Connect:Direct® for UNIX SNMP Agent

System Guide

Version 3.8

Sterling Commerce
An IBM Company

Connect:Direct for UNIX SNMP Agent System Guide
Version 3.8

First Edition

Sterling Commerce, Inc.

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

Contents

Chapter 1

About Connect:Direct for UNIX SNMP Agent

The Connect:Direct for UNIX SNMP Agent is a proxy agent that enables a Connect:Direct server to provide information to SNMP network management stations. The data that is sent to the SNMP agent is similar to the information available through the select statistics and select process commands in Connect:Direct. The remote monitoring provides access to the following information:

- General condition of the Connect:Direct server
- Alerts for events requiring further investigation, such as possible security violations, failing Processes, and session failure

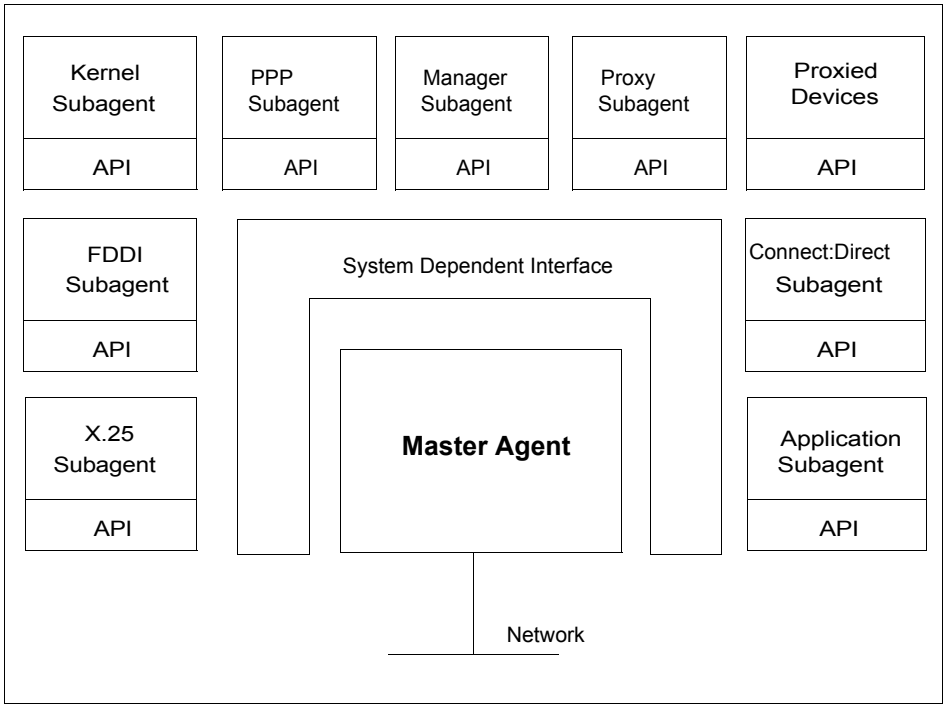
The Connect:Direct for UNIX SNMP Agent runs as a Subagent to the Master Agent. The Master Agent is the agent protocol engine and performs authentication, authorization, access control, and privacy functions. It is multi-threaded and communicates asynchronously with Subagents. Network management systems recognize these two entities as one agent.

Understanding the SNMP Architecture

The Master Agent directs retrieval and write processing, performs most of the trap processing, enrolls Subagents when they connect and removes them when they disconnect, and determines which Subagents receive a request.

Subagents perform the specific functions that collect or calculate the values for the Message Information Base (MIB) variable and pass that information to the Master Agent to send to the requesting manager. Messages are passed asynchronously between the Master Agent and the Subagent. Subagents can start in any order.

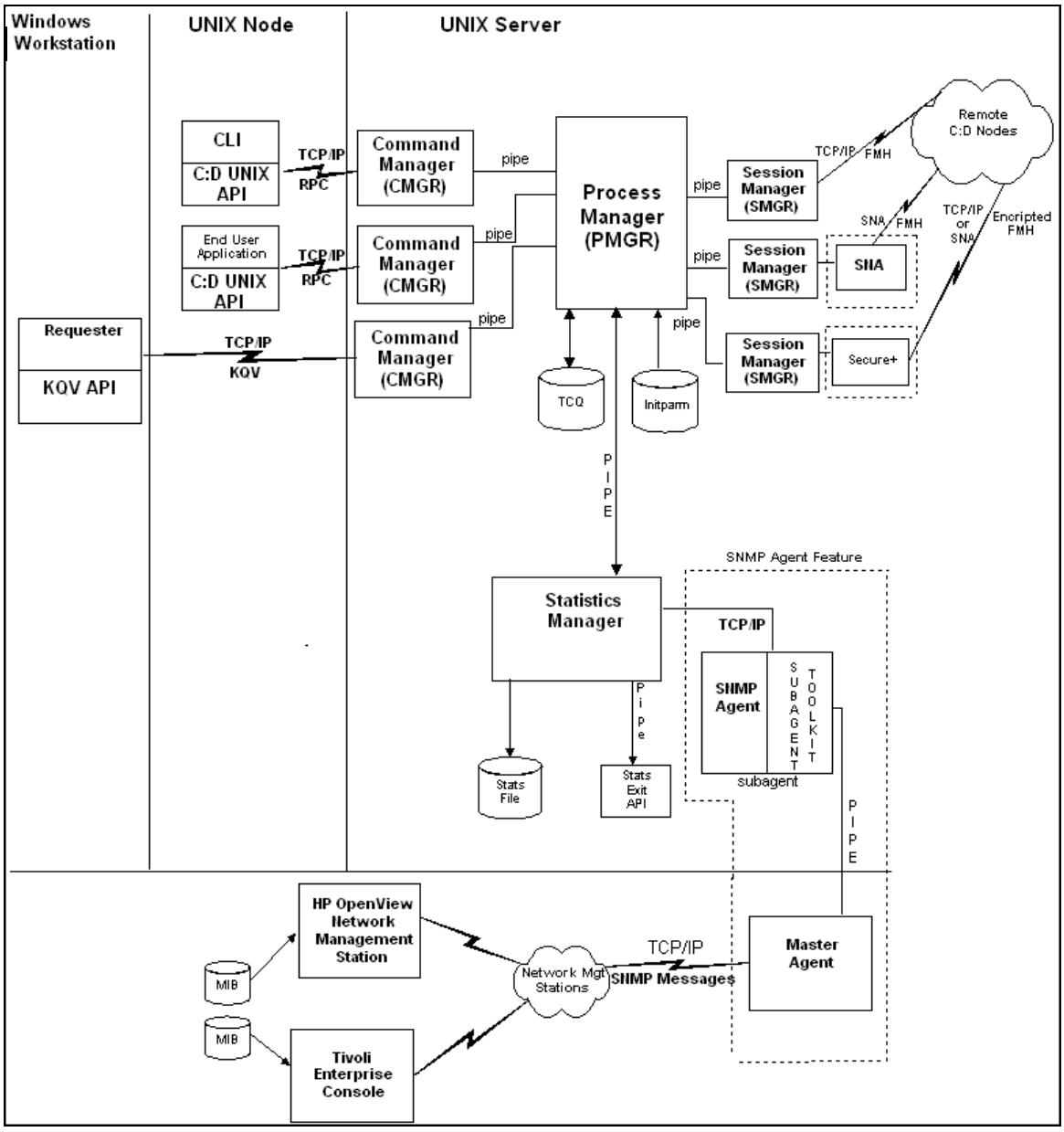
The following illustration displays the relationship between the Connect:Direct for UNIX SNMP Agent, SNMP Agent, other Subagents, and the Master Agent:



Note: The SNMP Agent architecture has many Subagents. However, when an SNMP request is sent to another network management system, the remote system recognizes one agent and does not need information about Subagents.

About the Connect:Direct for UNIX SNMP Agent Architecture

The following illustration displays the relationship between Connect:Direct for UNIX SNMP Agent and the SNMP Agent:



Connect:Direct for UNIX Documentation

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

About This Guide

The *Connect:Direct for UNIX SNMP Agent System Guide* document is for programmers and network operations staff who install and maintain the Connect:Direct for UNIX SNMP Agent with Connect:Direct for UNIX.

Read the first three chapters in the book to gain the general knowledge required to install and configure the Connect:Direct for UNIX SNMP Agent product. These chapters introduce you to the basic components and general concepts, and they summarize the preinstallation and installation procedures.

This guide assumes knowledge of the UNIX operating system, including its applications, network, and environment. If you are not familiar with the UNIX operating system, refer to the UNIX library of manuals.

Task Overview

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

Task	Reference
Understanding the SNMP Agent, its components, and architecture.	Chapter 1, *About Connect:Direct for UNIX SNMP Agent*
Installing the SNMP Agent	Chapter 2, *Installing the Connect:Direct for UNIX SNMP Agent*
Configuring the SNMP Agent on a new system	Chapter 3, *Configuring the SNMP Agent*
Preparing and using the SNMP Master Agent and using nonstandard ports	Chapter 4, *Connect:Direct for UNIX SNMP Master Agent*
Preparing and using the SNMP Subagent	Chapter 5, *Connect:Direct for UNIX SNMP Subagent*
Troubleshooting common problems that occur when using the SNMP agent	Appendix A, *Troubleshooting*
Defining MIS subagents	Appendix B, *Defining MIB Subagents*

Chapter 2

Installing the Connect:Direct for UNIX SNMP Agent

This section describes the tasks necessary to install the Connect:Direct for UNIX SNMP Agent.

Preparing for the Installation

Before you install Connect:Direct for UNIX SNMP Agent, read the *Connect:Direct for UNIX Release Notes* for any additional system requirements and upgrade considerations.

The Connect:Direct for UNIX software is shipped on the Connect:Direct CD-ROM, and it is also available for downloading from the ESD Portal. After you install Connect:Direct for UNIX, you can choose to immediately install the Connect:Direct for UNIX SNMP Agent by selecting option 6 from the main installation menu, or you can install the agent at another time as described in *Installing SNMP Agent* on page 10.

Before you begin the installation procedure, consider the following information:

- Privileges—You may need root privileges to install the configuration files. If you do not have root privileges, arrange for them before you install the Connect:Direct for UNIX SNMP Agent. Running the agent requires root privileges because the agent needs exclusive access to a privileged port (161 or 162 if traps are sent).
- Configuration files—A set of configuration files is located in the config directory. By default, these files are placed in the /etc/srconf directory of your system. You can install them separately if necessary.

- System requirements—Ensure the SNMP ports are defined in your system services by checking the /etc/services file on your system for the following entries:

```
snmp 161/udp
snmp-trap 162/udp
```

If these entries are not in the file, add them.

Note: Only one agent may have access to port 161 at a time; therefore, only one agent can be running at any time.

Installing SNMP Agent

You can install SNMP Agent immediately after you install Connect:Direct for UNIX or you can install the agent at another time.

If you choose to install the SNMP Agent immediately after you install Connect:Direct for UNIX, start with step 8 on page 13.

If you are installing SNMP Agent into a previously created (or existing) Connect:Direct for UNIX installation, complete the following steps:

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

Caution: Do not install as root.

2. You are prompted for the path to the installation files. Depending upon how you obtained the product, take one of the following actions:
   - If you have the product CDs, select the CD for your platform. Type the following command and press **Enter** to change to the CD-ROM drive and the directory that correspond to the UNIX platform:

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

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

UNIX Platform	Platform Directory Path
HP 9000 series	HP_PA-RISC
IBM System p5	IBM
Sun SPARC systems	Sun_Solaris

- If you downloaded the product from the ESD Portal, type the following command and press **Enter** to change to the location of the product download:

```
$cd /<location of product download>/cdunix
```

3. Type the following command to invoke the installation script and press **Enter**:

```
$ cdinstall
```

The following screen is displayed:

```
Sterling Commerce, Inc.,(TM) Connect:Direct(TM) UNIX(TM)
Installation Procedure

You are beginning the Connect:Direct for UNIX Installation
Procedure. You will be asked to specify a directory (called
the destination directory) where the Connect:Direct for UNIX
files will be stored.

Please follow the Getting Started Guide and/or Release Notes for the
proper Media Name.

Sterling Commerce, Inc.(TM) and Connect:Direct(TM) are trademarks
of Sterling Commerce, Inc. in the U.S.A. and other countries.

UNIX is a registered trademark of The Open Group
================================================================

Press ENTER when ready.
```

4. Read the information and press **Enter**. The following screen is displayed:

```
Enter the FULL path of the directory where
Connect:Direct for UNIX 3.8.00 will be installed.
You can use $HOME to shorten the name:[$HOME/cdunix]
```

5. Type the path and press **Enter**.

```
WARNING: "<directory path>" : Directory exists
Files in <directory path> could be overwritten by the
installation. Do you want to continue: [Y/n]
```

6. Press **Enter** to continue. The following screen 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 C:D for UNIX versions 3.4 or 3.5 that are on a
maintenance level prior to December 2004, then any existing Processes
in the TCQ 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** 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 SNMP Agent server and client(CLI/API)
 (2) Connect:Direct for UNIX SNMP Agent server
 (3) Connect:Direct for UNIX SNMP Agent client(CLI/API)
 (4) Connect:Direct for UNIX SNMP Agent File Agent
 (5) Connect:Direct for UNIX SNMP Agent Secure+ Option
 (6) Connect:Direct for UNIX SNMP Agent SNMP Agent
 (7) EXIT

Enter your choice:[1]
```

8. Type **6** and press **Enter**. The following message is displayed:

```
=====================================================================
Sterling Commerce, Inc., (TM) Connect:Direct SNMP Agent (TM)
Installation Procedure

You are beginning the Connect:Direct SNMP Agent Installation
Procedure.  You will be asked to specify a directory (called
the destination directory) where the Connect:Direct for UNIX SNMP Agent Server
files were stored.

After the files are extracted from the media, log on as root
and run snmp_cust.sh to configure the SNMP Agent.

Sterling Commerce, Inc.(TM) and Connect:Direct(TM) are trademarks
of Sterling Commerce, Inc. in the U.S.A. and other countries.

UNIX is a trademark of UNIX Systems Laboratories, Inc.
=====================================================================

Press ENTER when ready.
```

9. Press **Enter** to continue. The following screen is displayed:

```
Enter the path and filename of where the CD_ROM installation file
is mounted (e.g. /cdrom/IBM/cdunix):
```

10. Type the location of the installation file and press **Enter**.
11. If SNMP Agent was previously installed on your system, the following screen is displayed:

```
Installed configuration files detected in /etc/srconf
will be renamed to *.saved before the new configuration files are installed.
The renamed files may be used to reapply any customization to the new files.
Please press ENTER to continue:
```

12. Press **Enter** to continue. The following message is displayed:

```
Installing configuration files in: /etc/srconf
Where is Connect:Direct installed(full path please): /mnt/home/cdunix
```

13. Type the full directory path where Connect:Direct is installed and press **Enter**. The Connect:Direct installation extracts the necessary files. When the installation is complete, you are ready to configure the SNMP agent.

Configuring the SNMP Agent Initialization File

The customization script is automatically launched after the installation is complete. It merges the SNMP Agent MIB objects into the configuration file. It is also responsible for setting up the agent initialization file. After you complete the installation steps, the script automatically allows you to create or edit an initialization file. Complete the following procedure to configure the SNMP Agent initialization file:

1. Verify that group daemon is added to the /etc/group file.
2. Open the *initparm.cfg* file in any UNIX text editor such as vi.
3. Activate the SNMP Agent by setting the following variable:

```
:snmp.agent.activated=y:
```

4. Close the *initparm.cfg* file.
5. Change to the directory called *d_dir*/cdunix/ndm/snmp where *d_dir* is the directory where you installed Connect:Direct for UNIX.
6. As root, type the following command to start the *snmp_cust.sh* file.

```
snmp_cust.sh
```

The following screen is displayed:

```
Please follow the instructions in your Connect:Direct SNMP Agent System
Guide for changing the agent initialization file. If you do not have
an existing initialization file, please answer "n for NO". The
customization script will look for an existing initialization file
called "initfile". If it does not exist, a new "initfile" will
be created for you. All prompts must match the contact.name,
contact.phone, and descrip fields in the netmap.cfg local.node record.

Change your existing initialization file [Y/n]? n
```

7. Do one of the following:
   - Type **n** to create a new initialization file for a new SNMP Agent installation and press **Enter**. A confirmation is displayed if you create a new file.
   - Type **y** to update an existing initialization file and press **Enter**.

   The following screen is displayed:

```
Enter Connect:Direct Node Name: a.box33
```

8. Type the following information when prompted and press **Enter** after typing each entry.
   - Name of the network administrator
   - Phone number for the network administrator
   - A description for the network administrator
9. Do one of the following:
   - Type the default listening port and press **Enter**.
   - Press **Enter** to accept the default port.

   The Connect:Direct for UNIX SNMP Agent installation is complete. Now you are ready to install the master agent for Connect:Direct for UNIX.

Chapter 3

Configuring the SNMP Agent

This chapter describes how to configure the Connect:Direct for UNIX SNMP Agent including the Master Agent and the Subagent. You must configure the SNMP agent before running it in a new environment or on a new computer, in a new subnet. You must run the Master Agent that is included with Connect:Direct for UNIX.

To simplify initial product testing, a sample configuration file is included with the software. Replace the sample configuration file with a custom configuration before using the SNMP agent in a production environment.

About the SNMP Configuration File

The SNMP agent uses a configuration file called `snmpd.cnf` to define security access rights, default MIB-II values, and Master Agent performance parameters. By default, the agent configuration files are located in the /etc/srconf/agt directory.

If you want to change the directory that contains the agent configuration files, type one of the following commands to set the environment variable SR_AGT_CONF_DIR.

For C shell, type the following command:

```
# setenv SR_AGT_CONF_DIR /tmp/myconf
```

For K shell, type the following command:

```
# SR_AGT_CONF_DIR=/tmp/myconf
# export SR_AGT_CONF_DIR
```

Summary Configuration Steps

To configure an SNMP Agent, complete the following tasks:

1. Configure MIB-II system group variables
2. Configure authentication-failure traps
3. Define security access rights for SNMP management applications
4. Configure performance parameters for the Master Agent

Configuring MIB-II System Group Variables

The MIB-II system group identifies the system on which the SNMP agent is running. The following sample configuration file displays the system group variables:

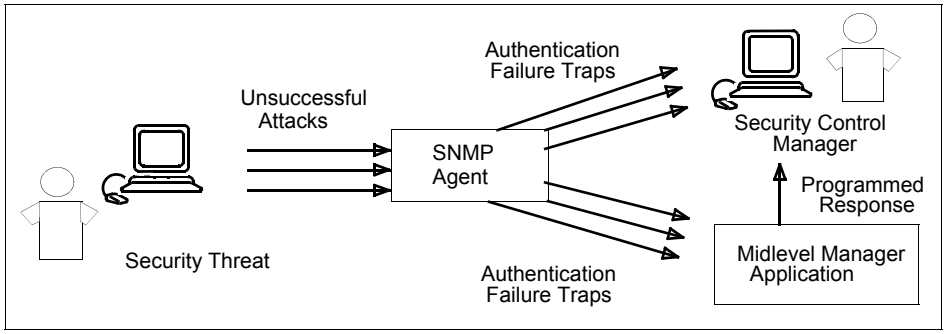
```
sysDescr "Fred Router from Flintstones, Inc."
sysObjectID 1.3.6.1.4.1.4242.1.1
sysLocation "Telephone closet, 3rd floor"
sysContact "Sterling Commerce, Inc. (800) 555-1234"
sysName fred.snmp.com
```

To configure the default values for MIB-II, add lines to the snmpd.cnf file and type the appropriate default value for each variable. The variables are described in the following table:

Variable Name	Description
sysDescr	A description of the managed entity, including the full name and version of the hardware type, operating system, and networking software.
sysObjectID	The authoritative identification of the network management subsystem contained in the entity. This value is allocated with the SMI enterprises subtree **1.3.6.1.4.1** and provides a means for determining what kind of system is being managed. For example, if the vendor "Flintstones, Inc." was assigned the subtree **1.3.6.1.4.1.4242**, it could assign the identifier **1.3.6.1.4.1.4242.1.1** to its "Fred Router."
sysLocation	The physical location of the managed entity.
sysContact	The contact person for the managed entity and information on how to contact this person.
sysName	The fully qualified domain name of the node.

Configuring Authentication-Failure Traps

When an SNMP agent receives a management request, the agent first checks the security configuration. If the request passes the security check, the agent attempts the management request. If a packet fails the security check and the authentication failure traps are enabled, the agent discards the packet and sends authentication-failure traps. The following figure illustrates how authentication-failure traps are generated. You must enable authentication failure traps if you want to generate a trap when an authentication failure occurs.



Use the snmpEnableAuthenTraps variable to enable authentication-failure traps.

- To enable authentication-failure traps, add the following line to the snmpd.cnf file:

```
snmpEnableAuthenTraps  1
```

- To disable traps, set the variable snmpEnableAuthenTraps to 2. Authentication-failure traps are disabled by default.

Configuring Security Access Rights

Security configuration is essential for an SNMP agent to determine which SNMP management applications can access which MIB variables. It also determines the degree of access for each management application.

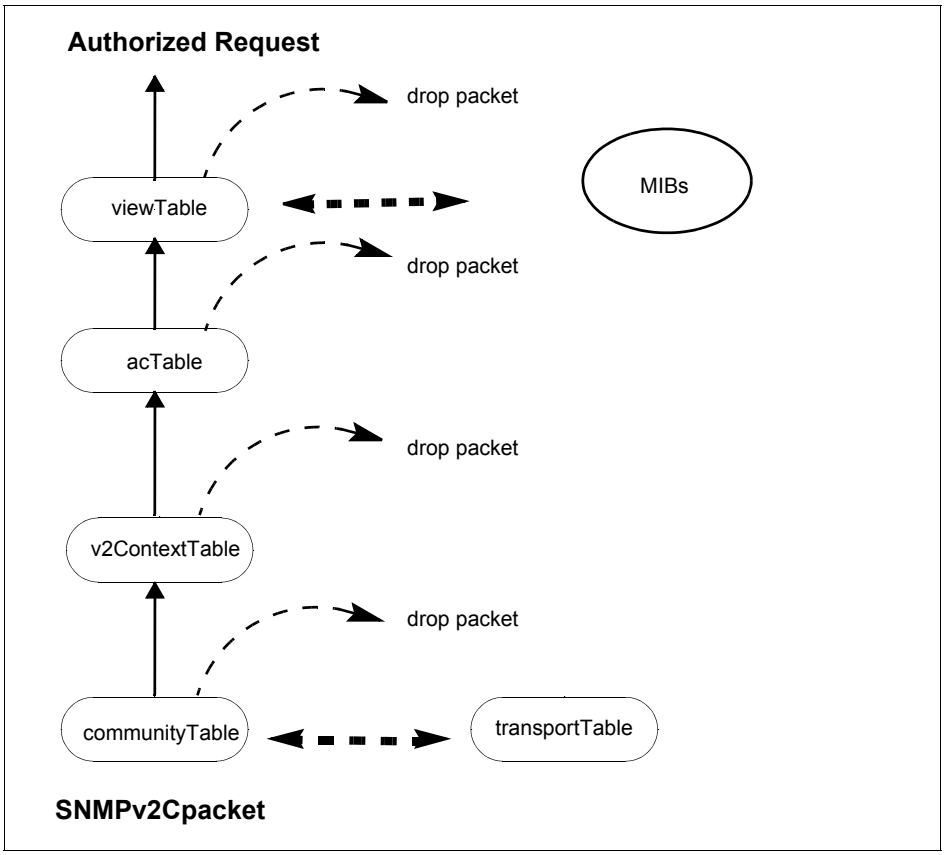
SNMPv1 and SNMPv2c agents check a list of authorized community variables before allowing or denying access. SNMP agents can also check the source address to ensure that information is being sent from an authorized location. SNMPv2 agents can also use authentication and security mechanisms to more securely authorize a user to access certain MIB variables.

Bilingual Agent Packet Processing

To configure an SNMPv2c-enabled agent, it helps to understand the relationship between the SNMPv2c MIB tables and the packet processing performed by the agent. Use the figures in this section when creating or modifying the configuration file.

For SNMPv1 and SNMPv2c packets, the agent performs the following tasks:

1. The agent searches for the community string in the communityTable to find a groupName, a contextName, and a transport label.
   - If the community string is not found in the communityTable, the packet is dropped.
   - If the transport label exists, the source address is validated. The agent searches for the transport label in the transportTable to determine what source addresses can use that community string. If the source address does not match, the packet is dropped.
2. The contextTable is checked to ensure that the context extracted from the communityTable exists. If the context does not exist, the packet is dropped.
3. The acTable is checked to ensure that the combination of SNMP version, groupName, and contextName is authorized to perform the type of operation in question (Get or Set). The appropriate viewName is also extracted from the acTable (acReadViewName or acWriteViewName).
4. The agent then processes the PDU portion of the packet, using the viewName to look up entries in the viewTreeTable. This is necessary to determine which VarBinds contained in the PDU are accessible.



Configuring Contexts

To configure a context, add a line to the snmpd.cnf configuration file using one of the following parameters:

```
v2ContextSnmpID v2ContextName v2ContextLocalEntity \
v2ContextLocalTime v2ContextMemoryType
```

The format of the VALUE clause includes the following conditions:

- **v2ContextSnmpID** is an OctetString, usually **localSnmpID**
- **v2ContextName** is the unique context identifier in ASCII text
- **v2ContextLocalEntity** is a nonunique string associated with the context
- **v2ContextLocalTime** is an integer, zero for a new context
- **v2ContextMemoryType** is nonVolatile, permanent, or readOnly

Note: The default context, v2ContextName = default, is created automatically and it is always available. Be sure to modify this value before using the configuration file.

Configuring Access Control

To configure access control, add a line to the snmpd.cnf configuration file using one of the following parameters:

```
acSPI acGroupName acContextName acContextNameMask \
acPrivs acReadViewName acWriteViewName acMemoryType
```

The format of the VALUE clause includes these conditions:

- **acSPI** is snmpv1, snmpv2c, usecNoAuth, usecAuth, or usecPriv.
- **acGroupName** is the group name in ASCII text.
- **acContextName** is the context name in ASCII text.
- **acContextNameMask** is an OctetString represented as a sequence of hexadecimal numbers separated by colons. Each octet is within the range 0x00 through 0xff. A zero-length OctetString is represented with a dash (-).
- **acPrivs** is nothing, readOnly, or readWrite.
- **acReadViewName** is a **viewTreeEntry**.
- **acWriteViewName** is a **viewTreeEntry**.
- **acMemoryType** is nonvolatile, permanent, or readOnly.

Configuring an MIB View Subtree

To configure an MIB view subtree, add a line to the snmpd.cnf configuration file using one of the following parameters:

```
viewTreeName viewTreeSubTree viewTreeMask
viewTreeType viewTreeMemoryType
```

The format of the **VALUE** clause includes these conditions:

- **viewTreeName** is the MIB view identifier in ASCII text.
- **viewTreeSubTree** is the OID of the MIB subtree.
- **viewTreeMask** is an OctetString represented as a sequence of hexadecimal numbers separated by colons. Each octet is within the range 0x00 through 0xff. A zero-length OctetString is represented with a dash (-).
- **viewTreeType** is included or excluded and indicates if the **viewTreeSubTree** is explicitly accessible or not accessible in this MIB view.
- **viewTreeMemoryType** is nonVolatile, permanent, or readOnly.

Restricting an MIB View Subtree

The **v**iewTreeMask field allows you to further restrict the MIB view at a finer granularity than that of the viewTreeSubTree and viewTreeType pair. For instance, a view can be restricted to one row of a table as illustrated in the following example.

When a dash (-) is specified for the value, it causes the corresponding viewTreeMask to be a NULL string, which in turn enables all entries below the viewTreeSubtree entry to be visible, unless canceled by another entry in the configuration file.

Note: Consult the SNMPv2 documentation for further explanation of the viewTreeMask string.

The viewTreeMask is built using octets that correspond to the OID being restricted. For example, you may want to restrict the view of the ifTable to only the second row, all columns. The OID for ifEntry.0.2 is:

```
1.3.6.1.2.1.2.2.1.0.2
```

The viewTreeMask is a series of ones and zeros used for masking out parts of the tree. A zero indicates a wildcard (matches anything), and a one indicates an exact match must be made.

OID	1	. 3	. 6	. 1	. 2	. 1	. 2	. 2	. 1	. 0	. 2
viewTreeMask	1	1	1	1	1	1	1	1	1	0	1

The previous table requires an exact match on all fields except the table column, for example, the 0 in ifEntry.0.2.

Using the preceding example, the bits of the viewTreeMask are grouped into bytes, and the right end is padded with ones if necessary to fill out the last byte.

byte 1								byte 2								
1	1	1	1	1	1	1	1	1	0	1						original mask
1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	padded with 1s
ff								bf								hex value

The viewTreeMask entry is listed in the following sample.

```
ff:bf
```

If you define the preceding value for viewTreeMask in the configuration file, the **getmany** command on the ifTable returns the following entries:

```
% getmany localhost public ifTable
ifIndex.2 = 2
ifDescr.2 = lo0
ifType.2 = softwareLoopback(24)
ifMtu.2 = 1536
ifSpeed.2 = 0
ifPhysAddress.2 =
ifAdminStatus.2 = up(1)
ifOperStatus.2 = up(1)
ifLastChange.2 = 0
ifInUcastPkts.2 = 182945
ifInErrors.2 = 0
ifOutUcastPkts.2 = 182949
ifOutErrors.2 = 0
ifOutQLen.2 = 0
ifSpecific.2 = ccitt.0
```

Note: The second row must exist before you can retrieve it.

Configuring Communities

To configure an SNMPv1 or SNMPv2c community, define an entry for the communityEntry TAG parameter in the snmpd.cnf configuration file, as illustrated in the following sample entries:

```
communityAuthSnmpID communityName communityGroupName \
communityContextSnmpID communityContextName \
communityTransportLabel communityMemoryType
```

The format of the **VALUE** clause includes these conditions:

- **communityAuthSnmpID** is an OctetString, usually localSnmpID.
- **communityName** is the community string in ASCII text.

- **communityGroupName** is the community group name in ASCII text.
- **communityContextSnmpID** is an OctetString, usually localSnmpID.
- **communityContextName** is the context of the community.
- **communityTransportLabel** is a transportEntry.
- **communityMemoryType** is nonVolatile, permanent, or readOnly.

Configuring Traps and Inform Requests

To configure traps (SNMPv1 or SNMPv2c) and inform requests (SNMPv2c), add one of the following notifyEntry TAG entries in the snmpd.cnf configuration file:

```
notifyIndex notifySPI notifyIdentityName \
notifyTransportLabel notifyContextName notifyViewName \
notifyMemoryType
```

The format of the **VALUE** clause includes these conditions:

- **notifyIndex** is a unique integer.
- **notifySPI** is snmpv1, snmpv1_5, usecNoAuth, usecAuth, or usecPriv.
- **notifyIdentityName** is the community string to authenticate the trap or inform.
- **notifyTransportLabel** is a transportEntry.
- **notifyContextName** is the context that authenticates the trap or inform.
- **notifyViewName** is a MIB view, viewTreeName value, that authenticates the trap or inform.
- **notifyMemoryType** is nonVolatile, permanent, or readOnly.

Configuring Transports

To configure a transport, add one of the following **transportEntry TAG** entries to the snmpd.cnf configuration file:

```
transportLabel transportSubIndex transportDomain \ transportAddress
transportReceiveMask transportMMS \ transportMemoryType
```

The format of the **VALUE** clause includes these conditions:

- **transportLabel** is the transport identifier in ASCII text. Transports are grouped by transportLabel. Entries which share a transportLabel must have unique transport subindex values.
- **transportSubIndex** is an integer.
- **transportDomain** is an OID which indicates the network type (UDP/IP, IPX). RFC-1906 defines several possible values for this field. The dotted-decimal values and their English

equivalents are listed in the following table. The value of this OID can be specified in dotted-decimal format or by the English name.

English Names	Numeric OIDs
snmpUDPDomain	1.3.6.1.6.1.1
snmpCLNSDomain	1.3.6.1.6.1.2
snmpCONSDomain	1.3.6.1.6.1.3
transportssnmpDDPDomain	1.3.6.1.6.1.4
transportssnmpIPXDomain	1.3.6.1.6.1.5

- **transportstransportAddress** is an address in the transportDomain. For example, if the transportDomain is snmpUDPDomain, a valid address is 127.0.0.1:0. This address is used as the destination address for outgoing Trap messages. Also, if source-address-checking is enabled, this field is used in conjunction with the transportstransportReceiveMask to determine if an incoming request has arrived from an authorized address.
- **transportstransportReceiveMask** is a bit field mask for the transportstransportAddress and is displayed in the snmpd.cnf file in the same format as the transportAddress. For example, if transportstransportDomain is snmpUDPDomain, a valid mask is 255.255.255.0:0. This mask is used in conjunction with the transportAddress to determine if an incoming request has arrived from an authorized address. For example, if the transportAddress is 192.147.142.254:0 and the transportReceiveMask is 255.255.255.0:0, a request from any port on the subnet 192.147.142.xxx is not rejected on the basis of the source address.
- **transportstransportMMS** is an integer representing the maximum message size capable on this transport.
- **transportstransportMemoryType** is nonVolatile**,** permanent**,** or readOnly**.**

Sample Transport Configurations Entry

The following samples illustrate four transport configuration entries:

- An entry on an NMS console, where 192.147.15.6: is the NMS IP address

```
transportEntry Console snmpUDPDomain \
192.147.15.6:0 255.255.255.255:0 1500 \
nonVolatile
```

- An entry for a single host

```
transportEntry mynet1 snmpUDPDomain \
192.147.142.254:0 255.255.255.255:0 1500 \
nonVolatile
```

- An entry for a subnet

```
transportEntry mynet 1 snmpUDPDomain \
192.147.142.0:0 255.255.255.0:0 1500 \
nonVolatile
```

- Entries for multiple subnets

```
transportEntry mynetA 1 snmpUDPDomain \
192.147.142.0:0 255.255.255.0:0 1500 nonVolatile
transportEntry mynetA 2 snmpUDPDomain \
192.147.143.0:0 255.255.255.0:0 1500 nonVolatile
transportEntry mynetB 1 snmpUDPDomain \
192.147.144.0:0 255.255.255.0:0 1500 nonVolatile
```

Configuring Master Agent Performance Parameters

Performance parameters control the way the Master Agent behaves under certain conditions. To fine tune the Master Agent performance in a target environment, add performance parameters to the snmpd.cnf configuration file. The following table describe each configuration performance parameter.

Parameter	Description	Value
MAX_THREADS	The maximum number of threads to use to operate the Master Agent. Since a thread is created for each PDU, this value is equivalent to the maximum number of SNMP requests that can be processed asynchronously. The larger the value, the more requests that can be serviced at one time. However, a large value uses more memory. Subagents are not asynchronous and can only process one request at a time. For example, two **Get** requests called A and B are processed by the Master Agent. The objects in PDU A and PDU B are supported in a single Subagent called Xagt. Xagt has to process request A completely before processing request B.	The default is 10.

Parameter	Description	Value
	As a second example if request A is performed by Xagt and request B is performed by Subagent, Yagt, the process requests can be performed in parallel. Determine how your environment is configured and then set this value to optimize performance. ◆ If you use a few Subagents to contain many MIB variables, set this parameter to a small value. ◆ If you use a many Subagents to contain a few MIB variables, set this value to a large value.	
MAX_PDU_TIME	The maximum length of time the Master Agent waits for a PDU to be processed before the Subagent times out.	The default is 2500 centiseconds.
MAX_OUTPUT_WAITING	The maximum amount of serialized Master Agent communication data to buffer before the communication stream overflows. An overflow condition can occur in the following instances: ◆ The sender is writing data to the Master Agent stream faster than the receiver can read the data from stream. ◆ The receiving process is paused or halted by a system crash. ◆ A Master Agent message is larger than the buffer. This may happen if a Subagent attempts to register a large number of MIB objects in a single registration event. This scenario can be averted even with a smaller value for MAX_OUTPUT_WAITING if large MIBs are registered piecemeal in several events.	The default value is 65536 bytes.
MAX_SUBAGENTS	The maximum number of Subagents that can be connected to the Master Agent at one time. This value includes all types of Subagents including shared library Subagents.	The default value is 10.

Chapter 4

Connect:Direct for UNIX SNMP Master Agent

This chapter provides instructions for running the Master Agent. It also discusses how to change certain default options when running the Master Agent.

Preparing the Master Agent

Before starting the Master Agent, perform the following steps:

1. Type the following command to log in as the root user. The Agent must be started by the super user.

```
% su -
```

2. Type the following command to ensure that the SNMP ports are defined in the /etc/services file:

```
# grep snmp /etc/services
snmp 161/udp
snmp-trap 162/udp
```

3. If the snmpd.cnf configuration file has been installed in a location other than the default of /etc/srconf/agt, then set the SR_AGT_CONF_DIR environment variable to identify the location:

   For C shell:

```
# setenv SR_AGT_CONF_DIR /etc/srconf/agt
```

4. Issue a kill command for any existing SNMP agents.

Running the Master Agent

You can run the Master Agent as a daemon, foreground process, or background Process. To run the Master Agent, you must be logged on as root. If you are not logged on as root, type the following command to log in as root:

```
% su -
```

Running the Agent as a Daemon

To run the Agent as a system daemon, perform the following steps:

1. Type the following command to change to the installation directory:

```
cd <install directory>/ndm/snmp/bin
```

2. Type the name of the Master Agent followed by any desired command line arguments. Command line arguments are described in *Command Line Arguments* on page 34.

```
# snmpdm
```

3. Type the following command to verify that the Master Agent is running:

```
# ps -ef | grep snmp
root 24825 0.0 0.3 2296 ? S 00:07 0:00 snmpdm
```

If the Master Agent is not running, check for the file called snmpd.log (in the /tmp directory on most UNIX systems).

Running the Agent as a Foreground Process

To run the Master Agent as a foreground Process, perform the following steps:

1. Type the name of the Master Agent followed by -d and any other desired command line arguments.

```
# snmpdm  -d
```

Some informational messages may be displayed. If the Master Agent runs successfully, the command prompt does not return. While the Master Agent runs in the foreground, it receives signals, which are issued at the keyboard, such as SIGINT (an interrupt signals sent by typing **Ctrl-C**).

Note: Typing **Ctrl-C** issues a kill command for the Master Agent.

Running the Agent as a Background Process

To run the Master Agent as a background process, type the name of the Master Agent followed by -d and any other desired command line arguments as described in *Command Line Arguments* on page 34. Type an ampersand (&) at the end of the command to run the Master Agent in background mode.

```
# snmpdm -d &
```

At any time after the Master Agent successfully starts, informational messages may be displayed. If the Master Agent does not start successfully, a message indicating that the agent has exited is displayed.

Using Nonstandard Ports

RFC-1157 specifies that SNMP entities receive all messages on UDP port 161 and traps on UDP port 162. To use nonstandard ports, either identify this information in environment variables or define them in the etc/services file.

Defining Nonstandard Ports in Environment Variables

You can configure the Master Agent to communicate on other ports by changing the SR_SNMP_TEST_PORT and SR_TRAP_TEST_PORT settings. If desired, change one or more of the following environment variables to set or change which ports to use:

Variable	Description
SR_SNMP_TEST_PORT	The UDP port number to use for all SNMP communications other than traps. This environment variable also changes the way the Master Agent communicates with Subagents. For example, on most UNIX systems, this variable modifies the name of the UNIX Domain Socket that receives connection requests from remotely coupled (or loosely coupled) Subagents.
SR_TRAP_TEST_PORT	The UDP port number to use for all SNMP traps.

Defining Nonstandard Ports in the /etc/services File

The **/etc/services** file defines the network services provided by the local network host. This file defines the ports to use for most SNMP network traffic. The following sample file illustrates entries:

```
snmp 161/udp
snmp-trap 162/udp
```

If the appropriate environment variables are not defined, SNMP entities search this file to determine which ports to use for SNMP communication.

To select a nonstandard port for SNMP communication, change 161 in **/etc/services** to another number. The number selected is the port number used by the Master Agent. Specify a unique port number between 5000 and 9999 for all UDP entries.

Chapter 5

Connect:Direct for UNIX SNMP Subagent

The Connect:Direct SNMP Subagent is a proxy agent that enables a Connect:Direct server to provide information to SNMP network management stations. This chapter provides procedures on using the Connect:Direct SNMP Subagent, including command line arguments, configuration files, and the status of each variable with published data within the MIB.

Preparing the Connect:Direct for UNIX SNMP Subagent

Complete the following steps to prepare the Connect:Direct for UNIX SNMP Subagent for use:

1. Type the following command to change to the snmp directory:

```
cd <installation directory>/ndm/snmp/bin
```

2. Type the following command to change the ownership to root:

```
chown root ./connectdirectagt
```

3. Type the following command to change the mode to 4755:

```
chmod 4755 ./connectdirectagt
```

4. If you are using a network monitoring system, load the Connect:Direct SNMP Agent MIB, connectdirect.my, located in <*installation directory*>/ndm/snmp/bin directory.

Running the Subagent

To start the Connect:Direct SNMP Subagent, type the connectdirectagt command followed by any desired arguments, as described in the table in *Command Line Arguments* on page 34.

```
./connectdirectagt -cdinitfile initfilename -cdkeyfile keyfilename&
```

Place an ampersand (&) at the end of the command to run the Subagent in background mode.

Command Line Arguments

The Connect:Direct SNMP subagent accepts the following command line arguments:

Argument	Description	Default
-help	Prints out all command line arguments.	
-cddumpinit	Writes a sample initialization file to stdout.	
-cdinitfile *filename*:	Specifies the location and name of the initialization file used by the SNMP subagent. The initialization file contains variables of the agent group and sets default values for the agent. If the initialization file does not exist, the subagent uses the default values.	
-cdmaxcache N:	Determines the maximum Process list cache size in megabytes. If this value is not defined or set to zero, the subagent places no constraints on the amount of memory allowed for a agentCacheRequested message. The subagent grants memory up to the maximum cache size in response to an agentCacheRequested message. This value overrides the agentMaxCache.0 parameter in the initialization file.	
-cdmaxpacket N:	Specifies the maximum SNMP packet size.	400
-cdport N:	Specifies the port that the SNMP subagent monitors. **Note:** If you change the port number using -cdport N, you must change the port number in the Connect:Direct for UNIX initparm.cfg file. Refer to the description of the **snmp.agent.port** parameter in the *Connect:Direct for UNIX Administration Guide*.	1365
-cdretry N:	Defines how long to wait before validating a connection. The lower this value, the more frequently the agent rechecks the connection. Rechecking frequently increases processor time, which may slow down response time.	5
-cdkeyfile *keyfilename*	Specifies the location of the asset protection license key file.	

Signals

The Connect:Direct for UNIX Subagent understands the following signals:

- **SIGHUP** reloads the Subagent initialization file. You can specify this file with the Subagent cdinitfile command line argument. This signal also clears the open process statistics variable.
- **SIGUSR1** is reserved.
- **SIGUSR2** is reserved.
- **SIGTERM** shuts down the Subagent.

Retrieving a Value from a Variable

Use the getnext SNMP utility to retrieve the next value after the specified variable from an SNMP entity. If any of the variables in the list is a malformed OID, nothing is sent out and an error is displayed.

Setting the SR_MGR_CONF_DIR Variable

If the default location of the etc/srconf/mgr has been changed or if the mgr.cnf configuration file has been installed in a location other than the default of /etc/srconf/mgr, then set the SR_MGR_CONF_DIR environment variable to identify the location of the file:

For C shell:

```
# setenv SR_MGR_CONF_DIR /etc/srconf/mgr
```

For K shell:

```
# SR_MGR_CONF_DIR=/etc/srconf/mgr
# export SR_MGR_CONF_DIR
```

Sample getnext Requests

The following getnext command line displays all available parameters:

```
getnext -v[1|2] [-ctx v2ContextName]
[-ctxid v2ContextSnmpID] [-d] [-timeout secs] [-retries num]
agent_addr userName|community variable_name ...
```

The following sample **getnext** request returns the variables sysDescr.0 and snmpInPkts.0.

```
% getnext -v1 localhost public sysDescr snmp
```

```
% getnext -v2c localhost public sysDescr snmp
```

The following two calls return snmpOutPkts.0, the next variable after snmpInPkts.0.

```
% getnext -v1 localhost public snmpInPkts.0
```

```
% getnext -v2 localhost Guest snmpInPkts.0
```

Defining the Initialization File

The Connect:Direct SNMP Agent uses an initialization file to set default parameter values. You can override the initialization file parameters agentMaxCache and agentPortNo by typing the corresponding command line arguments. Valid lines within the initialization file can include any of the following options:

- Comment (a line that begins with a # mark)
- Blank line
- Valid parameter of the form variable.0 = value

The following figure displays a sample initialization file:

```
# VARIABLES VISIBLE TO AND WRITABLE BY SNMP:
agentCacheRequested.0 = 0
agentDaysOnHandRequested.0 = 7
sendAlarmTraps.0 = 2
sendStatusTraps.0 = 1
agentPSTimeOut.0 = 28800
agentProcessErrorThreshold.0 = 0
agentSessionFailTimeVal.0 = 3600
agentProcessFailTimeVal.0 = 3600
agentFmhReceivedTimeVal.0 = 3600
agentSessionFailThreshold.0 = 10
agentProcessFailThreshold.0 = 10
agentFmhReceivedThreshold.0 = 10
# VARIABLES NOT VISIBLE TO OR NOT WRITABLE BY SNMP:
agentMaxCache.0 = 0
agentPortNo.0 = 1365
agentKeyFile.0 = path to license keyfile
processManagerContactName.0=SterlingCommerceTechnicalSupport
processManagerPhone.0= (800)292-0104
processManagerDescription.0 = Connect:Direct SNMP Agent
processManagerCDNodeName.0 = SNMP_NODE
```

This initialization file contains default values. Omitting a parameter results in the parameter being initialized to its default value. The keyword writable is present if a variable is a read-write variable. The initialization parameters are described in the following table:

Variable	Description	Value
agentCacheRequested writable	The amount of Process data that the agent stores in memory. The agent grants the cache on the basis of system resources at the time of the request.	number of megabytes
agentCacheGranted	The amount of cache granted for retrieving data. The agent grants the cache by comparing the request from the agentCacheRequested, the initialization file parameter agentMaxCache.0, or to the cdmaxcache command line parameter. The cdmaxcache overrides the initialization file parameter. The agentMaxCache.0 is maintained on the local file system. If the agentCacheGranted megabytes of data contains records older than the agentDaysOnHandGranted and agentDaysOnHandGranted is not zero, the agent only stores agentDaysOnHandGranted days of data, up to a maximum cache size of agentCacheGranted megabytes. If agentCacheGranted and agentDaysOnHandGranted are set to 0, no completed Process cache will be kept.	a numeric value | 0
agentDaysOnHandRequested writable	The amount of data the agent caches in memory. The cache is granted based on system resources at the time of the request.	number of days of data

Variable	Description	Value
agentDaysOnHandGranted	The amount of cache granted for retrieving Process data. This number represents what the system provides by comparing the request from the agentDaysOnHandRequested, the initialization file parameter agentMaxCache.0, or to the cdmaxcache command line parameter. The command line parameter overrides the initialization file parameter. The agentMaxCache.0 parameter is maintained on the local file system. If agentDaysOnHandGranted days of data takes up more memory than **agentCacheGranted** megabytes and **agentCacheGranted** is not zero, the agent only stores agentCacheGranted megabytes of data, up to a maximum of agentDaysOnHandGranted days of data. If agentCacheGranted and agentDaysOnHandGranted are set to 0, no completed Process cache is kept.	7
trapMessage	The error or warning message contained in the most recent trap send by the Subagent.	error message
sendAlarmTraps writable	Enables or disables alarm trap generation. 1 disables it and 2 enables it.	1 | 2
sendStatusTraps writable	Enables or disables status trap generation. 1 disables the status trap generation and 2 enables it.	1 | 2
agentUnknownMsgRcv writable	The number of records received and unrecognized.	number of records
agentPSTimeOut writable	The timeout value. The agent uses this value to close a Process or session. This variable is used if a Process close record (PERR, PRED, PFLS) is not received. This value prevents the agent from holding the Process records indefinitely.	seconds | 28800 (8 hours)
agentProcessErrorThreshold writable	The largest Process completion code value that is still considered successful. Any Process returning with a completion code larger than this value increments processFailedprocesses by 1.	numeric value | 0

Variable	Description	Value
agentSessionFailTimeVal writable	The session failure rate. If agentSessionFailThreshold sessions fail in the past agentSessionFailTimeVal seconds, a trap is generated.	number of seconds | 3600
agentProcessFailTimeVal writable	The Process failure rate. If agentProcessErrorThreshold remote authentications fail in agentProcessFailTimeVal seconds, the SNMP Subagent generates a trap.	number of seconds | 3600
agentFmhReceivedTimeVal writable	The function management header failure rate. If agentFmhReceivedThreshold FMRV messages are received in agentFmhReceivedTimeVal seconds, the SNMP Subagent generates a trap.	number of seconds | 3600
agentSessionFailThreshold writable	The maximum number of session failures received in the time value before a trap is generated.	number of session failures | 10
agentProcessFailThreshold writable	The maximum number of Process failures received in the time value before a trap is generated.	number of Process failures | 10
agentFmhReceivedThreshold writable	The maximum number of function management header failures received in the time value before a trap is generated.	number of header failures | 10

Defining Default Process Manager Data

The initialization file parameters identify the default values for the processManager variables. This information, including contact name, phone, description, and node name is overridden when the Subagent receives an NINF record from Connect:Direct. These values are specified within the initialization file for administrative purposes only and for any situation in which an unattached SNMP Subagent exists, for example, when an SNMP Subagent has been started, but which has no Connect:Direct Process connected to it. The processManager variables are described in the following table:

Variable	Description	Value
processManagerUpTime	The time since the Process Manager started.	number of seconds
processManagerContactName	The contact name of the person who manages the Process Manager.	name
processManagerPhone	The phone number of the contact person.	phone number

Variable	Description	Value
processManagerDescription	The description from the Connect:Direct initialization parameters configuration file. Descriptions longer than 256 characters are truncated.	description up to 256 characters
processManagerCDNodeName	The Connect:Direct node from the Connect:Direct initialization parameters configuration file.	node name

Viewing Process Statistics

The following processStats variables control features of the Process statistics:

Variable	Description
processOpenProcesses	The number of Processes currently being executed. Process Managers handle Processes. Connect:Direct starts Session Managers to handle Processes in the Transmission Control Queue (TCQ) waiting to be executed. An open Process for this implementation is the agent having received a Process started record with no corresponding Process end record or indication that a Process error occurred.
processTotalProcesses	The total number of Processes handled since this agent has been collecting data. Agent resets or system resets restart this number and counter rollovers. The agent starts with the Process Manager, and the count is the total number of Processes since the start of the Process Manager.
processFailedProcesses	The total number of failed Processes detected since the agent has been collecting data. The value is reset by counter rollover, agent reset, or system resets. A failed Process is a Process with a received Process end record that has a completion code indicating that the Process did not end successfully.
processCompletedLastDay	The number of Processes completed in the last day. last day is from 12:00:00 a.m. yesterday to 11:59:59 p.m. yesterday. This type is a gauge so the number can go up or down.
processCompletedLastHour	The number of Processes completed in the last hour.
processHighProcesses	The highest number of open Processes per hour that has ever been achieved.
processTotalBytes	The total number of bytes transferred since the Process Manager has been running. Maximum value of this counter is (1024*1024)-1, which equals 1,048,575. This value reflects bytes and kilobytes only. To get the total number of bytes transmitted, you must add this value to processTotalMBytes, processTotalGBytes, and processTotalTBytes.

Variable	Description
processTotalMBytes	The total number of megabytes transferred since the Process Manager started running. This counter increments when the corresponding byte counter rolls over. A megabyte is 1024*1024 (or 1,048,576) bytes, not 1,000,000 bytes. This value reflects megabytes only. To get the total number of bytes transmitted, you must add this value to processTotalBytes, processTotalGBytes, and processTotalTBytes.
processTotalGBytes	The total number of gigabytes transferred and received since the Process Manager started running. This counter increments when the corresponding megabyte counter rolls over. A gigabyte is 1024 megabytes. This value reflects gigabytes only. To get the total number of bytes transmitted, this value must be added to processTotalBytes, processTotalGBytes, and processTotalTBytes.
processTotalTBytes	The total number of terabytes transferred and received since the Process Manager started running. This counter increments when the corresponding gigabyte counter rolls over. A terabyte is 1024 gigabytes. This value reflects terabytes only. To get the total number of bytes transmitted, this value must be added to processTotalBytes, processTotalMBytes, and processTotalGBytes.
processBytesLastDay	The number of bytes transmitted from 12:00:00 a.m. yesterday to 11:59:59 p.m. yesterday. This type is a gauge so the number can go up or down. This value reflects bytes and kilobytes only. To get the total number of bytes transmitted during the last day, you must add this value to processMBytesLastDay, processGBytesLastDay, and processTBytesLastDay.
processMBytesLastDay	The number of megabytes transmitted from 12:00:00 a.m. yesterday to 11:59:59 p.m. yesterday. This type is a gauge, so the number can go up or down. This value reflects megabytes only. To get the total number of bytes transmitted during the last day, this value must be added to processBytesLastDay, processGBytesLastDay, and processTBytesLastDay.
processGBytesLastDay	The number of gigabytes transmitted from 12:00:00 a.m. yesterday to 11:59:59 p.m. yesterday. This type is a gauge, so the number can go up or down. This value reflects gigabytes only. To get the total number of bytes transmitted during the last day, this value must be added to processBytesLastDay, processMBytesLastDay, and processTBytesLastDay.
processTBytesLastDay	The number of terabytes transmitted from 12:00:00 a.m. yesterday to 11:59:59 p.m. yesterday. This type is a gauge, so the number can go up or down. This value reflects terabytes only. To get the total number of bytes transmitted during the last day, this value must be added to processBytesLastDay, processMBytesLastDay, and processGBytesLastDay.

Variable	Description
processBytesLastHour	The number of bytes transmitted from the beginning of the last hour to the end of the last hour. For example, if the time is 10:13 a.m., this value would represent the bytes transmitted from 9:00:00 a.m. to 9:59:59 a.m. This value reflects bytes and kilobytes only. To get the total number of bytes transmitted during the last hour, this value must be added to processMBytesLastHour, processGBytesLastHour, and processTBytesLastHour.
processMBytesLastHour	The number of megabytes transmitted from the beginning of the last hour to the end of the last hour. For example, if the time is 10:13 a.m., this value represents the megabytes transmitted from 9:00:00 a.m. to 9:59:59 a.m. This value reflects megabytes only. To get the total number of bytes transmitted during the last hour, this value must be added to processBytesLastHour, processGBytesLastHour, and processTBytesLastHour.
processGBytesLastHour	The number of gigabytes transmitted from the beginning of the last hour to the end of the last hour. For example, if the time is 10:13 a.m., this value represents the gigabytes transmitted from 9:00:00 a.m. to 9:59:59 a.m. This value reflects gigabytes only. To get the total number of bytes transmitted during the last hour, this value must be added to processBytesLastHour, processMBytesLastHour, and processTBytesLastHour.
processTBytesLastHour	The number of terabytes transmitted from the beginning of the last hour to the end of the last hour. For example, if the time is 10:13 a.m., this value represents the terabytes transmitted from 9:00:00 a.m. to 9:59:59 a.m. This value reflects terabytes only. To get the total number of bytes transmitted during the last hour, this value must be added to processBytesLastHour, processMBytesLastHour, and processGBytesLastHour.

Viewing Session Statistics Variables

The following variables provide session statistics information:

Variable	Description
sessionOpenSessions	The number of sessions currently open. Session Managers are started by the Process Manager. An open session for this implementation is the agent receiving a session started record with no corresponding session end record.

Variable	Description
sessionTotalSessions	The number of sessions since this agent has been collecting data. Agent resets, system resets, and counter rollovers restart this number. The agent starts with the Process Manager, so the count is the number of total sessions since the Process Manager has been up and running.
sessionFailedSessions	The number of failed sessions that have been detected since the agent has been collecting data. The value is reset by counter rollover, agent reset, or system resets. A failed session is a session end record received with a completion code indicating that the session did not end successfully (SEND, SERR, or RNCF).
sessionCompletedLastDay	The number of sessions completed in the last day. **Last day** is from 12:00:00 am yesterday to 11:59:59 pm yesterday. This type is a gauge so the number can go up or down.
sessionCompletedLastHour	The number of sessions completed in the last hour. This parameter will reflect Processes that were completed in the last whole hour. Note that this type is a gauge so the number can go up or down.
sessionHighSessions	The highest number of open sessions per hour that has ever been reached.
highSessionDate	The date and time string in MM/DD/YY HH:MM:SS format when the sessionHighSessions value was achieved.
sessionYestPeakSessions	Yesterday's peak number of sessions.
sessionYestPeakHour	The hour in which yesterday's peak number of sessions was achieved.

Viewing Open Process Variables

The following variables provide information on open Processes:

Variable	Description
openProcessTable	Process entries for all currently running Processes.
openProcessEntry	A list of active Processes. Records in this list can be referenced using an index number and the record number. The index number is a unique number assigned to a Process by the SNMP agent. The record number starts at 1. One is always assigned to the PSTR record for that Process. The following sequence is a logical association of related data about a Process record. The variable is not access by an SNMP get command. It is filled in by requesting each component separately.

Variable	Description

openProcessEntry (*continued*)	For example, if a recently completed Process occurred with an index of 2 and performed a PSTR and a CTRC as its first two steps, the records could be accessed using the following commands:
	`unix> getone -v1 localhost public processIndex.2.1`
	`processIndex.2.1 = 2`
	`unix> getone -v1 localhost public processPNUM.2.1`
	`processPNUM.2.1 = 5`
	`unix> getone -v1 localhost public processPnode.2.1`
	`processPnode.2.1 = solaris_box`
	`unix> getone -v1 localhost public processSnode.2.1`
	`processSnode.2.1 = solaris_box`
	`unix> getone -v1 localhost public processNodetype.2.1`
	`processNodetype.2.1 = S`
	`unix> getone -v1 localhost public processRecNum.2.1`
	`processRecNum.2.1 = 1`
	`unix> getone -v1 localhost public processRaw.2.1`
	`processRaw.2.1 = PSTR | log time=935173364 | pname=sample | pnum=5 | start ti`
	`me=935173364 | stop time=935173364 | elapsed time=0 | this node=S | pnode=solari`
	`s_box | snode=solaris_box | ccode=0 | message id=XSMG200I | message text=Remote`
	`process started`
	
	`unix> getone -v1 localhost public processIndex.2.2`
	`processIndex.2.2 = 2`
	`unix> getone -v1 localhost public processPNUM.2.2`
	`processPNUM.2.2 = 5`
	`unix> getone -v1 localhost public processPnode.2.2`
	`processPnode.2.2 = solaris_box`
	`unix> getone -v1 localhost public processSnode.2.2`
	`processSnode.2.2 = solaris_box`
	`unix> getone -v1 localhost public processNodetype.2.2`
	`processNodetype.2.2 = S`
	`unix> getone -v1 localhost public processRecNum.2.2`
	`processRecNum.2.2 = 2`
	`unix> getone -v1 localhost public processRaw.2.2`
	`processRaw.2.2 = CTRC | log time=935173365 | pname=sample | pnum=5 | start_ti`
	`me=935173364 | stop_time=935173365 | elapsed time=1 | snode=solaris_box | pnode=`
	`solaris_box | loc_node_ind=S | comp_code=8 | src_byte_cnt=0 | src_byte_xmit_cnt=`
	`0 | src_rec_cnt=0 | dest_byte_cnt=0 | dest_byte_rcv_cnt=538968064 | dest_rec_cnt=78 | src_file=/home/cdunix33/ndm/bin/direct | dest_file=(empty) |`

processIndex	A unique index issued by the SNMP Subagent that identifies Process records.
processPNUM	An index by the Process MGR to identify Process records.
processPnode	A primary node name.
processSnode	A secondary node name.
processNodetype	The node type.

Variable	Description
processRecNum	A sequential number assigned by the SNMP Subagent for each record that has been received for this index number.
processRaw	The Process record received.

Viewing Information on Recent Processes

The following variables provide information on recent Processes:

Variable	Description
recentProcessesTable	Process entries for all Processes that are running.

recentProcessEntry	A list of the recently completed Processes. Records within this list can be referenced by using the index number and the record number of the desired record. The index number is a unique number assigned to a Process by the SNMP agent. The record number starts at 1, which will always be assigned to the PSTR record received for that Process. For each record received after the PSTR, the record number is increased by one and assigned to the receiving record.
	The following SEQUENCE is a logical association of related data about a process record. recentProcessEntry is never actually accessed by an SNMP GET command, but is filled in by requesting each component separately.
	For example, if there was a recent (completed) process with an index number of 2 and it had performed a PSTR and a CTRC as its first two steps, the records could be accessed using the following commands:
	`unix> getone -v1 localhost public recProcessIndex.2.1`
	`recProcessIndex.2.1 = 2`
	`unix> getone -v1 localhost public recProcessPNUM.2.1`
	`recProcessPNUM.2.1 = 5`
	`unix> getone -v1 localhost public recProcessPnode.2.1`
	`recProcessPnode.2.1 = solaris_box`
	`unix> getone -v1 localhost public recProcessSnode.2.1`
	`recProcessSnode.2.1 = solaris_box`
	`unix> getone -v1 localhost public recProcessNodetype.2.1`
	`recProcessNodetype.2.1 = S`
	`unix> getone -v1 localhost public recProcessRecNum.2.1`
	`recProcessRecNum.2.1 = 1`
	`unix> getone -v1 localhost public recProcessRaw.2.1`
	`recProcessRaw.2.1 = PSTR | log time=935173364 | pname=sample | pnum=5 | start time=935173364 | stop time=935173364 | elapsed time=0 | this node=S | pnode=solaris_box | snode=solaris_box | ccode=0 | message id=XSMG200I | message text=Remote process started`
	`unix> getone -v1 localhost public recProcessIndex.2.2`
	`recProcessIndex.2.2 = 2`

Variable	Description
recentProcessEntry (*continued*)	unix> getone -v1 localhost public recProcessPNUM.2.2 recProcessPNUM.2.2 = 5 unix> getone -v1 localhost public recProcessPnode.2.2 recProcessPnode.2.2 = solaris_box unix> getone -v1 localhost public recProcessSnode.2.2 recProcessSnode.2.2 = solaris_box unix> getone -v1 localhost public recProcessNodetype.2.2 recProcessNodetype.2.2 = S unix> getone -v1 localhost public recProcessRecNum.2.2 recProcessRecNum.2.2 = 2 unix> getone -v1 localhost public recProcessRaw.2.2 recProcessRaw.2.2 = CTRC | log time=935173365 | pname=sample | pnum=5 | start_time=935173364 | stop_time=935173365 | elapsed time=1 | snode=solaris_box | pnode= solaris_box | loc_node_ind=S | comp_code=8 | src_byte_cnt=0 | src_byte_xmit_cnt=0 | src_rec_cnt=0 | dest_byte_cnt=0 | dest_byte_rcv_cnt=538968064 | dest_rec_cnt=78 | src_file=/home/cdunix33/ndm/bin/direct | dest_file=(empty) |
recProcessIndex	A unique index issued by the SNMP agent to identify Process records.
recProcessPNUM	An index by the Process Manager to identify Process records.
recProcessPnode	The primary node name.
recProcessSnode	The secondary node name.
recProcessNodetype	The Process node type.
recProcessRecNum	A sequential number assigned by the SNMP Subagent for each record that has been received for this process number.
recProcessRaw	The Process record received.

Traps

Traps are asynchronous, unsolicited messages sent to the network management station when certain conditions are detected by the agent. Connect:Direct traps are defined in two groups: status traps and alarm traps.

- Status traps are not critical to the operation of Connect:Direct, but they show information that can be of interest to the operator.
- Alarm traps are more serious, such as the Process Manager is down or a license has expired.

statusTrap Variables

The statusTrap Group of traps can be enabled or disabled by setting the sendStatusTraps OID in the agent group. To identify where traps are sent, view the settings in the snmpd.cnf configuration file.

The statusTrap variables are described in the following table:

Trap	Description
cdStartUpTrap	Process Manager start up has been detected.
cdShutDownTrap	Shutdown of the Process Manager has been detected.
configFileChangeTrap	A change to the configuration file has been detected.
agentResetTrap	The Subagent has been reset by a HUP signal.

alarmTrap Group

The alarmTrap group of traps can be enabled or diabled by setting the sendAlarmTraps OID in the agent group. To identify where traps are sent, view the settings in the snmpd.cnf configuration file.

The following alarmTrap variables can occur during operation:

Trap	Description
agentShutdownTrap	The SNMP Subagent is shut down via the KILL signal.
agentCoreDumpTrap	The SNMP Subagent encounters a segmentation violation.
completionCodeTrap	A Process record was received with a completion code other than zero. The agent puts the ccod and the message code in the trap message.
fmhReceivedTrap	The number of function management header failures detected is beyond the threshold set in agentFmhFailThreshold.
licensingError	Connect:Direct encountered a licensing error.
licenseExpiredTrap	The license for Connect:Direct has expired.
licenseExpiresTrap	The license for Connect:Direct will expire in 14 days.
pgmrNotRunningTrap	The agent has detected that the PMGR is not running.
processesFailedTrap	The number of process failures detected is beyond the threshold set in agentProcessFailThreshold.
securePlusConfigurationError	Connect:Direct encountered a Secure+ Option configuration error.
securePlusOperationError	Connect:Direct encountered a Secure+ Option operation error.
securePlusSecurityViolation	Connect:Direct encountered a Secure+ Option security violation.
sessionsFailedTrap	The number of sessions that failed has exceeded the value of agentSessionFailureThreshold for the interval specified in agentSessionFailTimeVal.

Appendix A

Troubleshooting

This appendix describes some of the problems that can occur when running the agent. For each problem, a description of the symptom is listed, including suggested steps for correction.

When I try to run the program, the message *Permission denied* is displayed.

On UNIX systems, this problem can happen to a privileged user if the mode of the executable image is set incorrectly. First, check the mode with **ls**:

```
# ls -lg snmpdm
-rw------- 1 root daemon 1966080 Apr 30 11:02 snmpdm
```

If the permission bits do not contain the execute privilege (**rwx**), change the permissions with the chmod command as illustrated in the following command line:

```
# chmod 700 snmpdm
# ls -lg snmpdm
-rwx------ 1 root daemon 196680 Apr 30 11:02 snmpdm
```

If you are an unauthorized user, contact the local system administrator to apply for access.

After I start the SNMP Agent, nothing about it seems to work.

On UNIX systems, the agent displays a banner message at start up and then activates the daemon to become a background process.

```
% snmpdm
SNMP Research EMANATE Agent Version 16.1.0.10
Copyright 1989-2004 NMP Research, Inc.
```

After startup, no direct interaction occurs with the user at the console or terminal. When the agent does not appear to be working, the program could have encountered a problem and quit.

To resolve this issue, first check to ensure the program is running. Type the **ps** command, as illustrated in the following sample. In this sample, no SNMP agent program is running.

```
% ps -ef | grep snmp
joeuser 28091 0.0 0.3 32 196 qe S 21:04 0:00 grep snmp
%
```

To determine why the agent is not running, check the snmpd.log file.

The *SNMP Port Already in Use Message* Is Displayed

If the following message is displayed, the SNMP port is already in use and the agent could not continue:

```
# cat snmpd.log
AgentSocketCreate: bind
       at line 479 in file tcp.c
Address already in use
```

To fix this problem, either shut down the SNMP agent or use a different UDP port.

Agent Does Not Have Necessary Privileges

If the following message is displayed, the agent does not have the privilege to bind to the SNMP port:

```
% AgentSocketCreate: bind: Permission denied
       at line 398 in file uds.c
```

This problem can happen to a nonprivileged user if the mode of the executable image is not set up correctly. First, check the mode with **ls**, as illustrated in the following example:

```
% ls -lg snmpdm
-r-s------ 1 root daemon 1966080 Apri 30 11:02 snmpdm
```

If the permission bits do not contain an **s** as shown in the preceding sample, only a privileged user can execute the program. To allow a nonprivileged user to start the agent, the system administrator can change the permissions with the **chmod** command:

```
# chmod 4500 snmpdm
```

The SNMP Agent Runs But Is Experiencing Unexpected Behavior

If you are experiencing unexpected agent behavior, your configuration could be set up incorrectly. Check the trace-level log messages to identify the cause of unexpected agent behavior.

Run the agent using the -apall option.

The following message is displayed and identifies where the agent looks for configuration files. If the directory where the agent is looking is not where the configuration files are located, an environment variable could be incorrectly set.

```
init_fnames: searching for configuration files in /tmp/srconf/agt from
getenv(SR_AGT_CONF_DIR)
```

The following message indicates that the agent does not understand a string in the snmpd.cnf configuration file while parsing a viewTreeEntry tag. The agent performs many name-to-OID translations—it only has a small list of well-known strings. If the English form of the OBJECT IDENTIFIER is not in the list, then the agent skips the configuration entry.

To resolve this problem, use the numeric representation of the **OBJECT IDENTIFIER** in the configuration file.

```
Reading config recordtype :viewTreeEntry
     at line 818 in file scanfile.c
MakeOIDFragFromDot, hash table lookup failed: srExamples
     at line 323 in file oidtran.c
MakeOIDFromDot: MakeOIDFragFromDot(srExamples) failed
     at line 441 in file oidtran.c
Can't make 'srExamples' into an OID
     at line 404 in file scanfile.c
ProcessConfigRecord: Error, cannot parse token srExamples
     at line 736 in file scanfile.c
```

Glossary

A

Agent

The SNMP process running on the server and serving variables to the client Network Management System.

Applications Programmer Interface

A set of calling conventions that define how a program or programmer calls a service.

C

Community

An administrative relationship between SNMP entities.

Community Name

An opaque string of octets that identify a community.

Connect Control Center

A centralized management system that provides operations personnel with continuous enterprise-wide business activity monitoring capabilities for Connect:Direct for z/OS, UNIX, and Windows servers. It manages multiple Connect:Direct servers to suspend, release, and delete Processes, stops Connect:Direct servers, and views detailed statistics on running or completed Processes. It monitors service levels to view Connect:Direct processing across Connect:Direct for z/OS, UNIX, and Windows servers within your network and retrieves information about active and completed Processes. It receives notification of data delivery events that occur or do not occur as scheduled and defines rules that, based on processing criteria, can generate an alert, send an e-mail notification, generate a Simple Network Management Protocol (SNMP) trap to an Enterprise Management System (ESM), or run a system command. It monitors for alerts, such as a server failure or a Process not starting on time.

Connect:Direct Browser User Interface

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

Connect:Direct SNMP Agent

An agent that provides remote monitoring of Connect:Direct for UNIX SNMP Agent events. The data used for monitoring is similar to the information available through the select statistics and select process commands.

D

Datagram

A self-contained unit of data transmitted independently of other datagrams.

Default Route

An entry in the routing table that will be used if no route is appropriate when you are sending an IP datagram.

E

External Data Representation

A transfer syntax defined by Sun Microsystems, Inc.

F

File Agent

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

Frame

A packet transmitted across media.

H

Host

The end-system.

Host-Identifier

The part of the IP address that corresponds to the host on the IP network.

Host-Number

The part of a subnetworked IP address that corresponds to the host-number on the subnet.

I

Internet

A collection of connected networks, primarily in the United States, running the Internet protocols.

Internet-Standard MIB

RFC1156.

L

Local Area Network

A technology that provides high speed and short delay with limited geographical size.

M

Management Information Base

A collection of objects that can be accessed through a network management protocol. It is abbreviated to MIB.

Management Station

The system that manages the network. This system is also called the network management station.

Master Agent

The agent that directs retrieval and write processing, and performs most of the trap processing. The Master Agent also handles the enrollment and de-enrollment of Subagents when they connect or disconnect and determines which Subagents will receive the request.

MIB

The abbreviation for Management Information Base.

MIB-I

The Internet-standard MIB (1156).

MIB-II

Currently RFC 1158 and RFCs.

MIB Objects

The objects that report statistics about TCP/IP network operations that take place on a particular system. The RFC1213 defines a set of MIB objects. The information provided by these objects is useful to network managers and is typically available on each and every managed node. The objects are organized into groups for logical divisions of network operations.

MIB-II System Group

A group that identifies the system or entity where the SNMP agent is running. The default value for each system group variable is defined by adding an entry to the snmpd.cnf configuration file.

MIB View

A collection of managed objects realized by an agent that is visible to a community.

N

National Institute of Standards and Technology

The institution that tracks standardization. This institute, previously known as the National Bureau of Standards, is a branch of the U.S. Department of Commerce.

Network Management Station (NMS)

The system that manages the network. This system is also called the management station.

P

Ping

A program that tests IP-level connectivity from one IP address to another.

R

Request for Comments

Describes the Internet suite of protocols and related experiments.

RFC

See Request for Comments.

Router

Software or hardware that directs packets through a network.

S

Simple Network Management Protocol

The application protocol offering network management in the Internet suite of protocols.

SNMP

See Simple Network Management Protocol.

snmpd.cnf

The configuration file that contains configuration information for security access rights, default MIB-II values and Master Agent performance parameters. The Master Agent uses this configuration file called `snmpd.cnf`.

Subagent

The agent that collects or calculates the values for the MIB variable and passes that information to the Master Agent. The Master Agent then sends the information to the requesting manager. Messages pass asynchronously between the Master Agent and the Subagent. Subagents can be started in any order.

T

TCP

See Transmission Control Protocol.

Transmission Control Protocol

A transport protocol that offers a connection-oriented transport service.

Trap

An asynchronous, unsolicited message sent to the Network Management Station when certain conditions are detected by the agent.

U

UDP

See User Datagram Protocol.

User Datagram Protocol

A protocol that offers a connectionless mode transport service in the Internet suite of protocols.

X

XDR

See External Data Representation.

Index

A

B

C

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V