fasp.bandwidth	Optional. Default is as stipulated in the FASP license key. Specifies how much bandwidth each transfer can use. • Default value can be changed, but cannot exceed the bandwidth specified in the license key. • You can use KB, MB, or GB designators. If no designator is included, the system uses bits per second. • This setting can be overridden by the process parameters, but cannot exceed the bandwidth specified in the license key.
fasp.policy	Optional. Specifies the fairness of each transfer. Default is *fair*. • This setting can be overridden by the process parameters.

Parameter	Value
	• Valid values are: – Fixed - FASP attempts to transfer at the specified target rate, regardless of the actual network capacity. This policy transfers at a constant rate and finishes in a guaranteed amount of time. This policy typically occupies a majority of the network's bandwidth, and is not recommended in most file transfer scenarios. – Fair - FASP monitors the network and adjusts the transfer rate to fully utilize the available bandwidth up to the maximum rate. When other types of traffic build up and congestion occurs, FASP shares bandwidth with other traffic fairly by transferring at an even rate. This is the best option for most file transfer scenarios. – High - FASP monitors the network and adjusts the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, a FASP session with high policy transfers at a rate twice of a session with fair policy. – Low - Similar to Fair mode, the Low (or Trickle) policy uses the available bandwidth up to the maximum rate as set in the Aspera license file. When congestion occurs, the transfer rate is decreased all the way down to the minimum rate as set in the Aspera license file.

FASP Messages

Use the following table to obtain FASP error message information.

Note: Long text message files for these message IDs can be viewed using the Connect:Direct Requester Message Lookup utility.

Non-Detailed Statistics Mode (Message ID only)	Detailed Statistics Mode
FASP001E	FASP001E: FASP server session creation failed.
FASP002E	FASP002E: FASP client session creation failed.
FASP003E	FASP003E: FASP could not be initialized.
FASP004E	FASP004E: Lock timeout.
FASP005E	FASP005E: Memory allocation failure.
FASP006E	FASP006E: Condition wait timed out.
FASP007E	FASP007E: No FASP listen ports available.
FASP008E	FASP008E: FASP disabled due to file size &FILESIZE < threshold &THRESHOLD

Non-Detailed Statistics Mode (Message ID only)	Detailed Statistics Mode
FASP009E	FASP009E: FASP session terminated unexpectedly.
FASP010E	FASP010E: SNODE refused FASP, FASP disabled.
FASP011E	FASP011E: FASP CRC verification failed.
FASP020E	FASP020E: Session Manager received invalid FASP control message.
FASP021E	FASP021E: FASP control message fragmented or invalid.
FASP022E	FASP022E: Session Manager failed to receive FASP control message.
FASP023E	FASP023E: The FASP control message to send exceeds the buffer size.
FASP024E	FASP024E: Session Manager failed to send FASP control message.
FASP030E	FASP030E: FASP license file not found.
FASP031E	FASP031E: FASP license file expired.
FASP032E	FASP032E: FASP license in error.
FASP033E	FASP033E: FASP license is malformed.
FASP034E	FASP034E: FASP license is malformed.
FASP035E	FASP035E: FASP License file at &LOCATION will expire in &VALUE day(s).
FASP040E	FASP040E: FASP initialization failed - remote &TYPE &NODE. Error=&ERROR.
FASP041E	FASP041E: FASP initialization failed - local &TYPE &NODE. Error=&ERROR.
FASP042E	FASP042E: FASP initialization failed.

Monitoring FASP transactions

You can view the Copy Termination Record (CTRC) for detailed statistics. For example, you can verify FASP was used, what bandwidth was used, and which policy was used.

In the example below, note the following explanations:

- FASP=>Y indicates that FASP was used to transfer this file. FASP=>N would indicate TCP/IP was used.
- FSPL=>FAIR is the policy negotiated for this file transfer.
- FSBW=>1000000000 is the bandwidth negotiated for this file transfer (in bits per second).
- FMBC =>2 is the high water mark for the number of FASP buffers used
- FBCS =>16777216 is the FASP buffer size
- FSTH =>1073741824 is filesize threshold
- FSLP =>23708 is listen port used for FASP

Example:

```
PROCESS RECORD Record Id => CTRC
Completion Code => 0
Message Id => SCPA000I
Short Text => Copy step successful.
```

```
Ckpt=>Y Lkfl=>N Rstr=>N Xlat=>N Scmp=>N Ecmp=>N CRC=>N
FASP=>Y  FSPL=>FAIR  FSBW=>10000000000  FMBC=>2  FBCS=>16777216 FSTH=>1073741824
FSLP=>23708
```

Limitations

The following features cannot be used with FASP and Connect:Direct for UNIX:

- Firewall navigation source ports should not be used with FASP
- Silent installation does not support the FASP configuration parameters

Chapter 6. Using object store providers with IBM Connect:Direct for UNIX

IBM Connect:Direct for UNIX can be configured to extend support to object storage providers including IBM Cloud Object Storage, Microsoft Azure Blob, Google Storage, AWS and S3 compatible providers like Minio, Dell EMC ECS, and Red Hat Ceph to execute public and on-premise cloud-based operations. Users can now continue using the benefits of Connect:Direct features like security, reliability, and point-to-point file transfers optimized for high-volume delivery along with versatility that comes with an object storage backend.

The Linux platform supports managed file transfers between the node and the object store. It is strongly recommended that Connect:Direct be installed as close as the object store storage devices as possible for high performance, consistency, and reliability. It is theoretically possible to remotely access object stores, but it is strongly discouraged as performance frequently suffers due to inconsistent access times to storage resources.

To set up accounts, instances and storage on cloud providers contact your IT Administrator.

A IBM Connect:Direct for UNIX node running this release could either be located on-premise or be running on an instance on the cloud. The user can also configure both the Pnode and Snode on two instances on Cloud. An object store can serve as a source or as a destination to send and receive files.

Setting up Connect:Direct Node on object store providers

By default, cloud support for Connect:Direct is not enabled. To enable cloud support, complete the following tasks:

- Pre-requisites to activate Connect:Direct Unix on cloud provider. For more information, refer to Account (amazon.com), Get started with Google Cloud|Documentation, IBM Cloud Docs, Azure documentation| Microsoft Docs
- Installing Connect:Direct Unix node on cloud.
- Configuring Connect:Direct node for object storage

Connect:Direct for UNIX can also be configured to extend support to other S3 object store providers such as Minio, Dell EMC ECS, and Red Hat Ceph or use dedicated endpoints to execute public and on-premise cloud-based operations. For endpoint configuration/override see the dedicated section.

Pre-requisites to set-up Connect:Direct Unix on cloud provider

Before you configure Connect:Direct node definitions necessary for using Connect:Direct for UNIX, you must complete the following tasks:

1. Set up cloud accounts and credentials
2. Select and create a compute instance, RedHat or SuSE
3. Create IAM user/roles
4. Create Security group. Port numbers which are specific to Connect:Direct should be added to the security group
5. Create storage
6. Obtain credentials for cloud storage object access

For more information see, Account (amazon.com), Get started with Google Cloud | Documentation, IBM Cloud Docs, Azure documentation | Microsoft Docs

Installing Connect:Direct Unix node on Cloud

No specific configuration is required to install Connect:Direct for UNIX node on a compute instance. For information to install Connect:Direct for UNIX see, "Installing Connect:Direct for UNIX" on page 18.

If you are upgrading from an old release of Connect:Direct for UNIX node note that:

- CD Unix, Linux platform, and JRE are now included in the base installation
- Initparms to be included during the S3 plugin configuration are updated during the upgrade process

Setting up Connect:Direct Node for object store providers

Store Object naming and initparm.cfg

Objects on object stores can be read and written with Connect:Direct.

A file name identified with an URI type format is an object in an object store.

```
scheme://bucketOrContainer/objectKey
```

The scheme in the Connect:Direct process file name is used to search for the right entry in the `initparm.cfg` file using the name field in a file.ioexit section as the key.

Connect:Direct for UNIX can read and/or write to the following object stores:

- Amazon S3
- IBM Cloud Object Storage
- Azure Blob
- Google Storage

Each of them is named an object provider. Each time a process involves an object from an object store, the right provider must be triggered. This is performed using properties set in the process or in the `initparm.cfg` file.

Selecting the right Provider

Main property

Store provider selection is triggered by the value of the `store.providerName` property (See "Stores Properties" on page 236). The following names are available and valid:

- Amazon S3: S3
- IBM Cloud Object Storage: COS
- Azure Blob: AZ
- Google Storage: GS

S3 is the default value and can be omitted if Amazon S3 is the expected provider.

Each provider owns its subset of properties. Use these properties to fine configure this provider usage for credentials, endpoints, and objects properties.

Using initparm.cfg file

`Initparm.cfg` file include a section dedicated to store providers. The `file.ioexit` section identifies a behavior to adopt thru a set of properties. More than one entry can exist and each of them can define a different provider and/or a different behavior for the same provider.

Note: Any colon characters (':') in a parameter value must be escaped with a backslash ('\').

```
# Azure
file.ioexit:\
 :name=AZ:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m \
 -Dstore.providerName=AZ \
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:

# Amazon S3 Production
file.ioexit:\
 :name=S3Prod:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m -Ds3.profileName=profileProd \
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:

# Amazon S3 QA
file.ioexit:\
 :name=S3QA:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m -Ds3.profileName=profileQA \
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:
```

Using sysopts

Connect:Direct sysopts can also be used to select the right store provider. All properties can be overridden using `sysopts`, the `store.providerName` can also be overridden.

```
# All-purpose entry, default to S3
file.ioexit:\
 :name=ALL:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m \
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:


#Write to an Azure container using the all-purpose entry
To (
FILE=ALL://container/object
sysopts=':store.providerName=AZ:az.connectionString='aconnectionstring':'
DISP=RPL

#Write to an S3 bucket using the all-purpose entry
To (
FILE=ALL://container/object
sysopts=':s3.accessKey=…:s3.secretKey=…:'
DISP=RPL
```

Setting up CA certificates used for secure connections to an object store

The default place where the Java JRE looks for CA certificates is `[CDU_DIR]/jre/ibm-java-x86_64-80/jre/lib/security/cacerts`. When CAs must be added, replaced in this file, there is a risk they are overridden when a Connect:Direct update is applied and the cacerts file replaced.

To avoid this situation, it's possible to use the Connect:Direct Secure Plus key store as a replacement or a complement to the JRE keystore.

Without any action on configuration, the JRE keystore remains the only source for secure connections validation. To activate a different behavior it's necessary to set a new property in configuration thru initparms.cfg file or process sysopts.

The property is store.keyStore and the possible values for this property are:

- JRE_ONLY (default)
- SP_ONLY: The secure Plus keystore will be used as the unique source for CAs
- JRE_SP: the JRE keystore is the first source for CAs, next Secure Plus keystore will be used
- SP_JRE: the Secure Plus keystore is the first source for CAs, next the JRE keystore will be used

Migrating to a JRE only to a SP only configuration

It's recommended to first set the property to SP_JRE, move from JRE keystore or install the necessary CAs in the Secure Plus keystore, next set the property to SP_ONLY when all the necessary CAs are in the Secure Plus keystore.

```
# A scheme using the JRE keystore only (default)
file.ioexit:\
 :name=JRE:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m \
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:

# A scheme using the JRE keystore and next the Secure Plus keystore
file.ioexit:\
 :name=JRESP:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m -Dstore.keyStore=JRE_SP\
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:
```

```
# A scheme using the Secure Plus keystore
file.ioexit:\
 :name=SP:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m -Dstore.keyStore=SP_ONLY\
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:

#This process overrides the default behavior for scheme SP
#Only the JRE CAs will be used

To (
FILE=SP://container/object
sysopts=':store.keyStore=JRE_ONLY:'
DISP=RPL
)
```

Credentials

Credentials are not managed identically and depend on the selected store provider. See "Stores Properties" on page 236 for properties definitions.

Azure Blob

Credentials are managed in the following order:

1. Connection String (az.connectionString). Connection String includes endpoint.
2. StorageSharedKeyCredential using account name and account key (az.accountName , az.accountKey) using calculated endpoint. For more information, refer to "Stores Properties" on page 236.
3. SAS token (az.sasToken)
4. Managed Identity (az.managedIdentityId)
5. Workload Identity (az.workloadIdentityId, optional: az.workloadTenantId, az.workloadServiceTokenFilePath) – Only available if running inside Azure
6. Environment variables credentials

Google Storage

Only the Google Account generated json credential file can be used. Set property gs.credentialsPath to locate this file.

IBM Cloud Object Storage

Credentials are managed in the following order:

1. Json credentials file path (cos.credentialsPath)
2. BasicIBMOAuthCredentials using Api key and service Id (cos.apiKey, cos.serviceId)
3. BasicAWSCredentials using hmac access key and secret key (cos.hmacAccessKey, cos.hmacSecretKey)
4. ProfileCredentialsProvider using profile path and profile name (cos.profilePath, cos.profileName)
5. The default credentials provider chain
   a. Environment Variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
   b. Java System Properties aws.accessKeyId and aws.secretKey
   c. JSON credential file at the default location (~/.bluemix/cos_credentials)
   d. Web Identity Token credentials from the environment or container.
   e. Credential profiles file at the default location (~/.aws/credentials)
   f. Credentials delivered through the Amazon EC2 container service if AWS_CONTAINER_CREDENTIALS_RELATIVE_URI" environment variable is set and security manager has permission to access the variable
   g. Instance profile credentials delivered through the Amazon EC2 metadata service

Amazon S3

Credentials are managed in the following order:

1. AwsBasicCredentials using hmac access key and secret key (s3.accessKey, s3.secretKey)
2. ProfileCredentialsProvider using profile path and profile name (s3.profilePath,s3.configPath,s3.profileName)
3. The default credentials provider chain
   a. Environment Variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
   b. Java System Properties aws.accessKeyId and aws.secretKey
   c. Web Identity Token credentials from the environment or container
   d. Credential profiles file at the default location (~/.aws/credentials)
   e. Credentials delivered through the Amazon EC2 container service if AWS_CONTAINER_CREDENTIALS_RELATIVE_URI" environment variable is set and security manager has permission to access the variable.
   f. Instance profile credentials delivered through the Amazon EC2 metadata service

Using S3 Role

It is possible to use the role arn mechanism directly inside profiles or specify it thru properties. When provided thru properties, a Secure Token Service client will be created to get temporary credentials the same way the profile mechanism does.

To provide the role properties, the following properties can be used:

- s3.roleArn
- s3.roleProfile
- s3.roleDuration (optional)

Credentials refresh

When a profile entry is updated in either the credentials or config file, whatever they are located in their default location or in a particular location (using s3.profilePath, s3.configPath), credentials will be validated again.

When refreshed, credentials may abort the current process if they became invalid.

Endpoints

Endpoints often have a default value but can be overridden with provided properties. It is sometime necessary to provide more info thru properties to have the endpoint correctly calculated.

Azure Blob

- Endpoint info are provided (az.endpoint*): endpoint is built with provided values
- From connection string if provided
- No endpoint info provided: endpoint is derived from account name using the following pattern: https://{az.accountName}.blob.core.windows.net if no connection string is provided.

Google Storage

Endpoint info provided (`gs.endpoint*`): endpoint is built using the provided values..

IBM Cloud Object Storage

- Endpoint info provided (cos.endpoint*): The endpoint is constructed using the specified values and may include location if provided (cos.location)
- No endpoint info provided: endpoint is derived from endpoint type and location name using the following pattern: `https://s3.{cos.endpointtype}[.{cos.location}].cloud-object-storage.appdomain.cloud`.

Amazon S3

- Endpoint info provided (s3.endpoint*): endpoint is built using the provided values.

  If `s3.useFipsEndpoint`=YES, the `s3.endpoint*` values are ignored.

Amazon S3 bucket arn and access points

A bucket can be located either with its name or its arn. In a process the following form are equivalent and supported:

- Classic URI form: S3://bucketname/objectKey
- Access point: S3://arn:aws:s3:region:****:accesspoint/bucketaccesspointName/objectKey
- Access point Alias: S3://bucketaccesspoint-****-s3alias/objectKey
- Multi region access point: S3://arn:aws:s3::****:accesspoint/*********.mrap/objectKey

In the previous examples, S3 was used for the scheme name, but any value can be used as long as scheme is declared.

Stores Properties

Usage

All properties can be used and set in initparm and/or sysopts. Any value set in sysopts will override value set in initparm if present.

```
# IO Exit parameters for Azure

file.ioexit:\
 :name=AZ1:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m \
 -Dstore.providerName=AZ -Dstore.tags='key1=AnotherValue\:key2=value2' \
 -Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:
Sysopts override in CD process

sysopts=':store.tags='key1=AnotherValue\:key2=value2':'
```

```
Sysopts override in CD process

sysopts=':store.tags='key1=AnotherValue\:key2=value2':'
```

`store.tags` value overrides value defined in `initparm.`

Property names are case sensitive when used in initparm but not case sensitive when set in `sysopts`.

If a property value contains “:” or “=”, value must be enclosed in quotes, and the ":" must be escaped with a backslash, "\".

```
propertyName='tag=abc\:error=true'
```

General properties

These properties are available for all stores providers. The **`store.providerName`** is the most important of them and will trigger the right provider.

Property name (, alternate name for compatibility)	Description	Possible values	Default value	Connect Direct	Integrated File Agent
store.providerName	Triggers the right store service	S3: Amazon S3 AZ: Azure Blob GS: Google Storage COS: IBM Cloud Object Storage	S3 for compatibility with previous version	YES	YES
store.keyStore	Keystore usage	JRE_ONLY: The cacerts file will be used SP_ONLY: Only Secure Plus keystore will be used JRE_SP: The cacerts file and next the Secure Plus keystore SP_JRE : The Secure Plus keystore and next the cacerts file	JRE_ONLY	YES	YES

Property name (, alternate name for compatibility)	Description	Possible values	Default value	Connect Direct	Integrated File Agent
store.configFromCD	Integrated file agent will get the store configuration from Connect:Direct and will not use the stores.properties content	YES NO	NO	NO, Only for Integrated File Agent	YES
store.contentType, s3ioexit.contentType	Object Content-Type	Free	None	YES	NO
store.contentEncoding	Object Content-Encoding For compatibility with previous version: if *.contentType contains 'charset', charset value will be used for Content-Encoding but only if store.contentEncoding is empty	Free	None	YES	NO
store.dwldRange, s3ioexit.dwldRange	Size of buffer to read from the provider stream	>= 5MB, <= 50MB	5MB	YES	NO
store.objectSize, s3ioexit.objectSize	Object size If CD can't provide it. This value can be used to calculate part size on multi parts uploads.	S3: up to 5TB AZ: up to 4.78TB GS: up to 5TB COS: up to 5TB	None	YES	NO

Property name (, alternate name for compatibility)	Description	Possible values	Default value	Connect Direct	Integrated File Agent
store.partSize, s3ioexit.partSize	Override to calculated part size	S3: not less than 5MB, up to 5GB AZ: not less than 64KB, up to 100MB GS: not less than 5MB, up to 5TB COS: not less than 5MB, up to 5GB	None	YES	NO
store.tags	Additional info to store with the object as tags or metadata.	S Tags must be enclosed into quotes and separated with semicolon. store.tags='key=value;otherKey=abc'	None	YES	NO
store.maxConnections	For multipart uploads, the maximum number of parallel connections the client can use. This value is not fixed or guaranteed and may vary depending on the size of the uploaded parts and available system resources.	integer	30	YES	NO
store.endpointUrl	Endpoint Override This value can be overridden by any `*.endpointUrl` value if specified.			YES	YES

Property name (, alternate name for compatibility)	**Description**	**Possible values**	**Default value**	**Connect Direct**	**Integrated File Agent**
store.endpointPort	Endpoint port Override This value can be overridden by any `*.endpointPort` value if specified.			YES	YES
store.endpointSecure	Endpoint will https or http This value can be overridden by any `*.endpointSecure` value if specified.	YES, NO	YES	YES	YES

Azure Blob properties (az.*)

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
az.connectionString	The connection string includes the full set of info to connect to the service. Value must be enclosed in quotes.	Example: az.connectionString='DefaultEndpointsProtocol= https;AccountName= cduioexit;AccountKey=abcd1r4BIZQlahie2V3cFqTg==; BlobEndpoint= https\:// cduioexit.blob.core.windows.net/; QueueEndpoint=https\:// cduioexit.queue.core.windows.net/; TableEndpoint= https\:// cduioexit.table.core.windows.net/; FileEndpoint= https\:// cduioexit.file.core.windows.net/'	None	YES	YES
az.applicationId	Additional info application can provide	Free	None	YES	YES
az.accountName	Credentials account name	provided by Azure account	None	YES	YES
az.accountKey	Credentials account key	provided by Azure account	None	YES	YES
az.sasToken	Credentials SAS token	provided by Azure account	None	YES	YES
az.workloadIdClientId	Credentials managed Identity client ID	provided by Azure account	None	YES	YES
az.workloadIdClientId	Credentials Workload Identity client ID	provided by Azure account	None	YES, only when running on Azure	YES, only when running on Azure

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
az.workloadTenantId	Workload tenant ID	provided by Azure account	None	YES, only when running on Azure	YES, only when running on Azure
az.workloadService TokenFilePath	File path to the service token file for workload identity	provided by Azure account	None	YES, only when running on Azure	YES, only when running on Azure
az.endpointUrl	Endpoint info to override default endpoint. Mainly used when using Azurite (Use Azurite emulator for local Azure Storage development | Microsoft Docs)		None	YES	YES
az.endpointPort	Endpoint port		None	YES	YES
az.endpointSecure	Endpoint will use https or http	YES | NO	YES	YES	YES
az.accessTier	Object storage class	HOT, COOL, ARCHIVE	None (inferred from bucket)	YES	NO

Google Storage properties (gs.*)

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
gs.credentialsPath	Path to the json credentials file	Provided by Google account	None	YES	YES
gs.projectId	Additional info application can provide	Free	None	YES	YES
gs.storageClass	Object storage class	STANDARD, NEARLINE, COLDLINE, ARCHIVE	None (inferred from bucket)	YES	NO

Property name	**Description**	**Possible values**	**Default value**	**Connect Direct**	**Integrated File Agent**
gs.partUploadFolder	GCP SDK does not provide an API for multipart uploads as other cloud providers do. Instead, CD creates parts with unique names and then composes them into the final object. Unlike other cloud providers, these parts are not hidden, which means scanning tools may detect them as separate objects. To address this issue, a new property is now available to store these temporary parts in a dedicated folder. When this property is enabled, parts are stored in the specified folder using the following naming pattern:`container/'partUploadFolderValue'/objectKey.uniqueId.Part.n`	A valid folder name	None	YES	NO

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
gs.composeDelay	The Object Store Service uses the compose API provided by the Google SDK to merge uploaded parts into the final object. This API has a rate limit of one call per second for an object, and in some cases, two consecutive calls may be too fast. To address this issue, a delay and retry mechanism have been implemented when this error occurs. Delay applied for two consecutive part uploads.	milliseconds	1000	YES	NO
gs.composeRetries	See gs.composeDelay Number of retry attempts when a delay error occurs.	integer	10	YES	NO
gs.endpointUrl	Endpoint override			YES	YES
gs.endpointPort	Endpoint port override			YES	YES
gs.endpointSecure	Endpoint will use https or http	YES, NO	YES	YES	YES

IBM Cloud Object Storage properties (cos.*)

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
cos.credentialsPath	Path to the json credentials file		None	YES	YES
cos.serviceInstanceId	CredentialsService Instance ID		None	YES	YES
cos.apiKey	Credentials API key		None	YES	YES
cos.hmacAccessKey	AWS S3 credentials hmac access key		None	YES	YES
cos.hmacSecretKey	AWS S3 credentials hmac secret key		None	YES	YES
cos.profilePath	AWS S3 credential file with hmac keys and profiles		None	YES	YES
cos.profileName	AWS S3 credential file with hmac keys, profile name to use	Profile names available in credentials file	default	YES	YES
cos.endpointUrl	Endpoint override		None	YES	YES
cos.endpointPort	Endpoint port override		None	YES	YES
cos.endpointSecure	Endpoint will use https or http	YES, NO	YES	YES	YES
cos.location	Data Center location: used to dynamically build endpoint (when not overridden by cos.endpoint*)	See Locations for resource deployment | IBM Cloud Docs		YES	YES
cos.endpointType	Endpoint Type, used to dynamically build endpoint (when not overridden by cos.endpoint*)	DIRECT, PRIVATE, PUBLIC	PUBLIC	YES	YES

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
cos.storageClass	Object storage class	Accelerated, DeepArchive, Glacier, IntelligentTiering, OneZoneInFrequentAccess, Standard, StandardInFrequentAccess	Inferred from bucket	YES	NO
cos.sseS3	Server side encryption requested	YES, NO	NO	YES	NO
cos.virtualHostedUri	Endpoint format for bucket access, path style access (https://xxx.com/bucket-name/key-name) or virtual hosted (https://bucket-name.xxx.com/key-name) Only works without endpoint override.	YES, NO	YES	YES	YES

Amazon S3 properties (s3.*)

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
s3.accessKey	AWS S3 credentials hmac access key	provided by Amazon account	None	YES	YES
s3.secretKey	AWS S3 credentials hmac secret key	provided by Amazon account	None	YES	YES
s3.roleArn	Role arn to assume	provided by Amazon account	None	YES	YES
s3.roleProfile	Role profile with credentials	provided by Amazon account	None	YES	YES

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
s3.roleDuration	Role duration in seconds	From 900 to 43200	None	YES	YES
s3.profilePath	AWS S3 credential file with hmac keys and profiles		None	YES	YES
s3.configPath	AWS S3 credential additional config file		None	YES	YES
s3.profileName	AWS S3 credential file with hmac keys, profile name to use	Profile names available in merged credentials file and config file	default	YES	YES
s3.region	AWS region	Will be retrieve from profile If not provided	None	YES	YES
s3.endpointUrl	Endpoint override			YES	YES
s3.endpointPort	Endpoint port override			YES	YES
s3.endpointSecure	Endpoint will use https or http	YES, NO	YES	YES	YES
s3.storageClass	Object storage class	Deep_Archive, Glacier, Glacier_IR Intelligent_Tiering, OneZone_IA, Outposts, Reduced_Redundancy, Standard, Standard_IA	Inferred from bucket	YES	NO
s3.sseS3	Server side encryption requested with SSE-S3	YES, NO	NO	YES	NO

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
s3.virtualHostedUri	Endpoint format for bucket access, path style access (https://xxx.com/bucket-name/key-name) or virtual hosted (https://bucket-name.xxx.com/key-name) Only works without endpoint override.	YES, NO	YES	YES	YES
s3.useFipsEndpoint	S3 FIPS endpoints must be used. See FIPS - Amazon Web Services (AWS) for more details.	YES, NO	NO	YES	YES

Property name	Description	Possible values	Default value	Connect Direct	Integrated File Agent
s3.proxyScheme	Default S3 http clients proxy scheme is HTTP. Only system properties http_proxyHost, http_proxyPort, http_proxyUser, http_proxyPassword or environment variable HTTP_PROXY can be set to establish a non-secure connection to a proxy. For a proxy secure connection, system properties https_proxyHost, https_proxyPort, https_proxyUser, https_proxyPassword or environment variable HTTPS_PROXY, proxy scheme must be "HTTPS" but this value can't be set thru a system property. s3.proxyScheme allows this override.	HTTP, HTTPS	HTTP	YES	YES

Logging

The logging for the basic object store plugin in Connect:Direct SMGR will be captured in the Connect:Direct SMGR trace when it is enabled. See Running System Diagnostics for instructions to enable this trace.

The object store plugin will also create its own log file via the following properties specified in the initparm.cfg file.ioexit record(s):

- -Ds3ioexit.trace_level=level \
  - Default level is INFO
  - Debug level is activated with level=DEBUG
- -Dcdioexit_trace_destination=file
  - file is the word ‘file’ and not a file path
  - log file will be created in cduser home directory
  - log file name is objectstorepluginlogs.log

Example:

```
file.ioexit:\
:name=S3:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
:home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m -Ds3ioexit.trace_level=DEBUG \
-Dcdioexit_trace_destination=file \
 -Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:
```

Enhancing logging

If a file with name `log4j2.properties` is stored in `/opt/cdunix/ndm/ioexit-plugins/s3`, logging will use this properties file. Connect Direct installation creates a default `log4j2.properties` file with logging level set to INFO. There are more loggers than the basic configuration. These loggers should be enabled at the DEBUG level on support request.

Override Mechanism

Irrespective of the store provider, the mechanism for overriding properties is consistent. A property can be defined at the `initparms.cfg` level and/or in system options (`sysopts`).

For the integrated File Agent, properties are specifically defined in the `config/stores.properties` file and cannot be overridden. Unlike other components, there is no provision for using sysopts options with the Integrated File Agent.

Properties defined in `initparms.cfg` can be overridden at the process level using `sysopts` statements, providing enhanced flexibility in both provider and process definitions.

Example 1: Only one entry in initparms.cfg but multiple providers

```
# A general definition

file.ioexit:\
 :name=ALL:\
 :library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
 :home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
 :options=-Xmx640m \
 -Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:
```

Process overrides the store.providerName property (with default S3). The scheme ALL is a multi-store scheme.

```
Process, truncated
Copy from S3 to Azure same bucket/container object

FROM (
FILE=ALL://container/object
sysopts=':store.providerName=S3:s3.accessKey=….:s3.secretKey=……:'
DISP=RPL )

)
```

```
TO (
FILE=ALL://container/object
sysopts=’:store.providerName=AZ:az.connectionString=’aconnectionstring’:’
DISP=RPL )
```

Example 2: Two entries in initparms.cfg for 2 providers, 2 sources, 1 destination

```
# Provider 1 – Amazon S3 default credentials
file.ioexit:\
:name=FROMS3:\
:library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
:home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
:options=-Xmx640m
-Dstore.providerName=S3
-Ds3.accessKey=… -Ds3.secretKey=… \
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:
# Provider 2 – Azure
file.ioexit:\
:name=TOAZ:\
:library=/opt/cdunix/ndm/lib/libcdjnibridge.so:\
:home.dir=/opt/cdunix/ndm/ioexit-plugins/s3:\
:options=-Xmx640m -Dstore.providerName=AZ\
-Daz.connectionString='...' \
-Djava.class.path=/opt/cdunix/ndm/ioexit-plugins/s3/cd-s3-ioexit.jar
com.aricent.ibm.mft.connectdirect.s3ioexit.S3IOExitFactory:
```

```
Process 1, truncated
Copy from S3 to Azure same bucket/container object
Nothing in sysopts, all properties come from initparms definition

FROM (
FILE=FROMS3://container/object
DISP=RPL )

)
TO (
FILE=TOAZ://container/object
DISP=RPL )

Process 2, truncated
Copy from another S3 bucket with other credentials to Azure same bucket/container object
anothercontainer needs other credentials, sysopts overrides the default

FROM (
FILE=FROMS3://anothercontainer/object
sysopts=’:s3.accessKey=…:s3.secretKey=…:’
DISP=RPL )

)
TO (
FILE=TOAZ://container/object
DISP=RPL )
```

Understanding properties origin

When logging is enabled, log shows properties origin.

Connect:Direct example

```
Properties in initparms.cfg file are provided thru system properties (-D…). So, label for
origin is SystemProperties.

Available property from CDProcessSysopts s3.configpath:/pathto/config-east-1
Available property from SystemProperties s3.accesskey:****
Available property from SystemProperties store.providername:S3
Available property from CDProcessSysopts s3.profilepath:/path/aws_credentials
Available property from SystemProperties s3.secretkey:****
```

Integrated File Agent example (IFA is the client)

```
IFA requested to use CD initparms properties. Label for origin is CDInitparms.

Available property from CDProcessSysopts s3.configpath:/pathto/config-east-1
Available property from CDInitparms      s3.accesskey:****
Available property from CDInitparms      store.providername:S3
Available property from CDProcessSysopts s3.profilepath:/path/aws_credentials
Available property from CDInitparms      s3.secretkey:****
Available property from Client           store.configfromcd:YES
```

Integrated File Agent

The Integrated File Agent can utilize the same set of properties, excluding those specific to object storage class or encryption. These properties are usually configured in the `config/stores.properties` file located in the Integrated File Agent installation directory.

Integrated File Agent executing on X86_64 Linux

When executed on X86_64 Linux, Integrated File Agent can take advantage of the `initparms.cfg` file to avoid configuration duplication.

In the `stores.properties` file, the conventional format is `cdfa.provider.setName=property=value`, where 'setName' serves as an identifier for grouping properties in a specific set.

The following example shows two groups of properties. The first one points to Azure, the second one uses Google Storage as provider.

```
cdfa.provider.AZURE=store.providerName=AZ
cdfa.provider.AZURE=scheme=MICROSOFT://
cdfa.provider.AZURE=az.connectionString=…

cdfa.provider.GOOGLE=store.providerName=GS
cdfa.provider.GOOGLE=scheme=GOOGLE://
cdfa.provider.GOOGLE=gs.credentialsPath=PathTo/google_credentials.json
cdfa.provider.GOOGLE=gs.projectId=ExampleProject
```

As Integrated File Agent is designed to submit processes to Connect:Direct, it's crucial that Connect:Direct includes schemes MICROSOFT and GOOGLE defined in `initparms.cfg` with the corresponding credentials properties set. Failure to provide these definitions will result in process failure.

To use the `initparms.cfg` definitions, just set property `store.configFromCD=YES` in the groups expecting credentials or properties from Connect:Direct.

```
cdfa.provider.AZURE=scheme=MICROSOFT://
cdfa.provider.AZURE=store.configFromCD=YES
```

Group AZURE for scheme MICROSOFT will get credentials or other properties from Connect:Direct. Any other properties in the group will be ignored.

The `scheme` property remains mandatory. The `scheme` property is used to associate the watched directories with an entry inside the `stores.properties` file. The outscheme property can also be used to override the name of object submitted to Connect:Direct. When used, this 'outscheme' must also be defined in initparms.cfg file

Limitations

- When running Connect:Direct for UNIX with an object store be aware of the following limitations:

  Instance is static, not elastic.
- For Run Task/Job operations on cloud objects, the object store CLI for the cloud in question should be invoked. UNIX system commands, such as mv or cp, referencing cloud objects are not supported.
- Depending on the cloud provider, the maximum object size which can be transferred may vary. Refer to each provider to identify the object size.
- The user should consider the cost associated with the usage of cloud resources.
- Storage object does not support the option MOD (modify) for files that are transferred via Connect:Direct. Only NEW or RPL(replace) is supported.

- Wildcard Copy sent from the cloud is not supported.
- If there are multiple versions of the same object present in a bucket/container then only the latest version can be downloaded and not any previous version.
- Multipart download is not supported. Multipart upload is supported.
- For multipart uploads, a user should consider creating a lifecycle rule to delete incomplete parts for aborted and not restarted uploads. Parts remain on storage and involve an additional cost.
- Checkpoint restart can be explicitly configured within a copy step through the ckpt parameter. If it is not configured in the copy step, it can be configured in the Initparms through the ckpt.interval parameter. For more information, refer to Getting Started Guide.

Chapter 7. Introduction to Connect:Direct Secure Plus

The Connect:Direct Secure Plus for UNIX application provides enhanced security for IBM Connect:Direct and can be optionally configured. It uses cryptography to secure data during transmission. You select the security protocol to use with Connect:Direct Secure Plus.

Introduction to Connect:Direct Secure Plus

The Connect:Direct Secure Plus for UNIX application provides enhanced security for IBM Connect:Direct and can be optionally configured. It uses cryptography to secure data during transmission. You select the security protocol to use with Connect:Direct Secure Plus.

Security Concepts

Cryptography is the science of keeping messages private. A cryptographic system uses encryption keys between two trusted communication partners. These keys encrypt and decrypt information so that the information is known only to those who have the keys.

There are two kinds of cryptographic systems: symmetric-key and asymmetric-key. Symmetric-key (or secret-key) systems use the same secret key to encrypt and decrypt a message. Asymmetric-key (or public-key) systems use one key (public) to encrypt a message and a different key (private) to decrypt it. Symmetric-key systems are simpler and faster, but two parties must somehow exchange the key in a secure way because if the secret key is discovered by outside parties, security is compromised. Asymmetric-key systems, commonly known as public-key systems, avoid this problem because the public key may be freely exchanged, but the private key is never transmitted.

Cryptography provides information security as follows:

- **Authentication** verifies that the entity on the other end of a communications link is the intended recipient of a transmission.
- **Non-repudiation** provides undeniable proof of origin of transmitted data.
- **Data integrity** ensures that information is not altered during transmission.
- **Data confidentiality** ensures that data remains private during transmission.

Secure Plus UNIX Video Tutorials

You can view video tutorials about the installation, configuration, troubleshooting, and other technical features of Connect:Direct Secure Plus for UNIX.

The Connect:Direct Secure Plus videos are useful for Connect:Direct administrators. These tutorials provide a quicker way to access information and remove the need to reference the Connect:Direct Secure Plus documentation library.

Click the link below to access the Connect:Direct Secure Plus for UNIX video channel to view tutorials about the following topics:

- Installation
- Configuration
- Troubleshooting

The Connect:Direct Secure Plus UNIX video channel can be found at this link:Connect:Direct Secure Plus for UNIX Video Channel.

Protocol Support

Before you configure Connect:Direct Secure Plus, you must determine the protocol that you and your trading partners will use to secure communications sessions. For planning information, see Plan Your Implementation of the SSL or TLS Protocol.

Transport Layer Security Protocol (TLS)

The TLS protocol use certificates to exchange a session key between the node that initiates the data transfer process (the primary node, or PNODE) and the other node that is part of the communications session (the secondary node, or SNODE). A certificate is an electronic document that associates a public key with an individual or other entity. It enables you to verify the claim that a given public key belongs to a given entity. Certificates can be self-issued (self-signed) or issued by a certificate authority (CA). See Self-Signed and CA-Signed Certificates for details on the differences between self-signed and CA-issued certificates.

When a CA receives an application for a certificate, the CA validates the applicant's identity, creates a certificate, and then digitally signs the certificate, thus vouching for an entity's identity. A CA issues and revokes CA-issued certificates.

Self-signed certificates are created and issued by the owner of the certificate, who must export the certificate in order to create a trusted root file that includes this certificate and supply the trusted root file to the partner in a connection.

TLS 1.3

TLS 1.3 adds new features for improved security. For more information, see *RFC 8446, section 1.2 Major Differences from TLS 1.2*.

Deprecated Protocols

SSL3.0, TLS 1.0 and TLS 1.1 protocols are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.

If deprecated protocols are required, TLS 1.3 should not be enabled in the trading partner's configuration, otherwise the handshake may fail. Deprecated protocols should be exclusively configured per node.

The Secure+ feature continues to support SSL 3.0, TLS 1.0 and TLS 1.1.

NIST SP800-131a and Suite B support

Connect:Direct supports a new standard from The National Institute of Standards and Technology (NIST), SP800-131a to extend the current FIPS standards, as well as Suite B cryptographic algorithms as specified by the National Institute of Standards and Technology (NIST).

The government of the Unites States of America produces technical advice on IT systems and security, including data encryption and has issued Special Publication SP800-131a that requires agencies from the Unites States of America to transition the currently-in-use cryptographic algorithms and key lengths to new, higher levels to strengthen security.

Applications must use strengthened security by defining specific algorithms that can be used and what their minimum strengths are. These standards specifies the cryptographic algorithms and key lengths that are required in order to remain compliant with NIST security standards.

To comply with the new requirements, IBM products with cryptographic functionality must:

- Enable TLS 1.2 and be prepared to disable protocols less than TLS 1.2
- Cryptographic keys adhere to a minimum key strength of 112 bits
- Digital signatures are a minimum of SHA-2

The following is included in Secure Plus for NIST SP800-131a and Suite B support:

- Support TLS 1.1 and 1.2 with SHA-2 cipher suites
- Support for SP800-131a transition and strict modes
- Support for NSA Suite B 128 and 192 bit cipher suites and modes
- Support for IBM CMS Keystore
- Support migrating existing Secure+ certificates to the IBM CMS Keystore
- Support for JRE 1.7 SR1 iKeyman/iKeycmd utilities for certificate management.

For more information on NIST security standards, see http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf.

For more information on Suite B security standards, see http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml

Connect:Direct Secure Plus Tools

Connect:Direct Secure Plus consists of five components:

- Connect:Direct Secure Plus Administration Tool (Secure+ Admin Tool)
- Parameters file (Secure+ parameters file)
- Access file (Secure+ access file)
- Strong Password Encryption Parameters file
- Connect:Direct Secure Plus Command Line Interface (Secure+ CLI).

The following sections describe these components and their function within Connect:Direct Secure Plus.

Note: Only one instance of the Secure+ Admin Tool or the Secure+ CLI may be used at a time because they access the same configuration file. Do not open these tools at the same time or multiple copies of the same tool at the same time (two instances of Secure+ Admin Tool or two instances of Secure+ CLI). Only the user who accessed the configuration file first will be able to save updates.

Secure+ Admin Tool

The Secure+ Admin Tool is a graphical user interface (GUI) that enables you to configure and maintain the Sterling Connect:Direct Secure Plus environment. The Secure+ Admin Tool is the interface for creating and maintaining the Secure+ parameters file; operating system utilities and editing tools cannot be used to create or update this file.

Note: To use the Secure+ Admin tool, the server where Connect:Direct for UNIX is installed must have X11 forwarding enabled and a *DISPLAY* variable defined that references the system with an X server installed where the GUI will be displayed.

Secure+ Parameters File

The Connect:Direct Secure Plus parameters file (Secure+ parameters file) contains information that determines the protocol and encryption method used during encryption-enabled Connect:Direct Secure Plus operations. To configure Connect:Direct Secure Plus, each site must have a Secure+ parameters file that contains one local node record and at least one remote node record for each trading partner who uses Connect:Direct Secure Plus to perform a secure connection. The local node record defines the most commonly used security and protocol settings for the node at the site. The local node record can also be used as a default for one or more remote node records. Each remote node record defines the specific security and protocol settings used by a trading partner. You should create a remote node record in the Secure+ parameters file for each Connect:Direct node that you communicate with even if the remote node does not use Connect:Direct Secure Plus.

Note: The Secure+ parameters file is not dynamically updated. When multiple users update the Secure+ parameters file, each user must close and reopen the file to display new records added by all sources.

When you create the Secure+ parameters file, a record named .SEAServer is automatically added to the file, which enables Connect:Direct to interface with Sterling External Authentication Server during TLS session. External authentication is configured in this record and enabled/disabled in the local and remote node records.

With v6.1, Connect:Direct Secure Plus support to cache certificate validation responses from External Authentication Server when it interfaces External Authentication Server during a TLS session. This minimizes the overhead associated with requesting certificate validation from External Authentication Server, thus eliminating the need for Connect:Direct Secure Plus to query External Authentication Server each time. External Authentication Server response caching feature is disabled by default. To enable it see, "Update the Sterling External Authentication Server Record" on page 286 and "Configure External Authentication in the .SEAServer Record" on page 273.

For additional security, the Secure+ parameters file is stored in an encrypted format. The information used for encrypting and decrypting the Secure+ parameters file (and private keys) is stored in the Secure+ access file.

Secure+ Access File

The Connect:Direct Secure Plus access file (Secure+ access file) is generated automatically when you create the Secure+ parameters file for the first time. You type a passphrase when you first initialize Connect:Direct Secure Plus. This passphrase is used to generate the keys necessary to encrypt and decrypt the entries in the Secure+ parameters file. The passphrase itself is not retained.

Your Connect:Direct Secure Plus administrator must secure the Secure+ access file (*<cdinstall>*/ndm/secure+/nodes/.cdspacf).The administrator must have full create and update permissions to update this file. The Connect:Direct server must have read authority. To maintain a secure Secure+ access file, the general user community should not have access permission. This file can be secured with any available file access restriction tool. Availability of the Secure+ access file to unauthorized personnel can compromise the security of data exchange.

Strong Password Encryption Parameters File

Strong Password Encryption protects Connect:Direct passwords which may be specified in a Connect:Direct Process by encrypting the Process when it is submitted and stored in the Connect:Direct work area. Strong Password Encryption uses the AES 256 encryption algorithm. Strong Password Encryption parameters are stored in the parameters file (*<cdinstall>*/ndm/secure+/nodes/.Password). This feature is enabled by default. For more information on using this feature, refer to Configure Strong Password Encryption.

Connect:Direct Secure Plus Command Line Interface

The Java-based Connect:Direct Secure Plus Command Line Interface (Secure+ CLI) is provided to enable you to create customized scripts that automate implementing Connect:Direct Secure Plus. Sample UNIX scripts are provided as models for your customized scripts. You can save these scripts with another name, modify them to reflect your environment, and distribute them throughout your enterprise. For more information about using the Secure+ CLI, commands and parameter descriptions, and the scripts, see Automate Setup with the Secure+ CLI.

Before You Begin

Before you configure the IBM Connect:Direct environment for secure operations, ensure that you complete the following tasks:

- Identifying Expert Security Administrator
- Assessing Security Requirements of Trading Partners
- Planning Your Implementation of IBM Connect:Direct
- Completing the Worksheets

Identifying Expert Security Administrator

The instructions and information provided to assist you in implementing IBM Connect:Direct assume that you have an expert UNIX security administrator who is familiar with your company's security environment. Identify who this individual is within your company and work with this individual as you plan your Connect:Direct implementation.

Assessing Security Requirements of Trading Partners

Security planning is a collaborative effort between you and your trading partners. You must know the expectations of your trading partners and plan your security implementation to meet those requirements.Contact your trading partners to gather the information necessary to coordinate your implementation of IBM Connect:Direct.

Planning Your Implementation of IBM Connect:Direct

After you have identified your security administrator and determined the security requirements of your trading partners, review the following information:

- "Plan Your Implementation of the TLS Protocol" on page 259
- "Set Up Connect:Direct Secure Plus" on page 262

Completing the Worksheets

Before you configure IBM Connect:Direct, complete the worksheets in Configuration Worksheets. Use this information to configure the local and remote nodes to use Connect:Direct.

Plan Your Implementation of the TLS Protocol

Before you configure Connect:Direct Secure Plus, review the following concepts, requirements, and terms to ensure that you have all the resources and information necessary to implement the Transport Layer Security (TLS) protocol.

Overview of the TLS Protocol

The TLS protocol provides three type of authentication:

- During the first type of authentication, called server authentication, the site initiating the session (PNODE) requests a certificate from its trading partner (SNODE) during the initial handshake. The SNODE returns its ID certificate (read from its KeyStore) and the PNODE authenticates it using one or more trusted root certificates stored in its KeyStore. Root certificates are signed by a trusted source—either a public certificate authority, such as Thawte, or by the trading partner acting as its own CA. If the ID certificate from the SNODE cannot be validated using any root certificate found in the KeyStore, or if the root certificate has expired, the PNODE terminates the session. IBM Connect:Direct writes entries to the statistics logs of both nodes and the session is aborted.
- The second type of authentication, called client authentication, is optional. If this option is enabled in the SNODE's IBM Connect:Direct parameters file definition for the PNODE, the SNODE will request a certificate from the PNODE and authenticate it using the information in its KeyStore. If this authentication fails, the SNODE terminates the session and IBM Connect:Direct writes information about the failure to the statistics log of both nodes.
- The third type of authentication is also optional and consists of validating the certificate common name. This authentication is enabled when the security administrator specifies the common name (CN) expected to be contained in the ID certificate to be validated in its IBM Connect:Direct Parameters file.
  - During the first type of authentication, the PNODE compares the common name it has specified for the SNODE in its IBM Connect:Direct Parameters file with the common name contained in the certificate sent by the SNODE. If the compare fails, that is, the information is not identical, the PNODE

terminates the session, and IBM Connect:Direct writes information about the failure to the statistics logs of both nodes.

– During the second type of authentication, the SNODE compares the common name it has specified for the PNODE in its IBM Connect:Direct Parameters file with the common name contained in the certificate sent by the PNODE. If the compare fails, that is, the information is not identical, the SNODE terminates the session, and IBM Connect:Direct writes information about the failure to the statistics logs of both nodes.

Self-Signed and CA-Signed Certificates

Determining the type of certificate to use for secure communications sessions and the method to generate the certificate is challenging. Self-signed certificates and digital certificates issued by certificate authorities offer advantages and disadvantages. You may also be required to use both types of certificates, depending on the security requirements of your trading partners. The following table compares the advantages and disadvantages of self-signed and CA-signed certificates:

Type of Certificate	Advantages	Disadvantages
Self-signed certificate	No cost	Requires you to distribute your certificate, minus the private key, to each trading partner in a secure manner
	Easy to generate	Difficult to maintain; anytime the certificate is changed, it must be distributed to all clients
	Self-validated	Not validated by a third-party entity
	Efficient for small number of trading partners	Inefficient for large number of trading partners
CA-signed certificate	Eliminates having to send your certificate to each trading partner	Trading partners must download digital CA-signed certificate used to verify the digital signature of trading partner public keys.
	No changes are required on the trading partner's system if you recreate the CA digitally-signed certificate using the same CA	Must be purchased from third-party vendor

Terminology for TLS Certificates

The following defines the security terms associated with TLS certificates and communication sessions. The terms are listed in alphabetical order.

CA-signed certificate

Digital document issued by a certificate authority that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. An identity certificate issued by a CA is digitally signed with the private key of the certificate authority.

Certificate (also known as digital certificate, public key certificate, digital ID, or identity certificate)

Signed certificate that is obtained from a certificate authority by generating a certificate signing request (CSR). It typically contains: (1) distinguished name and public key of the server or client; (2) common name and digital signature of the certificate authority; (3) period of validity (certificates expire and must be renewed); and (4) administrative and extended information. The certificate authority analyzes the CSR fields, validates the accuracy of the fields, generates a certificate, and sends it to the requester.

A certificate can also be self-signed and generated by any one of many tools available, such as OpenSSL. These tools can generate a digital certificate file and a private key file in PEM format, which you can combine using any ASCII text editor to create a key certificate file.

Certificate authority (CA)

An organization that issues digitally-signed certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them. The CA digital signature is assurance that anybody who trusts the CA can also trust that the certificate it signs is an accurate representation of the certificate owner.

Certificate signing request (CSR)

Message sent from an applicant to a CA in order to apply for a digital identity certificate. Before creating a CSR, the applicant first generates a key pair, keeping the private key secret. The CSR contains information identifying the applicant (such as a directory name in the case of an X.509 certificate), and the public key chosen by the applicant. The CSR may be accompanied by other credentials or proofs of identity required by the certificate authority, and the certificate authority may contact the applicant for further information.

Cipher suite

A cryptographic key exchange algorithm that enables you to encrypt and decrypt files and messages with the TLS protocol.

Client authentication

A level of authentication that requires the client to authenticate its identity to the server by sending its certificate.

Key certificate file

File that contains the encrypted private key and the ID (public key) certificate. This file also contains the certificate common name that can be used to provide additional client authentication.

Passphrase

Passphrase used to access the private key.

Private key

String of characters used as the private, “secret” part of a complementary public-private key pair. The symmetric cipher of the private key is used to sign outgoing messages and decrypt data that is encrypted with its complementary public key. Data that is encrypted with a public key can only be decrypted using its complementary private key.

The private key is never transmitted and should never be shared with a trading partner.

Public key

String of characters used as the publicly distributed part of a complementary public-private key pair. The asymmetric cipher of the public key is used to confirm signatures on incoming messages and encrypt data for the session key that is exchanged between server and client during negotiation for a TLS session. The public key is part of the ID (public key) certificate. This information is stored in the key certificate file and read when authentication is performed.

Self-signed certificate

Digital document that is self-issued, that is, it is generated, digitally signed, and authenticated by its owner. Its authenticity is not validated by the digital signature and trusted key of a third-party certificate authority. To use self-signed certificates, you must exchange certificates with all your trading partners.

Session key

Asymmetric cipher used by the client and server to encrypt data. It is generated by the SSL software.

Trusted root certificate file (also known as root certificate file

File that contains one or more trusted root certificates used to authenticate ID (public) certificates sent by trading partners during the IBM Connect:Direct protocol handshake.

Set Up Connect:Direct Secure Plus

Before you can configure the node definitions that are necessary for using Connect:Direct Secure Plus, you must complete the following tasks:

- Install Connect:Direct Secure Plus
- Starting the Secure+ Admin Tool
- Populating the Secure+ Parameters File

Install Connect:Direct Secure Plus

You can install Connect:Direct Secure Plus using the Connect:Direct installation script. For more information on installing Connect:Direct Secure Plus, see the *IBM Connect:Direct for UNIX Getting Started Guide*.

Note: After Connect:Direct Secure Plus is installed, the system administrator is responsible for securing access to the Secure+ Admin Tool, Secure+ CLI, and Secure+ parameters files. The Connect:Direct Secure Plus administrator and IBM Connect:Direct Server need full permission to the Connect:Direct Secure Plus directory; no other users require access.

Starting the Secure+ Admin Tool

Use the Connect:Direct Secure Plus Administration Tool (Secure+ Admin Tool) or the Connect:Direct Secure Plus Command Line Interface (Secure+ CLI) to set up and maintain a Connect:Direct Secure Plus operation. This section provides instructions on using the Secure+ Admin Tool. Refer to Automate Setup with the Secure+ CLI, for instructions on using the Secure+ CLI.

To start the Secure+ Admin Tool on a UNIX system, type the following command at the UNIX command prompt from within the ndm/bin directory:

```
spadmin.sh
```

The Secure+ Admin Tool starts and opens the Secure+ parameters file for the associated IBM Connect:Direct node.

Note: The Secure+ parameters file is not dynamically updated. When multiple users update the Secure+ parameters file, each user must close and reopen the file to display new records added by all sources.

Accessing Secure+ Admin Tool Help

Note: Secure+ Admin documents are available on the Knowledge Center. If you access the Secure+ Admin Tool Help from the Secure+ Admin Tool Help menu and the following error message displays:

```
java.lang.UnsupportedOperationException: The BROWSE action is not supported on the current
platform!
```

Be aware of the following limitation:

- This error is connected to launching of your web browser with java browse method.
- The BROWSE action is not supported on all LINUX desktops but works on gnome and KDE.

Populating the Secure+ Parameters File

To communicate with a trading partner using Connect:Direct Secure Plus, you define a node record for that partner in *both* the IBM Connect:Direct network map and the Connect:Direct Secure Plus parameters file (Secure+ parameters file). To set up the Connect:Direct Secure Plus environment, you can populate the Secure+ parameters file from entries defined in an existing network map.

When you populate the Secure+ parameters file from the network map, a record is automatically created in the Secure+ parameters file for each node entry in the network map. Initially, the .Local node record is disabled, and all other records are set to default to local.

Perform the following steps to populate the Secure+ parameters file with node entries defined in the IBM Connect:Direct network map:

1. From the **Secure+ Admin Tool Main Window**, click the **Sync with Netmap** option of the **File** menu item.

   The **Available Netmaps** dialog box is displayed.
2. Navigate to the netmap.cfg file located in the *d_dir*/ndm/cfg/node_name directory. Select the netmap to open and click **Sync**. The **Select Netmap Entries to Add** dialog box is displayed.
3. Click **Add All.**

   The **Select Parameters File Entries to Delete** dialog box is displayed.
4. Click **Skip** to close the Secure+ parameters file without deleting any entries.

   The Secure+ parameters file is populated and the **Secure+ Admin Tool Main Window** displays remote node records in the Secure+ parameters file including the records you added from the network map.

Node Configuration Overview

Before you begin using Connect:Direct Secure Plus, you must configure nodes for secure operations.

When you import the network map records into the **Secure+ parameters file**, Connect:Direct Secure Plus parameters are disabled. To configure the nodes for Connect:Direct Secure Plus, complete the following procedures:

- Import existing Certificates.
- Configure or create a new CMS Key Store through the Key Management menu on the Secure+ Admin Tool.
- Configure the Connect:Direct Secure Plus .Local node record

  Define the security options for the local node. Because TLS and SSL provide the strongest authentication with easy-to-maintain keys, configure the local node for one of these protocols. Determine which protocol is used by most trading partners and configure the local node with this protocol.
- Disable remote nodes that do not use Connect:Direct Secure Plus
- Customize a remote node for the following configurations:
  - To use a unique certificate file to authenticate a trading partner
  - To use a different self-signed or CA-signed certificate for client or server authentication
  - To identify a unique cipher suite used by a trading partner
  - To activate common name validation
  - To activate client authentication
  - To enable FIPS 140-2 mode
  - To activate external authentication
- Configure all remote nodes that use a protocol that is not defined in the local node

  When you configure the local node, all remote nodes are automatically configured to the protocol defined in the local node. If a trading partner uses a different protocol, you must turn on the protocol in the remote node record. For example, if you activate the TLS protocol in the .Local node record and a trading partner uses the SSL protocol, configure the SSL protocol in the remote node record for the trading partner.

Import Existing Certificates

Before performing your .Local node configuration, you need to import existing certificates.

To import existing certificates:

1. Import existing certificates, either keycerts or trusted root files from trading partners into the Key Store. On the Secure+ Admin Tool main window, from the Key Management menu, select **Configure Key Store**. The Key Store Manager window appears.
2. Verify the CMS Key Store path. If incorrect, click **browse**to locate the Key Store path. The Browse CMS KeyStore File window appears.
3. The default Key Store name is: cdkeystore.kdb To locate the default Key Store path, navigate to the Key Store file.

   ```
   Windows path: <cdinstalldir>\Server\Secure+\Certificates\cdkeystore.kdb
   Unix path:  <cdinstalldir>/ndm/secure+/certificates/cdkeystore.kdb
   ```
4. Click **Import**. On the Import PEM KeyStore File window, navigate to and select the certificate file you want to use and click **OK**.
5. If a key certificate file is being imported, the password must be entered. The KeyStore Password window appears. Type your password and click OK.
6. The PEM Certificate Viewer displays to allow a review of the certificate file. Verify the certificate is valid and click the **Import** button. Import Results window displays with status of imported certificate. Click **Close**.
7. The certificate is imported and given a Label based on the certificate Common Name, (CN=). Note the serial number to identify the correct certificate after import.

   Note: A common name is used for Label and identification which means that multiple certificates can have the same common name and therefore, can be overwritten depending on the setting of the Default Mode. Additionally, the Default Mode of Import is Add or Replace Certificates.
8. Click **OK** to create the new CMS KeyStore file. Key Store Manager will display contents of the new keystore.

Create CMS Key Store

Before performing your .Local node configuration, you may need to create a new CMS Key Store file.

To create a new CMS Key Store file:

1. On the Key Store Manager window, click **New**. The Create new CMS KeyStore File dialog box appears.
2. Enter the Directory location (you can also Browse to the location desired), the KeyStore file name, and the password for the new KeyStore file. You can also choose to Populate with standard certificate authorities. This will import all standard public CA Root certificates into the new KeyStore file.
3. Click **OK** to create the new CMS KeyStore file. Key Store Manager will display contents of the new keystore.
4. Click **Import**. On the Import PEM KeyStore File window, navigate to and select the certificate file you want to use and click **OK**.
5. If a key certificate file is being imported, the password must be entered. The KeyStore Password window appears. Type your password and click OK.
6. The PEM Certificate Viewer displays to allow a review of the certificate file. Verify the certificate is valid and click the **Import** button. Import Results window displays with status of imported certificate. Click **Close**.
7. The certificate is imported and given a Label based on the certificate Common Name, (CN=). Note the serial number to identify the correct certificate after import.

   Note: A common name is used for Label and identification therefore multiple certificates can have the same common name and therefore, can be overwritten depending on the setting of the Default Mode. Additionally, the Default Mode of Import is Add or Replace Certificates.

Configuring the Connect:Direct Secure Plus .Local Node Record

Before you can configure the .Local node record, you must either import your existing certificates or create and configure a CMS Key Store. For additional information, see Import Existing Certificates or Create CMS Key Store in the documentation library.

It is recommended that you configure the .Local node record with the protocol used by most of your trading partners. Because remote node records can use the attributes defined in the .Local node record, defining the .Local node record with the most commonly used protocol saves time. After you define the protocol in the .Local node record, all remote nodes default to that protocol. Also, identify the trusted root file to be used to authenticate trading partners.

To configure the local node, refer to the Local Node Security Feature Definition Worksheet that you completed for the .Local node record security settings and complete the following procedure:

Note: In Connect:Direct for UNIX 6.2, when Integrated File Agent is installed, if Secure+ .Local record has not already been assigned a key certificate, an automatically generated self-signed certificate is assigned to Secure+'s .Local record.

1. From the Secure+ Admin Tool Main Window, double-click the .Local record. The Edit Record dialog box displays the Security Options tab, the node name, and the type of node.
2. Set the Security Options for the local or remote node entry you are configuring and if necessary, modify the time-out value in **Authentication Timeout**.

   Refer to the following table for an explanation of the Security Options boxes:

   Note: The SSL3.0, TLS 1.0 and TLS 1.1 protocols are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2. If deprecated protocols are required, TLS 1.3 should not be enabled in the trading partner's configuration, otherwise the handshake may fail. Deprecated protocols should be exclusively configured per node. The Secure+ feature continues to support SSL 3.0, TLS 1.0 and TLS 1.1.

Field Name	Field Definition	Valid Values
Node Name	Specifies the node record name.	.Local This is not an editable field.
Base Record	Specifies the name of the base record. If an alias record is selected, the base record name is displayed in this box.	Name of the local Connect:Direct node.
Type	Specifies the current record type.	Local for a local record and Remote for a remote record. This is not an editable field.
Disable Secure+	Disables Connect:Direct Secure Plus.	Default value is Disable Secure+. Note: If this option is selected, override is enabled, and no remote node definition exists for the remote node in the Connect:Direct Secure Plus parameters file, Connect:Direct Secure Plus is bypassed.

Field Name	Field Definition	Valid Values
Enable SSL 3.0 Protocol	Enables SSL protocol to ensure that data is securely transmitted. The SSL3.0 is deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.	The default value is Disable Secure+.
Enable TLS 1.0 Protocol	Enables TLS protocol to ensure that data is securely transmitted. TLS1.0 is deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.	The default value is Disable Secure+.
Enable TLS 1.1 Protocol	Enables TLS protocol to ensure that data is securely transmitted. The TLS1.1 is deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.	The default value is Disable Secure+.
Enable TLS 1.2 Protocol	Enables TLS protocol to ensure that data is securely transmitted.	The default value is Disable Secure+.
Enable TLS 1.3 Protocol	Enables TLS protocol to ensure that data is securely transmitted.	The default value is Disable Secure+.
Disable	Disables the ability to override values in the .Local node record with values in the remote node record.	The default value is Disable.
FIPS 140-2	Enables FIPS 140-2 security.	The default value is Disable.
SP800-131A Transition	Enables NIST SP800-131a security in transition mode.	The default value is Disable.
SP800-131A	Enables NIST SP800-131a security mode.	The default value is Disable.
Suite B 128 bit	Enables Suite B 128 bit security.	The default value is Disable.
Suite B 192 bit	Enables Suite B 192 bit security.	The default value is Disable.

Field Name	Field Definition	Valid Values
Node or Copy Statement Override	There are several types of overrides. For both PNODE and SNODE, this parameter indicates whether Remote Node record parameters will override the .Local Node record parameters or not. If it is set to No, or if set to Yes and there is no correlating Remote Node record for a given session, then: • For PNODE, this parameter indicates whether process overrides, which may optionally be specified in Process, Submit, and Copy statements, will be allowed. • For SNODE, this parameter indicates whether: – The Secure+ protocol specified by the PNODE will be allowed to override that specified by the SNODE. – To allow unsecured incoming sessions to proceed.	The default value is No.
Authentication Timeout	Specifies maximum time, in seconds, that the system waits to receive the Connect:Direct Secure Plus blocks exchanged during the Connect:Direct Secure Plus authentication process. If you specify a value of 0, Connect:Direct waits indefinitely to receive the next message. Specify a time to prevent malicious entry from taking as much time as necessary to attack the authentication process.	A numeric value equal to or greater than 0, ranging from 0 to 3600. The default is 120 seconds.

3. Click the **TLS Options** tab. The **TLS Options** dialog box is displayed.
4. Select an existing Key Certificate from the key store. To select a Key Certificate from the keystore, click **Browse** next to **Key Certificate Label**. The **CMS KeyStore Certificate Viewer** appears.

   Note: You must add or import the key certificate into your key store prior to configuring your node. For additional information, see Import Existing Certificates or Create CMS Key Store in the documentation library. For additional information on how to use iKeyman, see http://www-01.ibm.com/support/knowledgecenter/SSYKE2_6.0.0/com.ibm.java.security.component.60.doc/security-component/ikeyman_overview.html?lang=en.
5. In the Key Certificates area, select the key certificate you want to use and click **OK** box.
6. Select cipher suites from the below list:

Name
TLS_AES_256_GCM_SHA384
TLS_AES_128_GCM_SHA256
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_128_CCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_NULL_SHA
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_NULL_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256

Name
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_NULL_SHA
TLS_RSA_WITH_NULL_MD5
The following are marked as deprecated and security warnings are issued when enabled.
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA
TLS_ECDHE_ECDSA_WITH_NULL_SHA
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_NULL_MD5
TLS_ECDHE_RSA_WITH_NULL_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_NULL_SHA

7. Click the **External Authentication** tab. The **External Authentication** dialog box is displayed.
8. Choose one of the following options:
   - To enable external authentication on the remote node, click **Yes** in the **Enable External Authentication** box.
   - To disable external authentication on the remote node, click **No**.
9. Type the Certificate Validation Definition character string defined in External Authentication Server.
10. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Customize Remote Node Records

After you configure the .Local node record, Connect:Direct Secure Plus enables the protocol and parameters that you configured for the local node for all remote node records. If all trading partners use the protocol and configuration defined in the .Local node record, you are now ready to begin using Connect:Direct Secure Plus.

However, even when a trading partner uses the same protocol as the one defined in the .Local node record, you may need to customize remote node records for the following configurations:

- Using a unique certificate file to authenticate a trading partner—During a TLS session, a certificate enables the PNODE to authenticate the SNODE. You identified a certificate in the .Local node record. If you want to use a unique certificate to authenticate a trading partner, you must identify this information in the remote node record.
- Using a self-signed certificate file to authenticate a trading partner—During a TLS session, a certificate enables the PNODE to authenticate the SNODE. If you want to use a self-signed certificate to authenticate a trading partner, you must identify this information in the remote node record.
- —Client authentication requires that the SNODE validate the PNODE. If you want to enable client authentication, activate this feature in the remote node record.
- Activating common name authentication—If you want another layer of security, you can activate the ability to validate the certificate common name by specifying the common name expected to be in the identity certificate received, either by the PNODE from the SNODE, or, when client authentication is enabled, by the SNODE from the PNODE.
- Identifying the cipher suite used by a trading partner—When configuring the TLS protocol, you enable cipher suites that are used to encrypt the transmitted data. When communicating with a trading partner, you and the trading partner must use the same cipher suite to encrypt data. If the trading partner does not enable a cipher suite that is enabled in your configuration, communication fails. If necessary, enable cipher suites in the remote node record.

Configuring a Remote Node Record

Before you can configure the .Remote node record, you must either import your existing certificates or create and configure a CMS Key Store. For additional information, see Import Existing Certificates or Create CMS Key Store in the documentation library.

Configure the Remote node record with the protocol used by most of your trading partners. Because remote node records can use the attributes defined in the Remote node record, defining the Remote node record with the most commonly used protocol saves time. After you define the protocol in the Remote node record, all remote nodes default to that protocol. Also, identify the trusted root file to be used to authenticate trading partners.

To configure the local node, refer to the Local Node Security Feature Definition Worksheet that you completed for the Remote node record security settings and complete the following procedure:

1. From the Secure+ Admin Tool Main Window, double-click the .Remote record. The Edit Record dialog box displays the Security Options tab, the node name, and the type of node.
2. Set the Security Options for the local or remote node entry you are configuring and if necessary, modify the time-out value in **Authentication Timeout**.

   Refer to the following table for an explanation of the Security Options boxes:

   Note: SSL3.0, TLS 1.0 and TLS 1.1 protocols are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2. If deprecated protocols are required, TLS 1.3 should not be enabled in the trading partner's configuration, otherwise the handshake may fail. Deprecated protocols should be exclusively configured per node. The Secure+ feature continues to support SSL 3.0, TLS 1.0 and TLS 1.1.

Field Name	Field Definition	Valid Values
Node Name	Specifies the node record name.	.Remote This is not an editable field.

Field Name	Field Definition	Valid Values
Base Record	Specifies the name of the base record. If an alias record is selected, the base record name is displayed in this box.	Name of the local Connect:Direct node.
Type	Specifies the current record type.	Local for a local record and Remote for a remote record. This is not an editable field.
Disable Secure+	Disables Connect:Direct Secure Plus.	Default value is Disable Secure+. Note: If this option is selected, override is enabled, and no remote node definition exists for the remote node in the Connect:Direct Secure Plus parameters file, Connect:Direct Secure Plus is bypassed.
Enable SSL 3.0 Protocol	Enables SSL protocol to ensure that data is securely transmitted. The SSL3.0 is deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.	The default value is Disable Secure+.
Enable TLS 1.0 Protocol	Enables TLS protocol to ensure that data is securely transmitted. TLS1.0 is deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.	The default value is Disable Secure+.
Enable TLS 1.1 Protocol	Enables TLS protocol to ensure that data is securely transmitted. The TLS1.1 is deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.	The default value is Disable Secure+.
Enable TLS 1.2 Protocol	Enables TLS protocol to ensure that data is securely transmitted.	The default value is Disable Secure+.
Enable TLS 1.3 Protocol	Enables TLS protocol to ensure that data is securely transmitted.	The default value is Disable Secure+.
Disable	Disables the ability to override values in the .Remote node record with values in the remote node record.	The default value is Disable.
FIPS 140-2	Enables FIPS 140-2 security.	The default value is Disable.
SP800-131A Transition	Enables NIST SP800-131a security in transition mode.	The default value is Disable.
SP800-131A	Enables NIST SP800-131a security mode.	The default value is Disable.
Suite B 128 bit	Enables Suite B 128 bit security.	The default value is Disable.

Field Name	Field Definition	Valid Values
Suite B 192 bit	Enables Suite B 192 bit security.	The default value is Disable.
Node or Copy Statement Override	For PNODE, this parameter indicates whether process overrides, which may optionally be specified in Process, Submit, and Copy statements, will be allowed. For SNODE, this parameter indicates whether the Secure+ protocol specified by the PNODE will be allowed to override that specified by the SNODE.	The default value is No.
Authentication Timeout	Specifies maximum time, in seconds, that the system waits to receive the Connect:Direct Secure Plus blocks exchanged during the Connect:Direct Secure Plus authentication process. If you specify a value of 0, Connect:Direct waits indefinitely to receive the next message. Specify a time to prevent malicious entry from taking as much time as necessary to attack the authentication process.	A numeric value equal to or greater than 0, ranging from 0 to 3600. The default is 120 seconds.

3. Click the **TLS Options** tab. The **TLS Options** dialog box is displayed.
4. Select an existing Key Certificate from the key store. To select a Key Certificate from the keystore, click **Browse** next to **Key Certificate Label**. The **CMS KeyStore Certificate Viewer** appears.

   Note: You must add or import the key certificate into your key store prior to configuring your node. For additional information, see Import Existing Certificates or Create CMS Key Store in the documentation library. For additional information on how to use iKeyman, see http://www-01.ibm.com/support/knowledgecenter/SSYKE2_6.0.0/com.ibm.java.security.component.60.doc/security-component/ikeyman_overview.html?lang=en.
5. In the Key Certificates area, select the key certificate you want to use and click **OK** box.
6. Click the **External Authentication** tab. The **External Authentication** dialog box is displayed.
7. Choose one of the following options:
   - To enable external authentication on the remote node, click **Yes** in the **Enable External Authentication** box.
   - To disable external authentication on the remote node, click **No**.
8. Type the Certificate Validation Definition character string defined in External Authentication Server.
9. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Validating the Configuration

Perform this procedure to ensure that the nodes have been properly configured. The validation process checks each node to ensure that all necessary options have been defined and keys have been exchanged. Perform the following steps to validate the Secure+ parameters file:

1. From the **Secure+ Admin Tool Main Menu**, click **Validate Secure+** from the File menu. The **Secure+ Admin Tool - Validation Results** window is displayed.

2. If the Secure+ parameters file is not correctly configured, warning and error messages are displayed.
3. Go back to the Secure+ parameters file and make changes to correct each error reported.
4. Read each warning message. If necessary, change the Secure+ parameters file to correct each warning.

   Warning messages do not always mean that the Secure+ parameters file is configured incorrectly. Some warning messages are informational only.
5. Click **Close** to close the **Validation Results** window.

Configure External Authentication in the .SEAServer Record

At installation, a record named .SEAServer is created in the parameters file, which enables Connect:Direct Secure Plus to interface with External Authentication Server during TLS sessions to validate certificates. External Authentication Server properties are configured in this record and enabled/disabled in the local and remote node records.

Complete the following procedure to configure the server properties that will allow Connect:Direct for UNIX to interface with External Authentication Server:

Note: The values specified for this procedure must match the values specified in External Authentication Server.

1. Double-click the record called **.SEAServer**.
2. Type the Host Name for External Authentication Server.
3. Type the Port Number where External Authentication Server is listening. The default is 61366.
4. To enable caching SEAS certificate validation response, select **Enable Caching**.

   When enabled, Connect:Direct Secure Plus can reuse previously fetched certificate validity responses from External Authentication Server that is, cache the responses to ease the certificate validation process when Connect:Direct interfaces with External Authentication Server during a TLS sessions.
5. Type the **Cache Validity per certificate in hours**. Default is 24 hours. Range: 1-720 hours.
6. **Cache grace validity time per certificate when SEAS is unavailable in hours**

   Type the number of hours when the local cache entry of certificate expires and External Authentication Server is unavailable such that Connect:Direct Secure Plus can accept it from its cache. Default is 0 hours which means cache grace validity time does not apply. Range: 0-720 hours.

   Note: Cache grace validity time per certificate when SEAS is unavailable in hours should always be greater than or equal to **Cache Validity per certificate in hours**.
7. Click **OK** to update the record.

Configure Strong Password Encryption

This feature uses strong encryption to encrypt all Connect:Direct Process data stored on disk in the Connect:Direct work area while a Process is on the TCQ. This feature is enabled by default.

Disabling Strong Password Encryption

Complete the procedure below to disable Strong Password Encryption:

1. From the **Secure+ Admin Tool Main Menu** screen, select **Password Encryption** from the **Edit** menu. The **Secure+ Admin Tool - Password Encryption** window is displayed.
2. Click the **No** option for **Enable Strong Password Encryption**.
3. Click **OK** to disable Strong Password Encryption. The following message is displayed:

   ```
   The IBM Connect:Direct Server must be restarted for the changes to Strong Password Encryption to become effective.
   ```
4. Restart the IBM Connect:Direct Server.

Enabling Strong Password Encryption

Complete the procedure below to enable Strong Password Encryption:

1. From the **Secure+ Admin Tool Main Menu** screen, select **Password Encryption** from the **Edit** menu. The **Secure+ Admin Tool - Password Encryption** window is displayed.
2. Click the **Yes** option for **Enable Strong Password Encryption**.
3. Click **OK** to enable Strong Password Encryption. The following message is displayed:

   ```
   The IBM Connect:Direct Server must be restarted for the changes to Strong Password Encryption to become effective.
   ```
4. Restart the IBM Connect:Direct Server.

Resetting Passwords

If the Strong Password Encryption key stored in the .Password file is out of sync with the Strong Password Encryption key used to encrypt the passwords, you must reset all Strong Password Encryption passwords.

The .Password file can get out of sync if one of the following occurs:

- You restore the .Password file from a backup—The .Password file is updated each time the IBM Connect:Direct server is started, so the backup will probably not contain the current parameters.
- The .Password file is deleted—The .Password file is recreated as needed, so the Strong Password Encryption key used to encrypt the passwords no longer exists.
- The .password file is corrupt—The Strong Password Encryption Key used to encrypt the passwords is not accessible.

Complete the procedure below to reset the passwords:

1. Stop the IBM Connect:Direct server.
2. Delete the *<cdinstall>*/ndm/secure+/nodes/.Password file.
3. Start the IBM Connect:Direct server.
4. Manually delete all Processes in the TCQ. Refer to the *IBM Connect:Direct for UNIX User Guide* for command syntax and parameter descriptions for the delete Process and flush Process commands.

Decryption Failure

If the process KQV file fails decryption at startup or during runtime, the server places the Process in the HOLD/Error queue to raise the visibility of the error.

Configure Certificate Authentication for Client API Connections

The API connection certificate authentication feature allows clients to connect to a Connect:Direct server by using only an SSL Certificate with the Common Name (CN) specified as a user name.

If the intended client usage does not include submitting processes, then the user name does not have to be a real UNIX system user name and only needs to be defined in the Connect:Direct UNIX user authorization file. If process submission is intended, then the user specified in the CN must be a real UNIX system user. You can configure this feature in the user authorization information file of a Connect:Direct node. The API certificate authentication requires no user password to be presented.

This feature improves password management in large deployments of Connect:Direct, as it removes the extra administrative steps that result from password usage.

Note:

This feature is specific only to AIJ based API connections. When you use the authentication feature, ensure that the version of the AIJ is at least 1.1.00 Fix 000025. API connection certificate authentication is not supported by Windows SDK clients including Direct.exe CLI and Connect:Direct Requester.

Configuring API certificate authentication

Client Authentication must be enabled on the Connect:Direct Secure Plus. Client record. Client authentication is not enabled by default in Connect:Direct Secure Plus. During an API connection, a peer certificate is required from Control Center or the AIJ client. That certificate must contain a common name field of an SSL certificate whose contents match a Connect:Direct local user record in the Connect:Direct node. You also must use a blank password in order for Connect:Direct to trigger the API certificate authentication process.

Note: Although it is possible for a Connect:Direct Administrator to *create* a user name for an API program that does not submit processes, identity management is simplified by using a standard identity supported by an internal Certificate Authority. For example, if the API program runs on UNIX and the internal CA issues certificates for UNIX system users, the user name (and certificate Common Name) could be the UNIX system user name under which the API program runs. Or, if the internal CA issues certificates for systems, the user name (and certificate Common Name) could be the DNS name of the API program's host system.

A new functional authorities configuration parameter known as *client.cert_auth* is added to Connect:Direct for UNIX. The parameter specifies whether a specific user can log in as a client via API certificate authentication, and it must be set to Yes when you configure API certificate authentication.

Automate Setup with the Secure+ CLI

The Java-based Connect:Direct Command Line Interface (Secure+ CLI) and sample script enable you to create customized script that automate creating an initial installation of IBM Connect:Direct, populating the Secure+ parameters file, and managing node records. You can then distribute these scripts throughout your enterprise to implement the IBM Connect:Direct application. Before you create the scripts for distribution, consider creating an installation of Connect:Direct Secure Plus using the Secure+ Admin Tool and testing it to verify the results.

Start and Set Up the Secure+ CLI

The following sections describe the commands and parameters used to start and set up the command line environment.

Starting the Secure+ CLI

To start the Secure+ CLI:

1. Go to *d_dir*/ndm/bin.
2. Type the following command:

```
spcli.sh
```

3. Press **Enter**.

Control the Display of Commands

Set the following parameters to define how error messages are captured:

Parameter	Definition	Values
-li	Switch to enable the display of commands to the terminal.	y | <u>n</u>
-lo	Switch to enable the display of output and error messages to the terminal.	<u>y</u> | n
-le	Switch to enable the display of errors to STDERR.	y | <u>n</u>

Parameter	Definition	Values
-e	Switch to tell the Secure+ CLI to exit when the return code is higher than the specified number. If you do not include this parameter, Secure+ CLI continues to run even after an error has occurred.	0 | 4 | 8 | 16
-p	The full path of the default Secure+ parameters file directory. The Secure+ parameters file in this directory is opened automatically.	
-h	Switch to display the usage of the Secure+ CLl.	

Control Help

The Help command determines what help information is displayed. You can list all Secure+ CLI commands and display help for individual commands.

Command	Description
help	Displays all the Secure+ CLI commands.
help <command>	Displays help for the specified command.

Specify Delimiter Characters

Define the following commands to determine how error messages are captured:

Command	Definition	Values
Set begdelim= enddelim=	Defines beginning and ending character to use to enclose keywords that use blanks and other special characters.	Any character The default value is **“** (double quotes).

Encrypt Passwords to Use with the Secure+ CLI

The Secure+ CLI displays passwords in plain text. If your company security policy mandates that you use encrypted passwords, you can use the Local Connection Utility (LCU) to create an LCU file that contains non-encrypted information used to encrypt the password and the encrypted password. For more information on creating and using LCU files, see Encrypt Passwords for use with CLI.

Sample Script

The following script is provided as a model for creating custom scripts to define your IBM Connect:Direct environment and automate the implementation of it. To prevent any loss of data, you cannot run the script, but you can save it with a different name and modify it to suit your needs.

The sample script is available in Automation Script. The script is designed to assist you as follows:

spcust_sample1.sh

An example of configuring IBM Connect:Direct to use the TLS protocol with the Secure+ CLI. The example demonstrates the configuration of Connect:Direct with the trusted root file, key certificates, and ciphers.

Maintain the Secure+ Parameters File

The commands in the following table describe how to maintain the Secure+ parameters file from the command line interface.

Command	Description	Parameter	Values
Init Parmfile	Creates the Secure+ parameters file. Must be initialized before you can define nodes.	localnode=Name of the local node where the Secure+ parameters file will be created.	local node name
		path=Location where the Secure+ parameters file will be created.	directory location For example, *d_dir*/ndm/secure+/node
		passphrase=Arbitrary set of characters that encrypts the Secure+ parameters file.	a string at least 32 characters long
Open Parmfile	Opens a Secure+ parameters file so that you can configure it.	path=Location where the Secure+ parameters file will be created.	directory location For example, *d_dir*/ndm/secure+/node
Close Parmfile	Closes the Secure+ parameters file. After this command is issued, no more updates can be performed on the Secure+ parameters file.	None	None
Refresh Parmfile	Refreshes the Secure+ parameters file. This will close the current parameters file and reopen it, bringing in any changes since last opened.	None	None
Validate Parmfile	Validates the Secure+ parameters file and ensures that it is a valid file.	None	None
Rekey Parmfile	Recreates the Secure+ parameters file if it becomes corrupted.	passphrase=Arbitrary set of characters that encrypts the Secure+ parameters file.	passphrase, up to 32 characters long

Command	Description	Parameter	Values
Sync Netmap	Imports remote node records defined in the IBM Connect:Direct network map.	path=Location and name of the network map file.	location of network map file
		name=Name of the node in the network map. Use wildcard characters to resync more than one node at a time.	node name or wildcard Wildcard values are: Asterisk (*)—any number of characters. Example: kps.* syncs up all nodes with a name that starts with kps. Question mark (?)—a single character. Example: k?s.* syncs up kas.* and kbs.*.

Display Information

The following commands are available to display information:

Command	Description	Parameter
display info	Displays information about when the Secure+ parameters file was last updated.	None
display all	Displays all nodes in the Secure+ parameters file.	None
display localnode	Displays the values defined in the .Local node record.	None
display remotenode	Displays the values defined in remote node records.	node name or wildcard name—The name of the node to display information about. Use wildcard characters to display information about a group of remote node records. The options are: Asterisk (*)—Indicates any number of characters. For example, kps.* displays all nodes with a name that starts with kps. Question mark (?)—Indicates a single character. For example: k?s.* displays kas.* and kbs.*.
display client	Displays the values defined in the .Client node record.	None
display seaserver	Displays the values defined in the .SEAServer record.	None

Command	Description	Parameter
display protocols	Displays supported security protocols which should be defined in a comma separated list . Supported protocols are: TLS1.2,TLS1.3 TLS1.0, TLS1.1, and SSL3.0 are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.	None
display securitymodes	Displays supported security modes for additional security. These modes are: FIPS140-2 | SP800-131A_TRANSITION | SP800-131A_STRICT | SUITE_B-128 | SUITE_B-192	None
display ciphersuites	Displays all supported Cipher Suites for Secure+ which can be defined either as a single cipher suite or in a comma separated list.	None

Manage CMS Keystore

The commands in the following table describe how to create and maintain the CMS keystore file from the command line interface.

Command	Description	Parameter	Values
create keystore	Will create a new CMS Key Store file.	File=While a default keystore file is created at installation and can be used, you may need to create a new CMS KeyStore File.	<path to CMS KeyStore file (*.kdb)> Default path is in: d_dir/ndm/secure+/certificates/cdkeystore.kdb
		Passphrase=The password for the new KeyStore file.	A string with a minimum of three characters and a maximum of eighty characters. *This password must be retained; it will be required to administer the Secure+ KeyStore.
		PopulateRoots=Populate with standard certificate authorities. This will import all standard public CA Root certificates into the new KeyStore file.	y | <u>n</u>
update keystore	Updates the password used to access the CMS KeyStore	File=Path to existing CMS KeyStore and filename.	<path to CMS KeyStore file (*.kdb)> Default path is in: d_dir/ndm/secure+/certificates/cdkeystore.kdb

Command	Description	Parameter	Values
		Passphrase=The password for the KeyStore file.	The retained password of the KeyStore.
import keycert	Imports existing keycerts into the keystore file.	File=Existing key certificate file. *This file contains the private key*	Full path and filename to key certificate file to be imported.
		Passphrase=Password of key certificate file to be imported.	Pre-defined password of key certificate file.
		Label=(optional) Name of imported key certificate file.	A string of characters which can be an alias name but if it is not defined, the Common Name of the certificate will be the label used.
		SyncNodes=Update node/certificate references	y | <u>n</u>
		ImportMode=Type of import to be used.	Add | Replace | <u>AddOrReplace</u>
import trustedcert	Imports public certificate files from trading partners.	File=Trusted public file from trading partner.	Full path and filename to trusted certificate file to be imported.
		ImportMode=Type of import to be used.	Add | Replace | <u>AddOrReplace</u>
delete keystoreentry	Deletes certificates from CMS keystore.	File=Can be either key certificate file or trusted public trading partner file.	Full path and filename to certificate file.
		Label=Specified label of imported certificate file.	Label which was defined at time of import of the certificate file.
		DeleteChain=Defines whether to delete the entire chain, if it exists.	y | <u>n</u>
		SyncNodes=Reset node/certificate references	y | <u>n</u>

Update the .Local Node Record

The **update localnode** command configures the protocol for the .Local node record. The command has the following parameters:

Command	Parameter	Values
update localnode	protocol=Specifies a comma delimited list of Protocols to use in the .Local node record.	Disable | TLS 1.2,TLS 1.3 (See Display Protocols) TLS1.0, TLS1.1, SSL3.0 are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2.
	SecurityMode	Disable | FIPS140-2 | SP800-131A_TRANSITION | SP800-131A_STRICT | SUITE_B-128 | SUITE_B-192 (See Display SecurityModes)
	override=Identifies if values in the remote node can override values defined in the .Local node record.	y | n
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the IBM Connect:Direct control blocks exchanged during the IBM Connect:Direct authentication process.	0–3600 The default is **120** seconds.
	KeyCertLabel=Identifies the label of the key certificate.	keycert label | null **Note:** If no keycert label is specified, the following should be noted: Pnode sessions will fail if the remote node requires client authentication. Snode sessions will fail.
	EncryptData=If no is specified, Encrypt Only Control Block Information; data is sent unencrypted. Default is Yes - data and control block information are encrypted.	y | n
	ClientAuth = Enables client authentication in a .Client node record.	y | n
	CipherSuites= Specifies the cipher suites enabled. **Note:** Only certain cipher suites are supported in FIPS-mode.	comma delimited list of cipher suites | all | null all—Enables all ciphers. null—Clears any existing values from the node definition.
	SeaEnable=Enables certificate validation by Sterling External Authentication Server	y | n

Command	Parameter	Values
	SeaCertValDef=Character string defined in Sterling External Authentication Server (SEAS).	character string | null null—Clears any existing values from the node definition.

Manage Remote Node Records

This section contains the commands and parameters used to create, update, display, and delete remote node records.

Create a Remote Node Record

The **create remotenode** command creates a remote node record and configures the protocol settings. The command has the following parameters:

Command	Parameter	Values
create remotenode	model=Name of an existing node to use as a model to copy from.	name of a valid remote node
	Name=Identifies name of the remote node record.	name
	protocol=Specifies a comma delimited list of Protocols to use in the remote node record.	Disable | TLS 1.2,TLS 1.3 |<u>DefaultToLN</u> TLS1.0, TLS1.1, and SSL3.0 are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2. (See Display Protocols)
	SecurityMode	Disable | FIPS140-2 | SP800-131A_TRANSITION | SP800-131A_STRICT | SUITE_B-128 | SUITE_B-192 | <u>DefaultToLN</u> (See Display SecurityModes)
	override=Identifies if values in the copy statement can override values defined in the remote node record.	y | n | <u>DefaultToLN</u>
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the IBM Connect:Direct control blocks exchanged during then Connect:Direct authentication process.	0–3600 The default is **120** seconds.
	KeyCertLabel=Identifies the label of the key certificate.	keycert label | null
	EncryptData=If no is specified, Encrypt Only Control Block Information; data is sent unencrypted. Default is Yes - data and control block information are encrypted.	<u>y</u> | n | <u>DefaulttoLN</u>

Command	Parameter	Values
	ClientAuth = Enables client authentication with a remote trading partner.	y | n | <u>DefaultToLN</u>
	CertCommonName=The certificate common name defined in the certificate.	name | null null—Clears any existing values from the node definition.
	CipherSuites= Specifies the cipher suites enabled.	comma delimited list of cipher suites | All | null
	SeaCertValDef=Character string defined in Sterling External Authentication Server (SEAS).	character string | null null—Clears any existing values from the node definition.

Update the Remote Node Record

The **update remotenode** command creates a remote node record and configures the protocol settings. The command has the following parameters:

Command	Parameter	Values
update remotenode	Name=Specifies name for the remote node record.	remote node name | wildcard Use wildcard characters to update a group of remote node records. The options are: Asterisk (*)—Any number of characters. Example: kps.* displays remote nodes with a name that starts with kps. Question mark (?)—Single character. Example: k?s.* displays kas.* and kbs.*.
	protocol=Specifies a comma delimited list of Protocols to use in the remote node record.	Disable |TLS 1.2,TLS 1.3 | <u>DefaultToLN</u> TLS1.0, TLS1.1, SSL3.0 are deprecated and should not be used. It is recommended that trading partners using depcrecated protocols migrate to TLS 1.3 or TLS 1.2. (See Display Protocols)
	SecurityMode	Disable | FIPS140-2 | SP800-131A_TRANSITION | SP800-131A_STRICT | SUITE_B-128 | SUITE_B-192 | <u>DefaultToLN</u>
	override=Identifies if values in the copy statement can override values defined in the remote node record.	y | n | <u>DefaultToLN</u>
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Connect:Direct authentication process.	0–3600 The default is **120** seconds.

Command	Parameter	Values
	KeyCertLabel=Identifies the label of the key certificate.	keycert label | null
	EncryptData=If no is specified, Encrypt Only Control Block Information; data is sent unencrypted. Default is Yes - data and control block information are encrypted.	y | n | DefaulttoLN
	ClientAuth = Enables client authentication with a remote trading partner.	y | n | DefaultToLN
	CertCommonName=The certificate common name defined in the certificate.	name | null null—Clears any existing values from the node definition.
	CipherSuites= Specifies the cipher suites enabled. **Note:** Only certain cipher suites are supported in FIPS-mode.	comma delimited list of cipher suites | All | null
	SeaEnable=Enables certificate validation by Sterling External Authentication Server.	y | n | DefaultToLN DefaultToLN—Defaults to the setting specified in the .Local node record
	SeaCertValDef=Character string defined in Sterling External Authentication Server (SEAS).	character string | null null—Clears any existing values from the node definition.

Display a Remote Node Record

The **display remotenode** command displays information about one or more remote node records. The command has the following parameter:

Command	Parameter	Values
display remotenode	name=Name of the remote node record to display information about.	node name | wildcard value To display information about more than one remote node record, use wildcard characters. Use wildcard characters to display information about a group of remote node records. The options are: Asterisk (*)—Any number of characters. Example: kps.* displays remote nodes with a name that starts with kps. Question mark (?)—A single character. Example: k?s.* displays kas.* and kbs.*.

Manage Remote Node Records

Create Alias

The **create alias** command will create an alias record for an existing node record in the Secure+ parmfile. The command has the following parameter:

Command	Parameter	Value
create alias	name=The alias name to be used.	An alias name for an existing node name record. **Important:** Characters used in Netmap Node Names (or Secure+ Node Names or Secure+ Alias Names) should be restricted to A-Z, a-z, 0-9 and @ # $. _ - to ensure that the entries can be properly managed by Control Center, Sterling Connect:Direct Browser User Interface, or IBM Sterling Connect:Direct Application Interface for Java for Java (AIJ) programs.
	basename=The name of the existing node record.	The existing node name

Delete a Remote Node Record

The **delete remotenode** command deletes one or more remote node records. The command has the following parameter:

Command	Parameter	Values
delete remotenode	name=Name of the remote node record to display information about. Use wildcard characters to delete a group of remote node records.	remote node name | wildcard value To display information about more than one remote node record, use wildcard characters. Use wildcard characters to display information about a group of remote node records. The options are: Asterisk (*)—Any number of characters. Example: kps.* displays remote nodes with a name that starts with kps. Question mark (?)—A single character. Example: k?s.* displays kas.* and kbs.*.

Update the .Client Node Record

The **update client** command creates a .Client node record and configures the protocol settings. The command has the following parameters:

Command	Parameter	Values
update client	protocol=Specifies a comma delimited list of Protocols to use in the .Client record	Disable | TLS 1.2,TLS 1.3 | <u>DefaultToLN</u> TLS1.0, TLS1.1, and SSL3.0 are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2. (See Display Protocols)

Command	Parameter	Values
	SecurityMode	Disable | FIPS140-2 | SP800-131A_TRANSITION | SP800-131A_STRICT | SUITE_B-128 | SUITE_B-192 | DefaultToLN (See Display SecurityModes)
	override=Indicates whether a client will be allowed to override the specified Secure+ protocol. For example, *y* will allow an unsecure client to connect when a secure protocol is configured.	y | n
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Connect:Direct authentication process.	0–3600 The default is **120** seconds.
	KeyCertLabel=Identifies the label of the key certificate	keycert label | null
	EncryptData=If no is specified, Encrypt Only Control Block Information; data is sent unencrypted. Default is Yes - data and control block information are encrypted.	y | n | DefaulttoLN
	CipherSuites= Specifies the cipher suites enabled.	comma delimited list of cipher suites | All | null

Maintain the Sterling External Authentication Server Record

This section contains the commands and parameters used to update and display the **.SEAServer** record.

Update the Sterling External Authentication Server Record

The **update seaserver** command configures properties for Sterling External Authentication Server (SEAS) in the .SEAServer record that is created at installation. The command has the following parameters:

Command	Parameter	Values
update seaserver	Protocol=Specifies a comma delimited list of Protocols to use in the .SEAServer record.	Disable | TLS1.2,TLS 1.3 DefaultToLN TLS1.0, TLS1.1, and SSL3.0 are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS 1.2. (See Display Protocols)
	SeaHost=External authentication host name defined in SEAS.	host name | null null—Clears any existing values from the node definition

Command	Parameter	Values
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Connect:Direct authentication process.	0–3600 The default is **120** seconds.
	SeaPort=External authentication server port number (listening) defined in SEAS.	port number | 61366
	SeaCacheEnable=Enable caching External Authentication Server certificate validation response.	Y|N The default is **N**.
	SeaCacheValidityTime=Time duration during which the local cache entry is valid for certificates	The default is **24** hours. Range: 1 to 720 hours
	SeaGraceValidityTime=Number of hours when the local cache entry of certificate expires and External Authentication Server is unavailable such that Connect:Direct Secure Plus can accept it from its cache.	The default is **0** hours which means cache grace validity time does not apply. Range: 0 to 720 hours

Display the Sterling External Authentication Record

The **display SEAServer** command displays information about the **.SEAServer** record. This command has no parameters.

Strong Password Encryption

This section contains the commands and parameters used to update and display the **.Password** file.

Update the .Password File

The **update password** command enables or disables Strong Password Encryption. The update goes into effect after you start the IBM Connect:Direct Server. The command has one parameter, **SpeEnable**, which can be set to **Y** or **N** to enable or disable Strong Password Encryption. Following is an example:

```
Update Password
SpeEnable=<Y>
;
```

If you enable or disable Strong Password Encryption, the server displays the following warning:

```
SPCG741W=The IBM Connect:Direct Server must be restarted for the changes to Strong Password
Encryption to become effective.
```

Display the .Password File

The **Display Password** command displays the Strong Password Encryption setting and .Password history.

Displaying the IBM Connect:Direct Node Information

After you set up node records in Connect:Direct Secure Plus, you can view all of the nodes and their attributes from the **Secure+ Admin Tool Main Menu Window**. To display a Connect:Direct Secure Plus node record, open it by double-clicking the node record name.

Node List Field Descriptions

Below is a description of all the fields displayed in the **Node Name** List:

Field Name	**Field Definition**	**Values**
Node Name	Displays the node record name.	.Local | remote node name | .client
Type	Displays the current record type.	L | R L—Local record R—Remote record
Connect:Direct Secure Plus	Displays the status of IBM Connect:Direct.	N | TLS | SSL | * N—Disabled TLS—TLS protocol SSL—SSL is deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS1.2. *—Default to local node
Override	Displays the status of override. Enable override in the local node to allow remote node records to override the settings in the local node record.	Y | N | * Y—Enabled N—Disabled *—Default to local node
CipherSuites	Displays the TLS or SSL cipher suites that are enabled for the node record.	Varies, based on the cipher suites enabled.
ClientAuth	Displays the status of client authentication. If the TLS protocol is used, enabling client authentication means the SNODE verifies the identity of the PNODE.	Y | N | * Y—enabled N—Disabled *—Default to local node
LimExpr	Identifies if the Limited Export version is being used by a remote node.	Y | N | * Y—Enabled N—Disabled *—Default to local node

Field Name	Field Definition	Values
AutoUpdate	Indicates if the option to automatically update key values during communication is enabled.	Y | N | * Y—enabled N—disable *—default to local node
Base Record	Displays the name of the base record for the alias records.	There are no parameter values.

Viewing Node Record Change History

Perform the following steps to view the history of changes to a Connect:Direct Secure Plus node record.

1. From the **Secure+ Admin Tool Main Window**, double-click the node record name.
2. Click the **Security Options** tab.

   The history of changes is displayed in the **Update History** field.

Viewing Information about the Secure+ Parameters File

Perform the following steps to view information about the Secure+ parameters file:

1. Open the **Secure+ Admin Tool**.
2. On the **File** menu option of the **Secure+ Admin Tool Main Window,** click Info. The **File Information** dialog box is displayed.

   Refer to the following table for an explanation of the fields.

Field Name	Field Definition
Current File	Displays the name of the Secure+ parameters file opened.
Number of Records	Lists the number of nodes defined in the Secure+ parameters file.
Number of Updates	Displays how many times the Secure+ parameters file has been updated.
Last 3 Updates	Displays the name of the last three nodes updated.

3. Click **OK** to close the **File Information** dialog box.

Modify a Connect:Direct Secure Plus Configuration

After using Connect:Direct Secure Plus, it may be necessary to modify a configuration. This section provides the following procedures for modifying Connect:Direct Secure Plus information:

- Disabling Connect:Direct Secure Plus
- Deleting a Connect:Direct Secure Plus remote node record
- Resecuring the Secure+ parameters file and Secure+ access file
- Changing the cipher suites

Disabling Connect:Direct Secure Plus

You can use this procedure to disable all nodes in a configuration or one remote node. Perform the following steps to disable Connect:Direct Secure Plus:

1. Do one of the following:
   - To disable all nodes in a configuration, open the local node record.
   - To disable one node, open the remote node record for that node.
2. Click the **Security Options** tab.
3. Click the **Disable Secure+**.
4. Click **OK** to update the node record.

   Note: In order to continue IBM Connect:Direct operations with IBM Connect:Direct disabled, both trading partners must disable Connect:Direct Secure Plus.

Deleting a Remote Node Record

If a remote node record is no longer defined in the network map, you can remove it from the Secure+ parameters file. The following procedure deletes nodes that are defined in the IBM Connect:Direct parameters file but not in the selected network map:

1. From the **Secure+ Admin Tool Main Menu Window**, click the **Sync with Netmap** of the **File** menu.
2. Click the network map to use from the pull down list.
3. Click **OK**.
4. Click **Skip** to move through the **Select Netmap Entries** to the **Add** dialog box.
5. Do one of the following to delete node records:
   - To delete selected node records, highlight the remote nodes to delete and click **Delete Selection**.
   - To delete all remote node records that are not found in the network map, click **Delete All**.

   Note: Do not delete the remote node record that is named for the IBM Connect:Direct node. It is the base record for the **.Local** node record. You cannot delete the **.Local** node record.

Resecuring the Secure+ Parameters File and Secure+ Access file

Routinely, or if your Secure+ access file is compromised, perform the following steps to resecure Connect:Direct Secure Plus:

1. From the **Secure+ Admin Tool Main Window**, click **Rekey Secure+** from the **File** menu. The **Rekey Secure+** dialog box is displayed.
2. Type an alphanumeric string at least 32 characters long in the **Passphrase** field. Connect:Direct Secure Plus uses the passphrase to re-encrypt the Secure+ parameters file the and Secure+ access files. You do not have to remember this passphrase value.
3. Click **OK** to accept the new passphrase. The Connect:Direct Secure Plus decrypts and re-encrypts the Secure+ parameters file and Secure+ access file.

   CAUTION: Do not type a new passphrase if an error occurs. If an error occurs while you are resecuring the files, restore the node records from the ACFSave directory. This directory is created after the **Rekey Secure+** feature is executed.

Connect:Direct Secure Plus Statistics Record Information

IBM Connect:Direct logs statistics for IBM Connect:Direct Process activity. IBM Connect:Direct statistics includeConnect:Direct Secure Plus information for a Process.

Fields are included in Connect:Direct Process statistics records to provide Connect:Direct Secure Plus information about the Process. Connect:Direct Secure Plus information is included in the Process statistics information only when you attach to aConnect:Direct Secure Plus server. For information on viewing statistics, refer to the Sterling*Sterling™ Connect:Direct Browser User Interface User Guide*. When you use the **select statistics** function to view information about a IBM Connect:Direct Process, statistics information about a particular Process is displayed. If Connect:Direct Secure Plus is enabled,Connect:Direct Secure Plus fields are also displayed.

The Connect:Direct Secure Plus fields and values available using the **select statistics** function are shown in the following table:

Field Name	**Field Description**	**Values**
Connect:Direct Secure Plus Enabled	Specifies whether Connect:Direct Secure Plus is enabled.	Y | N
Connect:Direct Secure Plus Protocol	Specifies which protocol is enabled.	TLS 1.2 | TLS 1.3 SSL 3.0, TLS 1.0, and TLS 1.1 protocols are deprecated and should not be used. It is recommended that trading partners using deprecated protocols migrate to TLS 1.3 or TLS1.2.
Cipher Suite	Displays the cipher suite used during a session.	cipher suite name For example: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
PNode Cipher List	Specifies the encryption algorithms available for the PNODE during the session.	IDEACBC128 | TDESCBC112 | DESCBC56
PNode Cipher	Specifies the preferred data encryption as specified in the Secure+ parameters file of the PNODE.	Y | N | algorithm name
SNode Cipher List	Specifies the encryption algorithms available for the SNODE during the session as specified in the Secure+ parameters file of the SNODE.	IDEACBC128 | TDESCBC112 | DESCBC56
SNode Cipher	Specifies the preferred data encryption algorithm as defined in the Secure+ parameters file of the SNODE.	Y | N | algorithm name
Control Block Cipher	Specifies the algorithm used for encrypting control blocks. This value is determined during authentication when the PNODE and SNODE are merged.	IDEACBC128 | TDESCBC112 | DESCBC56
Copy Data Cipher	Specifies the encryption method used for encrypting data. The value is determined after the values in the SNODE and the PNODE are merged.	IDEACBC128 | TDESCBC112 | DESCBC56

Field Name	Field Description	Values
PNODE Signature Enabled	Indicates whether digital signatures are enabled for the PNODE. This value is obtained from the Secure+ parameters file settings. If the COPY statement overrides the Secure+ parameters file, the value from the COPY statement is used.	Y | N
SNODE Signature Enabled	Indicates whether digital signatures are enabled for the SNODE. This value is obtained from the Secure+ parameters file settings.	Y | N
Signature Enabled	Identifies the digital signature value used for a copy operation. In the Session Start record, this value is the result of the merged value between the PNODE and SNODE. In the Copy Termination record, if the COPY statement overrides the Secure+ parameters file value, the merged value depends on the value supplied in the COPY statement. (The unprocessed value from the COPY statement is recorded in the Signature Enabled field of the PNODE).	Y | N
Current Signature Verified	Indicates whether the current digital signature was verified.	Y | N
Previous Signature Verified	Indicates whether the previous digital signature was verified.	Y | N

IBM Connect:Direct CLI Select Statistics Detail

When you use the IBM Connect:Direct CLI **select statistics** function to view the information about a Connect:Direct Process, you see statistics information about a particular Process. Connect:Direct Secure Plus fields are shown in bold in the following samples. The Connect:Direct field names, descriptions, and valid values are shown in Connect:Direct Secure Plus Statistics Record Information. For more information on Connect:Direct certificate auditing, see Secure+ Parameters File Auditing.

Session Start (SSTR) Record

The following sample Session Start Record (SSTR) displays the output of an SSL session:

```
Record Id         => SSTR
Process Name      =>                      Stat Log Time  => 15:23:21
Process Number    => 0                    Stat Log Date => 10/16/2004
Submitter Id      =>

Start Time        => 15:23:20             Start Date     => 10/16/2004
Stop Time         => 15:23:21             Stop Date      => 10/16/2004

SNODE             => JKTIB8100
Completion Code   => 0
Message Id        => LSMI004I
Message Text      => PNODE session started - remote node &NODE
Secure+ Protocol => SSL 3.0
SSL Cipher Suites => ssl_RSA_WITH_RC4_128_MD5
---------------------------------------------------------------------
```

Copy Termination (CTRC) Record

The Copy Termination Record (CTRC) sample below uses the SSL protocol:

```
Record Id          => CTRC
Process Name       => XX              Stat Log Time  => 15:26:32
Process Number     => 195             Stat Log Date => 10/16/2004
Submitter Id       => user1
Start Time         => 15:23:47        Start Date     => 10/16/2004
Stop Time          => 15:26:32        Stop Date      => 10/16/2004
SNODE              => DLAS8100
Completion Code    => 0
Message Id         => SCPA000I
Message Text       => Copy operation successful.
COPY DETAILS: Ckpt=> Y Lkfl=> N Rstr=> N XLat=> N Scmp=> N Ecmp=> N
From node          => S
Src File           => D:\long path
Dest File          => D:\long path
Src CCode          => 0              Dest CCode       => 0
Src Msgid          => SCPA000I       Dest Msgid       => SCPA000I
Bytes Read         => 23592960       Bytes Written    => 23592960
Records Read       => 1024           Records Written  => 1024
Bytes Sent         => 23791420       Bytes Received   => 23791420
RUs Sent           => 30721          RUs Received     => 30721
Secure+ Protocol =>SSL 3.0
SSL Cipher Suites =>SSL_RSA_WITH_RC4_128_MD5
---------------------------------------------------------------------
```

Secure+ Parameters File Auditing

IBM Connect:Direct provides auditing of Secure+ parameters files and certificates for archival purposes.

The Connect:Direct Secure Plus Administration Tool (Secure+ Admin Tool) and the Connect:Direct Secure Plus Command Line Interface (Secure+ CLI) log changes made to the Connect:Direct Secure Plus parameters file (Secure+ parameters file). The following events are logged:

- Application Startup
- Init Parmfile
- Open Parmfile
- Sync Netmap
- Rekey Parmfile
- Create Node
- Update Node
- Delete Node

The Secure+ parameters file logging feature has the following operational characteristics:

- The logging feature is always enabled and cannot be disabled.
- If errors occur when the log is being updated, the application terminates.
- Each log entry contains a timestamp, user ID, and a description of the action/event.
- When an existing node is updated, any changed fields are reported.
- When a node is created or deleted, the values of all non-empty fields are reported.
- Any commands that modify a node are logged.

Note: The certificates used by Connect:Direct Secure Plus are individual files that can be stored anywhere on the system. As a result, the logging feature cannot detect when existing certificate files are modified. Connect:Direct Secure Plus only stores the certificate path name and detects changes to this field only.

Access Secure+ Parameters File Audit Logs

The Secure+ parameters file audit logs are stored in a dedicated directory, ...\secure+\log. The log file naming convention is SP[YYYY][MM][DD].001 (using local time), and the contents of a log file are limited to a single calendar date. You can view these log files using any text editor. Log files are not deleted by IBM Connect:Direct.

Secure+ Parameters File Audit Log Entries

Each audit log has the following header:

```
[YYYYMMDD][HH:MM:SS:mmm][userid]
```

When a parameter file is created or opened, an ID is generated that associates the change with the node being updated as shown in the following:

```
[YYYYMMDD][HH:MM:SS:mmm][userid][ParmFileID]
```

The following fields may appear in a **create**, **update**, or **delete** audit record.

Field Name	Description
Name	Name of the node
BaseRecord	Name of the base record
Type	Record type of local, remote, or alias

Field Name	Description
Protocol	Enables Connect:Direct Secure Plus protocol
Override	Enables overriding the current node
AuthTimeOut	Authentication timeout
SslTlsTrustedRootCertFile	Pathname to trusted roots file
SslTlsCertFile	Pathname to key certificate file
SslTlsCertPassphrase	Key certificate passphrase (masked)
SslTlsEnableClientAuth	Enable client authentication
SslTlsCertCommonName	Common name of the remote certificate to verify
SslTlsEnableCipher	List of SSL/TLS cipher suites
SslTlsSeaEnable	Enable external authentication
SslTlsSeaCacheEnable	Enable caching External Authentication Server certificate validation response.
SeaCacheValidityTime	Time duration during which the local cache entry is valid for certificates
SeaGraceValidityTime	Number of hours when the local cache entry of certificate expires and External Authentication Server is unavailable such that Connect:Direct Secure Plus can accept it from its cache.
SeaCertValDef	External authentication validation definition
SeaHost	External authentication host name
SeaPort	External authentication port number

Secure+ Parameters File Audit Log Error Reporting

Errors are reported for the following logging functions: **open log**, **write log**, and **lock log**. If an error occurs during one of these functions, an error message is displayed, and the application is terminated. The lock function times out after 30 seconds. Typically, Secure+ Admin Tool or the Secure+ CLI hold the lock for less than one second per update.

IBM Connect:Direct Secure Plus Certificate Auditing

In a TLS session, audit information about the identity certificate and its signing certificate is logged in the statistics log in the Session Start (SSTR) and Copy Termination (CTRC) records. The audit information is included in the response data from a **select statistics** command in the SSTR and CTRC records. In a TLS session, the PNODE (client) always logs the audit information. The SNODE (server) only logs the information when client authentication is enabled. For logging to occur, the session handshake must succeed and progress to the point of logging the SSTR and CTRC records.

Certificate Audit Log Entries

The audit consists of the subject name and serial number of the identity and its signing certificate. The identity certificate also contains an issuer attribute, which is identical to the signing certificate subject name. Although many signing certificates may exist between the identity and final root certificate, the audit includes only the last two certificates in a chain: an intermediate certificate and an end certificate.

In the SSTR and CTRC records, the CERT contains the common name and serial number of the key certificate, and the CERI contains the common name of the issuer and the serial number of an intermediate or root CA. They may also contain the certificate serial number, for example:

```
CERT=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/CN=Test ID/SN=99c0ce01382e6c83)|
CERI=(/C=US/ST=MA/L=Marshfield/O=test.org/CN=root CA/SN=da870666bbfb5538)
```

Connect:Direct Secure Plus certificate audits may contain the following fields:

Field Name	Abbreviation	Max Lengths (RFC 2459)
Common Name	CN	64
Country	C	2
Locality	L	128
State	ST	128
Organization	O	64
Organization Unit	OU	64
Email Address	emailAddress	128
Serial Number	SN	128 (estimated)

Access Certificate Audit Logs

Certificate audit information located in the SSTR and CTRC records cannot be accessed directly using Connect:Direct Requester or Sterling Connect:Direct Browser User Interface. To access certificate information, you can issue a query directly to the database or use an SDK-based or JAI-based program to issue a **Select Statistics** command. The response to the **Select Statistics** command contains the **AuditInfo** field of the statistics records, including the SSTR and CTRC records. This field contains certificate audit information.

The following example was generated using a database query. The certificate audit information is highlighted in bold.

```
'2007-05-21 14:50:27', 2, 'SSTR', 'CAEV', '', 0, '2007-05-21 14:50:26', '2007-05-21 14:50:27', '', '', 'JLYON-
XP.4400', 0, 'MSGI=LSMI004I|SBST=(&NODE=JLYON-XP.4400)|PNOD=JLYON-XP.4400|CSPE=Y|CSPP=TLSv1|
CSPS=TLS_RSA_WITH_AES_256_CBC_SHA|

CERT=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/
CN=Example Test ID/SN=a9febbeb4f59d446)|
CERI=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/CN=Example
IntermediateCA/SN=a69634a8a7830268)|STSD=2|TZDI=-14400|'

'2007-05-21 14:50:28', 2, 'CTRC', 'CAPR', 'SAMPLE', 1, '2007-05-21 14:50:27', '2007-05-21
14:50:28', 'JLYON-XP.4400', 'jlyon', 'JLYON-XP.4400', 0, 'MSGI=SCPA000I|LCCD=0|LMSG=SCPA000I|OCCD=0|
OMSG=SCPA000I|PNAM=SAMPLE|PNUM=1|
SNAM=STEP1|SBND=JLYON-XP.4400|SBID=jlyon|PNOD=JLYON-XP.4400|SNOD=JLYON-XP.4400|LNOD=P|
FROM=P|XLAT=N|ECZI=N|ECMP=N|SCMP=N|OERR=N|CKPT=Y|LKFL=N|RSTR=N|
RUSZ=65535|PACC=|SACC=|PPMN=|SFIL=C:\Program Files\Sterling Commerce\Connect Direct
v4.4.00\Server\Process\Sample.html|SDS1= |SDS2= |SDS3= |SFSZ=0|SBYR=861|SRCR=1|SBYX=863|
SRUX=1|SNVL=-1|SVOL=|DFIL=C:\Program Files\Sterling Commerce\Connect Direct
v4.4.00\Server\Process\Verify.html|PPMN=|DDS1=R|DDS2= |DDS3= |DBYW=861|DRCW=1|DBYX=863|
DRUX=1|DNVL=0|DVOL=|CSPE=Y|CSPP=TLSv1|CSPS=
TLS_RSA_WITH_AES_256_CBC_SHA|CERT=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/
CN=Example Test ID/SN=a9febbeb4f59d446)|CERI=(/C=US/ST=MA/L=Marshfield/O=test.org/
OU=Dev/CN=Example Intermediate CA/SN=a69634a8a7830268)

|PCRC=N|ETMC=60|ETMK=10|ETMU=0|STSD=2|TZDI=-14400|'
```

Certificate Audit Log Error Reporting

If an error occurs when the subject name is extracted from the identity (CERT) or issuer's (CERI) certificates, the following message ID is logged:

```
CERT=(MSGI=CSPA310E)|CERI=(MSGI=CSPA310E)
```

Only the message ID is displayed with the CERT or CERI tokens; the standard IBM Connect:Direct error function is not used. After the error occurs, the session continues.

Connect:Direct Secure Plus Troubleshooting

Use the following table to help troubleshoot problems with Connect:Direct Secure Plus:

Problem	Possible Cause	Solution
Connect:Direct Secure Plus features are enabled in the Secure+ parameters file, but the statistics record indicates that these functions are disabled.	The Connect:Direct network maps do not contain entries for the PNODE and SNODE.	Verify that the network map entries for both the PNODE and the SNODE exist.
Running a Process with a remote node fails with an authentication error.	Unique public/private key pairs are generated for the remote node record and the .Local node record is set to Enable Override=N.	Change the .Local node record to Enable Override=Y.
The ENCRYPT.DATA parameter, specified from the COPY statement causes the copy step to fail with error message CSPA080E.	The algorithm name used in the COPY statement is not in the supported algorithm list for both nodes.	Verify that the algorithm name in the COPY statement is in the supported algorithm list for both nodes.
Connect:Direct Secure Plus is installed, but error message CSPA001E occurs on transfers not using Connect:Direct Secure Plus.	Remote node records do not exist.	• A remote node record must exist for every node in the netmap. Use the Sync with Netmap feature to create any missing nodes. • Disable Connect:Direct Secure Plus by clicking Disable Secure+ in the .Local node record.
Signature verification fails with error message CSPA002E.	Configuration settings missing or incorrect.	• If this is a non-Secure node, make sure the remote node record has Disable Connect:Direct Secure Plus selected. • Check the Connect:Direct Secure Plus settings for the node.
Signature verification fails with error message CSPA003E, CSPA004E, or CSPA005E.	• Configuration settings missing or incorrect. • A security attack in progress.	• Execute standard operating procedure for investigating security violation.

Problem	Possible Cause	Solution
Signature verification fails with error message CSPA007E.	Expired Signature Previous Key Pair. Date exceeded or keys have been changed.	If Auto Update is disabled, check the expiration date for the signature key pair for both nodes. Check the update history log on both nodes for the last change to the record. Verify that the signature public key is correct for both nodes.
Running a Process with a remote node fails with an authentication error, CSPA008E.	Authentication Previous Key Pair Expiration Date exceeded or keys have been changed.	If Auto Update is disabled, check the authentication previous key pair expiration date for both nodes. Check the update history log on both nodes for the last change to the record. Verify the authentication public key is correct for both nodes.
Strong authentication fails with the error, CSPA010E.	• The time allowed for strong authentication expired. • A security attack in progress.	• Increase the timeout value. • Execute standard operating procedure for investigating security violation.
Connect:Direct Secure Plus session fails with the error, CSPA011E.	An illegal attempt to override Connect:Direct Secure Plus parameters.	• Turn on Enable Override in the remote node record to allow the COPY statement to override the node settings. • Check the COPY statement and remove the override statements.
Connect:Direct Secure Plus session fails with the error, CSPA014E.	Connect:Direct Secure Plus cannot read the remote node definition.	Check the remote node definition settings.
Connect:Direct Secure Plus session fails with the error, CSPA016E.	Connect:Direct Secure Plus is not enabled in the local node definition.	Make sure Connect:Direct Secure Plus is enabled for the local node.
Connect:Direct Secure Plus session fails with the error, CSPA019E.	Error generating digital signature.	• Resubmit the Process. • Call IBM Customer Support.
Connect:Direct Secure Plus session fails with the error, CSPA077E.	The COPY statement requested Connect:Direct Secure Plus parameters but Connect:Direct Secure Plus is not configured.	Remove the SECURE= parameter from the COPY statement.
Connect:Direct Secure Plus session fails with the error, CSPA079E.	Invalid encryption algorithm identified in COPY statement.	Change the ENC.DATA parameter and specify one of the following values: Y, N, IDEACBC128, TDESCBC112, or DESCBC56 and resubmit the Process.
Connect:Direct Secure Plus session fails with the error, CSPA080E.	No common algorithms are available for both nodes.	Verify the algorithm list for both nodes contains at least one common algorithm name.
Connect:Direct Secure Plus session fails with the error, CSPA091E.	Session attempted but remote node is not configured.	Make sure both nodes are defined.

Problem	Possible Cause	Solution
Connect:Direct Secure Plus session fails with the error, CSPA200E.	Both nodes are not configured for the same protocol.	Check the protocol setting at both sites and verify that the same protocol is configured at each site.
Connect:Direct Secure Plus session fails with the error, CSPA202E.	TLS protocol handshake failed.	Edit the cipher suite list and add a cipher suite used by the trading partner.
Connect:Direct Secure Plus session fails with the error, CSPA203E or CSPA204E.	The TLS protocol could not validate the server's certificate.	Make sure the certificate information is typed into the node record.
Connect:Direct Secure Plus session fails with the error, CSPA205E.	A trading partner is not using TCP/IP for communication.	Make sure that both ends of the communication use TCP/IP.
Connect:Direct Secure Plus session fails with the error, CSPA206E.	The TLS protocol could not validate the server's certificate.	Make sure the certificate information is entered into the node record.
Connect:Direct Secure Plus session fails with the error, CSPA208E.	The common name in the certificate received does not match the Connect:Direct Secure Plus configuration.	Make sure the certificate common name is spelled correctly and uses the same case as that in the certificate.
Connect:Direct Secure Plus session fails with the error, CSPA209E.	The certificate has expired or is invalid.	Obtain a new certificate and reconfigure the node record.
Connect:Direct Secure Plus session fails with the error, CSPA210E.	The COPY statement attempts to override settings in the TLS protocol.	• The system continues to operate. • If desired, change the Process statement and remove the COPY override options.
Connect:Direct Secure Plus session fails with the error, CSPA211E.	The remote trading partner failed to send a certificate.	Notify the trading partner that a certificate is required.
Connect:Direct Secure Plus session fails with the error, CSPA280E.	The trusted root certificate could not be loaded.	Check the local node configuration and make sure the location of the trusted root certificate is correctly identified.
Connect:Direct Secure Plus session fails with the error, CSPA281E.	The trusted root certificate is empty.	Check the local node configuration and make sure the location of the trusted root certificate is correctly identified.
Connect:Direct Secure Plus session fails with the error, CSPA282E.	The user certificate file cannot be loaded.	Check the local node configuration and make sure the location of the user certificate file is correctly identified.
Connect:Direct Secure Plus session fails with the error, CSPA303E.	The Secure+ parameters files have not been initialized.	Run the Secure+ Admin Tool to initialize the Secure+ parameters files.
Connect:Direct Secure Plus session fails with the error, CSPA309E.	The SSL library failed during the handshake.	Examine all related errors to determine the cause of the failure.

Problem	Possible Cause	Solution
Connect:Direct Secure Plus session fails with the error, CSPA311E.	Certificate validation failed.	Verify that the root certificate is properly configured. An alternate certificate may be required.

Configuration Worksheets

Use the worksheets in this topic to record the configuration information for Connect:Direct Secure Plus.

- The Local Node Security Feature Definition Worksheet is a record of the security functions defined for the local IBM Connect:Direct node.
- The Remote Node Security Feature Definition Worksheet is a record of the security functions defined for remote nodes. For each trading partner, define a remote node record. Make a copy of the blank Remote Node Security Feature Definition Worksheet for each remote node that you are configuring for Connect:Direct Secure Plus operations.

Local Node Security Feature Definition Worksheet

Record the security feature definition for the IBM Connect:Direct .Local node record on this worksheet.			
Local Node Name:	_______________________________________		
Configured Security Functions			
Enable TLS protocol:	Yes _______	No _______	
Enable SSL protocol:	Yes _______	No _______	
Authorization Timeout:	________________		
Trusted Root Certificate File location:	____		
Key Cert File:	____		
Certificate Passphrase:	____		
Cipher Suite(s) Enabled:	____		
External Authentication			
Enable External Authentication	Yes _______	No _______	
Certificate Validation Definition	____ _		
Enable FIPS 140-2 mode	Yes _______	No _______	

Remote Node Security Feature Definition Worksheet

Make a copy of this worksheet for each remote node defined in the IBM Connect:Direct parameters file that you are configuring for IBM Connect:Direct operations. Record the security feature definitions for a remote node record on this worksheet.		
Remote Node Name:	______________________________	
Security Options		
Protocol defined in the .Local node record:	<protocol> _______	
Is the remote node using the protocol defined in the .Local node record?	Yes _______	No _______

If you answered No to the question above, identify the protocol to use for the Remote Node:			
Note: If you do not enable the override option, IBM Connect:Direct generates an error message.			
Enable TLS protocol:	Yes _______	No _______	
Enable SSL protocol:	Yes _______	No _______	
If you want to use the same protocol defined in the local node, select Default to Local Node.			
Enable Override:	Yes _______	No _______	
Note: The COPY statement cannot override settings in SSL-enabled or TLS-enabled remote nodes.			
Authorization Timeout:	_________________		
TLS or SSL Protocol Functions			
Trusted Root Certificate File location:	_______________________________________		
Certificate File:	_______________________________________		
Certificate Passphrase:	_______________________________________		
Cipher Suite(s) Enabled:	____ _		
Enable Client Authentication:	Yes _______	No _______	Default to local node _____
Certificate Common Name:	_______________________________________		
Note: If you want to add a second level of security, enable client authentication for the remote node and type the certificate common name.			
External Authentication			
Enable External Authentication	Yes _______	No _______	Default to local node _____
Certificate Validation Definition	____ _		
Enable FIPS 140-2 mode	Yes _______	No _______	

Certificate Files

The TLS security protocol use a secure server RSA X.509V3 certificate to authenticate your site to any client that accesses the server and provides a way for the client to initiate a secure session. You obtain a certificate from a certificate authority or you can create a self-signed certificate. When you obtain a certificate file, a trusted root certificate file and key file are created. This topic describes the layout of the trusted root certificate file and the key certificate file.

Connect:Direct Secure Plus uses two certificate files to initiate TLS session: a trusted root certificate file and a key certificate file.

When you obtain a root certificate from a certificate authority, you receive a trusted root certificate file. To configure Connect:Direct Secure Plus, add the name and location of the trusted root certificate file to the node record using the Secure+ Admin Tool.

A sample trusted root certificate file called trusted.txt is installed in the Connect:Direct Secure Plus\certificates directory when you install Connect:Direct Secure Plus. Use any text editor to add or delete certificate information to this file. In simple configurations, only one trusted root certificate file is used. In more sophisticated configurations, you may associate individual trusted root files with one or more node records.

When you use a certificate signing request (CSR) tool you do not need to change the contents of the key certificate file.

If you set up your own PKI infrastructure, you may chain more than two certificates, including a CA root certificate, one or more intermediate CA certificates, and an identity certificate. You can create chained certificates using one of the following methods:

- Using the Local Key Certificate File—In a chain of two certificates, the local key certificate file contains a private key and an identity certificate. In a longer chain, the key certificate file contains the private key and the identity key, followed by the intermediate CA certificates.
- Using the Remote Trusted File— In a chain of two certificates, the remote trusted file contains the CA root certificate. In a longer chain, the remote trusted file contains the CA root certificate and all the intermediate CA certificates.

Formats

The formats discussed in this section apply to the certificate files used with Connect:Direct Secure Plus. The formats are illustrated in the sample certificate files below.

General Object Format

All objects are formatted in the Privacy Enhanced Mail (PEM) style, beginning with a line in the format. Below is a sample object format:

```
-----BEGIN <object>-----
```

and end with:

```
-----END <object>-----
```

In this sample, <object> is a placeholder for the name of the object type: CERTIFICATE or ENCRYPTED PRIVATE KEY.

Certificate Format

A certificate is encoded as a general object with the identifier string CERTIFICATE or X.509 CERTIFICATE. The base64 data encodes a Bit Error Rate (BER)-encoded X.509 certificate. This is the same format used for PEM. Anyone who provides or understands PEM-format certificates can accommodate the certificate format. For example, VeriSign commonly fulfills certificate requests with certificates in this format, SSLeay supports them, and SSL servers understand them. Both Netscape and Microsoft support this format for importing root CA certificates.

Private Key Format

A private key is encoded as a general object with the identifier string ENCRYPTED PRIVATE KEY. The base64 data encodes a BER-encoded PKCS#8 Private Key object. The passphrase associated with the Private Key is required for Connect:Direct Secure Plus and is stored in the Secure+ parameters file. Additional encryption is used to prevent the passphrase from being discovered.

Sample Certificate Files

In the sample user certificate below, a private key is followed by the server certificate, which is followed by the root certificate.

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIICCDAaBgkqhkiG9w0BBQMwDQQIIfYyAEFKaEECAQUEggHozdmgGz7zbC1mcJ2r
.
.
```

```
.
IGpupStY5rLqqQ5gwLn45UWgzy6DM96CQg6+Dyn0N9d1M5lIg2wlnUwE8vI=
-----END ENCRYPTED PRIVATE KEY-----
User/Server Certificate
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----
// Final Root Certificate (optional)
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----
```

In the sample root certificate below, the trusted.txt file contains a list of trusted root certificates.

```
RSA Commercial CA - exp. Dec 31, 2003
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----
RSA Commercial CA - exp. Dec 31, 2010
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----
```

Model Automation Scripts

The following scripts are provided as models for creating custom scripts to define your Connect:Direct Secure Plus environment and automate the implementation of it. To prevent any loss of data, you cannot run the scripts, but you can save them with a different name and modify them to suit your needs.

Configure Connect:Direct Secure Plus to Use the TLS Protocol

The spcust_sample1 script demonstrates using the Secure+ CLI to configure Connect:Direct Secure Plus to use the TLS protocol with the trusted root file, key certificates, and ciphers.

```
#! /bin/sh
#
##############################################################################
# Licensed Materials - Property of IBM
#
# Connect:Direct for UNIX
#
# (C) Copyright IBM Corp. 1992, 2014 All Rights Reserved.
#
# US Government Users Restricted Rights - Use, duplication or disclosure
# restricted by GSA ADP Schedule Contract with IBM Corp.
##############################################################################
#
# spcust_sample1.sh contains an example of configuring
# Secure+ to use SSL or TLS protocols with the Secure+ CLI.
# The example demonstrates the configuration of Secure+
# with the trusted root and key certificates and ciphers
#
#
# Variables
#
# The return code.
# spcli.sh returns the highest return code of the commands
# it executed. Possible return codes and their meanings are
#      0    success
#      4    warning
#      8    error
#     16    fatal error
RC=0
#
#
# Functions
#
#
# Custom initialization logic written by customer.
#
initCustom()
{
    # Customer adds custom initialization code here.
    echo "Init custom..."
    # rm -rf /sci/users/jlyon/cd42/ndm/secure+/nodes
}
#
# Invoke CLI to configure Secure+.
#
invokeCLI()
{
    /sci/users/jlyon/cd42/ndm/bin/spcli.sh -e 8 -li y << EOF
    ;
    display info
    ;
    ;
    ; -- Synch with netmap
    ;
    sync netmap
        path=/sci/users/jlyon/cd42/ndm/cfg/<node name>/netmap.cfg
        name=*
    ;
    ;
    ; -- Import KeyCert
    ;
    Import KeyCert
        File=<path to Key Certificate file>
        Passphrase=<KeyStore passphrase>
        Label=<optional, destination name of key certificate>
        ImportMode=<Add | Replace | AddOrReplace>
    ;
    ;
    ; -- Import TrustedCert
    ;
    Import TrustedCert
        File=<path to Trusted Certificate file>
        ImportMode=<Add | Replace | AddOrReplace>
    ;
    ;
    ; -- Update LocalNode
    ;
    Update LocalNode
        Protocol=<Comma delimited list of Protocols, see Display Protocols>
        SecurityMode=<One Security Mode, see Display SecurityModes>
        Override=<y | n>
        AuthTimeout=<nnn seconds>
        KeyCertLabel=<label of key certificate | null>
        EncryptData=<y | n>
        ClientAuth=<y | n>
        CipherSuites=<Comma delimited list of Ciphersuites | All | null>
        SeaEnable=<y | n>
        SeaCertValDef=<external authentication server certificate validation definition | null>
    ;
    ;
    ; -- Display localnode
    ;
    display localnode
    ;
    ;
    ; -- Validate parmfile
    ;
    validate parmfile
    ;
EOF
    return $?
}
#
# Custom termination logic written by customer.
#
terminateCustom()
{
    # Customer adds custom termination code here.
    # For example, E-mail standard out log for review.
    # Send error messages to system monitoring facility.
    echo "$RC"
    echo "Custom Terminating ... "
}
#
#  Main script
#
echo
echo "This script has been prevented from running because it will alter the configuration"
echo "of Secure+. Before removing this warning and its exit call, please modify the script"
echo "so that it carries out only desired modifications to the configuration of Secure+."
echo
exit
initCustom
invokeCLI
RC=$?
terminateCustom
exit $RC
```

Encrypt Passwords for use with CLI

The Secure+ CLI displays passwords in plain text. If you need to encrypt passwords for use with the Secure+ CLI, use the Local Connection Utility (LCU) to create an LCU file that contains non-encrypted information used to encrypt the password and the encrypted password, such as a keycert passphrase. You can then refer to this file when prompted for passwords.

Self-signed Certificate for Integrated File Agent

When Integrated File Agent is installed, an automatically generated self-signed certificate is added in Secure+'s .Client and .Local record if a key certificate has not already been assigned .

Self-signed certificate allows Integrated File Agent to connect securely to Connect:Direct. It also allows Connect:Direct to run a loopback (PNODE equals SNODE) process securely, which helps new customers to get started Secure+.

However, self-signed certificate is not suitable for securing production Connect:Direct to Connect:Direct and remote-API-client to Connect:Direct connections. If you want to secure these connections, you must replace the key certificate in the .Local node record with a certificate suitable for production, such as a CA certificate issued by an internal or public CA. For more details, refer “Plan Your Implementation of the TLS Protocol” on page 259.

Chapter 8. About SMF

The Service Management Facility (SMF) is a core component of the Predictive Self-Healing set of technologies introduced in Solaris 10. With SMF, system administrators use simple command line utilities to identify, observe, and manage services provided by the system, and the system itself.

This document describes how to place IBM Connect:Direct under the control of SMF.

Place IBM Connect:Direct under Solaris Service Management Facility Control

This procedure uses the following sample names and files to describe how to putIBM Connect:Direct under the control of SMF. Modify the names and files in the table for your implementation.

Note: You need either root access or a user ID with role-based access control (RBAC) authorization to add and modify SMF services. For additional information, see Implementing Solaris Role-Based Access Control with SMF for Connect:Direct.

Sample Name	**File**
Service Name	connect-direct
Service Manifest	/var/svc/manifest/network/connect-direct.xml
Fault Managed Resource Identifier (FMRI)	svc:/network/connect-direct:default
Run Script	/lib/svc/method/connect-direct
Log File	/var/svc/log/network-connect-direct:default.log

Installing and Configuring the SMF Script

1. Stop IBM Connect:Direct using the Command Line Interface (CLI). For information, refer to IBM Connect:Direct for UNIX User Guide.
2. As root, copy the following Bourne shell script file to /lib/svc/method/connect-direct.

```
#!/sbin/sh
#
# /lib/svc/method/connect-direct
#
# For Solaris SMF. This file goes in /lib/svc/method/
# Please set the top three variables before using this script.
# Notes:The CDunix Admin ID is needed below because SMF scripts are run as root at
# system boot and shutdown. And since the root ID is not in the Connect:Direct
# userfile.cfg, the root ID does not have permission to "stop" C:D.
# We don't want to put the root ID in the userfile.cfg just to get around this.
# So we su to the CDunix Admin ID that installed Connect:Direct, (or su to an
# ID with permissions to stop Connect:Direct). Then issue the Direct CLI
# command to stop the Connect:Direct service.
# Also, a "stop force;" is given in the Connect:Direct CLI in order to have
# Connect:Direct do an immediate clean stop. A simple "stop;" would wait for
# currently running Connect:Direct jobs to complete. However, the OS is not
# going to wait for C:D to complete long running jobs. Instead, after a few
# moments the OS would finally kill the cdpmgr, which is generally not what we
# want. However, using a "stop force;" the cdpmgr will take a checkpoint of
# job progress and cleanly shut down. Then after Connect:Direct restarts it
# will pick up from that last checkpoint.
# For further questions about this script, contact IBM Customer Support.
. /lib/svc/share/smf_include.sh
# Installation path
CDUNIX_HOME=/opt/cdunix
# Name of this Connect:Direct instance.
CDUNIX_NODE_NAME=chicago_CD
# The ID that installed Connect:Direct or an ID with authority to issue
# the "stop" command.
CDUNIX_ADMIN=jsmith
```

```
# Do not change the remainder of this file unless you require different behavior.
startup()
{
INITPARM_CONF=${CDUNIX_HOME}/ndm/cfg/${CDUNIX_NODE_NAME}/initparm.cfg
[ ! -f ${INITPARM_CONF} ] && exit $SMF_EXIT_ERR_CONFIG
[ ! -x ${CDUNIX_HOME}/ndm/bin/cdpmgr ] && exit $SMF_EXIT_ERR_PERM
exec ${CDUNIX_HOME}/ndm/bin/cdpmgr -i ${INITPARM_CONF} 2>&1
}
shutdown()
{
[ ! -f ${CDUNIX_HOME}/ndm/cfg/cliapi/ndmapi.cfg ] && \
exit $SMF_EXIT_ERR_CONFIG
# Don't exec this. C:D may already be down. If so, CLI returns 8
# and SMF will then put us in "maintenance" mode.
su ${CDUNIX_ADMIN} -c \
"NDMAPICFG=${CDUNIX_HOME}/ndm/cfg/cliapi/ndmapi.cfg; \
export NDMAPICFG; \
echo 'stop force;' | ${CDUNIX_HOME}/ndm/bin/direct" 2>&1
}
case "$1" in
start)
startup
;;
stop)
shutdown
exit 0
;;
refresh|restart)
    shutdown
startup
;;
*)
echo "Usage: $0 {start|stop|restart}"
exit 1
;;
esac
```

3. To match your installation, edit the following shell variables: CDUNIX-HOME, CDUNIX_NODE_NAME, and CDUNIX_ADMIN.

4. Verify that the owner and permissions of this script file match those of the other scripts in the /lib/svc/ method directory.
5. To test the /lib/svc/method/connect-direct script:
   a) Change to the /lib/svc/method/ directory.
   b) As root, start the IBM Connect:Direct service by typing the following command:**`./connect-direct start`**
   c) Verify that the IBM Connect:Direct service is running.
   d) As root, stop the IBM Connect:Direct service by typing the following command:**./connect-direct stop**
   e) Verify that the IBM Connect:Direct service stopped.
6. As root, copy the SMF Service Manifest in the following sample to /var/svc/manifest/application/ connect-direct.xml.

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<!--
  Example SMF Service manifest for Connect:Direct for UNIX on Solaris.
-->
<service_bundle type='manifest' name='Connect:Direct'>
<service
        name='application/connect-direct'
        type='service'
        version='1'>
        <!--
          If we have multiple instances of application/connect-direct provided
          by different licensed implementations, we keep dependencies
          and methods within the instance.
        -->
        <instance name='default' enabled='false'>
                <!--
                  Wait for network interfaces to be initialized.
                -->
                <dependency name='network'
                    grouping='require_all'
                    restart_on='error'
                    type='service'>
                    <service_fmri value='svc:/milestone/network:default'/>
                </dependency>
                <!--
                  Wait for all local filesystems to be mounted.
                -->
                <dependency name='filesystem-local'
                    grouping='require_all'
                    restart_on='none'
                    type='service'>
                    <service_fmri
                        value='svc:/system/filesystem/local:default'/>
                </dependency>
                <!--
                  Wait for automounting to be available, as we may be
                  transfering data from home directories or other remote
                  filesystems.
                -->
                <dependency name='autofs'
                    grouping='optional_all' restart_on='error'
                    type='service'>
                    <service_fmri
                        value='svc:/system/filesystem/autofs:default'/>
                </dependency>
                <exec_method
                        type='method'
                        name='start'
                        exec='/lib/svc/method/connect-direct start'
                        timeout_seconds='30' >
                        <method_context>
                                 <method_credential user='root' group='root' />
                        </method_context>
                </exec_method>
                <exec_method
                        type='method'
                        name='stop'
                        exec='/lib/svc/method/connect-direct stop'
                        timeout_seconds='30' >
                        <method_context>
                                <method_credential user='root' group='root' />
                        </method_context>
                </exec_method>
                <exec_method
                        type='method'
                        name='refresh'
                        exec='/lib/svc/method/connect-direct restart'
                        timeout_seconds='45' >
                        <method_context>
                                <method_credential user='root' group='root' />
                        </method_context>
                </exec_method>
                <property_group name='cdpmgr' type='application'>
                        <stability value='Evolving' />
                </property_group>
                <property_group name='startd' type='framework'>
                        <propval name='ignore_error'
                                 type='astring'
                                 value='core,signal' />
                </property_group>
                <property_group name='general' type='framework'>
                        <propval name='action_authorization' type='astring'
                           value='solaris.smf.manage.connect-direct' />
                        <propval name='value_authorization'  type='astring'
                           value='solaris.smf.manage.connect-direct' />
                </property_group>
        </instance>
        <stability value='Evolving' />
        <template>
                <common_name>
                        <loctext xml:lang='C'>
                                Connect Direct for UNIX
                        </loctext>
                </common_name>
                <documentation>
                        <doc_link name='ibm.com'
                           uri='http://www.ibm.com/
                                support/knowledgecenter/en/SS4PJT/landing/cd_welcome.html'/>
                </documentation>
        </template>
</service>
</service_bundle>
```

7. Verify that the owner and permissions of the manifest of the connect-direct.xml file match those of the other xml files in the /var/svc/manifest/application/ directory.
8. Verify that the Connect:Direct FTP+ service stopped.

Controlling IBM Connect:Direct Using the SMF Script

To place the IBM Connect:Direct service under the control of SMF:

1. As root, import the file by typing the following command:

   /usr/sbin/svccfg import /var/svc/manifest/application/connect-direct.xml
2. As root, start the IBM Connect:Direct service under SMF control by typing the following command:

   svcadm enable connect-direct

   Verify the IBM Connect:Direct service is running.
3. To verify IBM Connect:Direct restarts under the control of SMF, type the following command:

   svcs -p connect-direct

   Observe the Process number in use.
4. Stop the IBM Connect:Direct service by using the Command Line Interface (CLI) method. For additional information, refer to IBM Connect:Direct *User's Guide*.

   IBM Connect:Direct stops, and immediately restarts under SMF control.
5. To confirm IBM Connect:Direct is under the control of SMF, type the following command:

   svcs -p connect-direct

   The system generates a Process number different from that observed in step 3.

Implementing Solaris Role-Based Access Control with SMF for IBM Connect:Direct

Implementing role-based access control (RBAC) is optional. After you place under the control of SMF, only the root ID or a user ID with RBAC authorization for SMF is authorized to issue the SMF commands to stop and start IBM Connect:Direct. To authorize additional specific user IDs to stop and start IBM Connect:Direct, you must implement basic RBAC to grant authority to the user.

Many solutions exist for setting up RBAC on Solaris. If you frequently add users to or remove users from administration, consider creating role accounts and profiles. For additional RBAC information, see the *Solaris System Administration Guide: Basic Administration* and *Solaris System Administration Guide: Advanced Administration*. Consider using the following procedure if you enable only a few users.

1. Open the file: /etc/security/auth_attr.
2. Add the following line anywhere in the file:

   solaris.smf.manage.connect-direct:::Manage Connect Direct Service States::

   The corresponding FMRI manifest entry copied to connect-direct.xml eliminates the need to edit the connect-direct.xml manifest file.
3. As root, type the following command, substituting the user ID you want to authorize for userID**:**

   usermod -A solaris.smf.manage.connect-direct userID
4. If this message appears: usermod: ERROR: userID is not a local user, then do the following:

   Open the file: /etc/user_attr.

   Add the following line anywhere in the file, substituting the user ID you want to authorize for userID: userID::::type=normal;auths=solaris.smf.manage.connect-direct
5. If an entry for your user ID already exists in the /etc/user_attr file, merge the entries. You only merge the auths portion, which is a comma-delimited list of entries found in /etc/security/auth_attr.

The user ID is authorized to control only and can issue commands, including the following:

- svcadm enable connect-direct
- svcadm disable connect-direct
- svcadm refresh connect-direct

Starting and Stopping IBM Connect:Direct under SMF Control

After the IBM Connect:Direct service is under the control of SMF, the service starts when you boot the Solaris system, and stops when you shut down the Solaris system. Examples are provided of basic SMF commands to control the IBM Connect:Direct service. For simplicity, the examples use the FMRI shortcut name connect-direct in place of the full FMRI instance name, svc:application/connect-direct.

If you add service manifest to control IBM Connect:Direct at a later time, use the full FMRI name to avoid ambiguity. The example commands assume that only one FMRI exists for IBM Connect:Direct on this system, so the shortcut name is used.

Note: The stop commands issued from the IBM Connect:Direct CLI stop the IBM Connect:Direct service; however, SMF immediately restarts it. This behavior gives the impression that the stop commands function improperly. To eliminate confusion when you stop the service, advise IBM Connect:Direct users of the following SMF control commands: Starting the Connect:Direct Service, and Stopping the Connect:Direct Service.

Starting the Connect:Direct FTP+ Service

To start the service, as root type the following command:

/usr/sbin/svcadm enable connect-direct

Service starts running, and restarts when the Solaris system boots.

Stopping the IBM Connect:Direct Service

To stop the service, as root type the following command:

/usr/sbin/svcadm disable connect-direct

The service stops, and remains disabled when the Solaris system boots.

After you disable the service, you can stop and start IBM Connect:Direct using the same method used before IBM Connect:Direct was placed under the control of SMF.

Correcting Errors in the Script or FMRI File

If you modify the SMF script or the FMRI file and errors result, SMF places IBM Connect:Direct into a maintenance state. Correct the affected script or FMRI file.

To signal to the assigned restarter that the service is repaired, as root type the following command:

/usr/sbin/svcadm clear connect-direct

Removing IBM Connect:Direct from SMF Control

To remove IBM Connect:Direct from SMF control, stop IBM Connect:Direct using the svcadm disable method:

1. As root, type the following command:

   ```
   /usr/sbin/svccfg delete svc:application/connect-direct:default
   ```

2. Delete the following files:

   - /var/svc/manifest/network/connect-direct.xml
   - /lib/svc/method/connect-direct

   For additional information, see Managing Services in the *Solaris System Administration Guide:Basic Administration*.

Notices

Trademarks

Terms and conditions for product documentation

Applicability

Personal use

Commercial use

Rights

Part Number:
Product Number: 5655-X01

(1P) P/N: