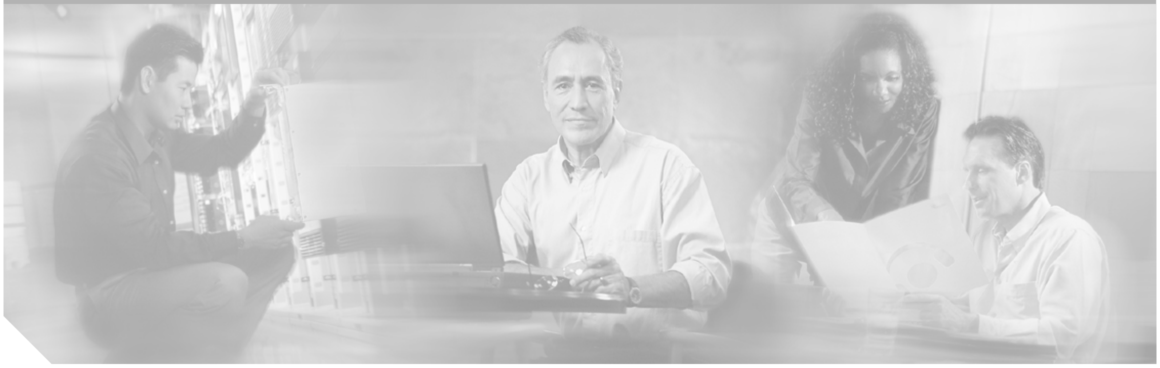

Network Connectivity Monitor System Administration Guide

Cisco Network Connectivity Center

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Text Part Number: OL-6301-01

Contents

Preface

The purpose of this guide is to provide information about administering Cisco Network Connectivity Center (CNCC) Network Connectivity Monitor (NCM) software. This includes information such as using the proper method to edit NCM files, locating NCM files, setting up security, managing log files, installing software licenses, and starting and stopping NCM programs.

Note: This document provides information pertaining to the NCM software platform. Such information is applicable to all NCM products that are based on this platform.

Intended Audience

This guide is intended to be read by system administrators or any user who needs to administer or maintain NCM software.

Prerequisites

This document assumes that the reader has installed NCM software. The reader will also require administrative privileges for both the NCM software and the systems on which the software is running.

Document Organization

This guide consists of the following sections:

1. INTRODUCTION	Provides an introduction to NCM system administration and an overview of tasks you may need to perform when administering NCM software.
2. LOCATING AND MODIFYING FILES	Describes the directory structure of NCM software and the recommended method for editing user modifiable files.
3. LICENSING NCM SOFTWARE	Describes how to obtain and install both evaluation and permanent licenses for NCM software.
4. CONTROLLING THE STARTUP OF NCM SOFTWARE	Describes how to start and stop programs running as services and as manual processes. It also explains how to configure a manual process to run as a service and how to modify a service's default parameters.
5. SECURING ACCESS TO NCM	Describes the security feature for NCM. It also provides instructions and examples for securing access to NCM software, and details the use of encrypted connections.
6. OPERATION OF THE NCM BROKER	Describes the purpose of the NCM Broker and explains how to view the Broker registry.
7. MANAGING LOG FILES	Describes how to manage log files.
A. ENVIRONMENT VARIABLES USED BY NCM SOFTWARE	Describes the environment variables used by NCM software.
B. WILDCARDS USED BY NCM SOFTWARE	Describes the wildcard patterns used by NCM software.

Table 1: **Document Organization**

Documentation Conventions

Several conventions may be used in this document as shown in Table 2.

CONVENTION	EXPLANATION
`sample code`	Indicates code fragments and examples in Courier font
keyword	Indicates commands, keywords, literals, and operators in bold
`%`	Indicates C shell prompt
#	Indicates C shell superuser prompt
<parameter>	Indicates a user-supplied value or a list of non-terminal items in angle brackets
[option]	Indicates optional terms in brackets
/InCharge	Indicates directory path names in italics
yourDomain	Indicates a user-specific or user-supplied value in bold, italics
File > Open	Indicates a menu path in italics

Table 2: **Documentation Conventions**

Directory path names are shown with forward slashes (/). Users of the Windows operating systems should substitute back slashes (\) for forward slashes.

Also, if there are figures illustrating consoles in this document, they represent the consoles as they appear in Windows. Under UNIX, the consoles appear with slight differences. For example, in views that display items in a tree hierarchy such as the Topology Browser, a plus sign displays for Windows and an open circle displays for UNIX.

Finally, unless otherwise specified, the term CNCC Manager is used to refer to NCM programs such as Domain Managers, Global Managers, and adapters.

Additional Resources

In addition to this manual, Cisco provides the following resources.

Commands

Descriptions of commands are available as HTML pages. The *index.html* file, which provides an index to the various commands, is located in the ***BASEDIR****/smarts/doc/html/usage* directory.

Documentation

Readers of this manual may find other documentation (also available in the ***BASEDIR****/smarts/doc/pdf* directory) helpful.

Network Connectivity Monitor Documentation

The following documents are product independent and thus relevant to users of all Network Connectivity Monitor products:

- *Release Notes for Network Connectivity Monitor 1.1*
- *Network Connectivity Monitor Documentation Roadmap*
- *Network Connectivity Monitor System Administration Guide*
- *ICIM Reference*
- *InCharge ASL Reference Guide*
- *Cisco Network Connectivity Center Perl Reference Guide*

Network Connectivity Monitor Service Assurance Management Documentation

The following documents are relevant to users of the NCM Service Assurance Management product suite.

- *Network Connectivity Monitor Service Assurance Management Suite Installation Guide*
- *An Introduction to Network Connectivity Monitor Service Assurance Manager*
- *Network Connectivity Monitor Operator's Guide*
- *Network Connectivity Monitor Service Assurance Manager Configuration Guide*
- *InCharge Service Assurance Manager Business Dashboard Configuration Guide*
- *InCharge Service Assurance Manager User's Guide for Business Impact Manager*

- *InCharge Service Assurance Manager User's Guide for Report Manager*
- *InCharge Service Assurance Manager Failover System User's Guide*

The following documents are relevant to NCM Service Assurance Manager adapters.

- *Network Connectivity Monitor Service Assurance Manager Notification Adapters User's Guide*
- *InCharge Service Assurance Manager SQL Data Interface Adapter User's Guide*
- *Network Connectivity Monitor Service Assurance Manager Adapter Platform User's Guide*
- *InCharge XML Adapter User's Guide*
- *InCharge Service Assurance Manager User's Guide for Remedy Adapter*
- *InCharge Service Assurance Manager User's Guide for Concord eHealth Adapter*
- *InCharge Service Assurance Manager User's Guide for InfoVista Adapter*

InCharge Application Services Manager Documentation

The following documents are relevant to users of InCharge Application Service Manager.

- *InCharge Application Management Suite Installation Guide*
- *InCharge Application Services Manager User's Guide*
- *InCharge Application Services Manager Discovery Guide*
- *InCharge Application Connectivity Monitor User's Guide*

Network Connectivity Monitor IP Management Documentation

The following documents are relevant to users of CNCC NCM IP Management Suite products.

- *Network Connectivity Monitor IP Management Suite Installation Guide*
- *Network Connectivity Monitor IP Deployment Guide*
- *Network Connectivity Monitor IP Discovery Guide*
- *Network Connectivity Monitor IP Availability Manager User's Guide*

- *InCharge IP Performance Manager User's Guide*
- *InCharge IP Adapters User's Guide*

Common Abbreviations and Acronyms

The following lists common abbreviations and acronyms that are used in the InCharge guides.

ASL	Adapter Scripting Language
CDP	Cisco Discovery Protocol
ICIM	InCharge Common Information Model
ICMP	Internet Control Message Protocol
IP	Internet Protocol
MSFC	Multilayer Switch Feature Card
MIB	Management Information Base
MODEL	Managed Object Definition Language
RSFC	Router Switch Feature Card
RSM	Router Switch Module
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
VLAN	Virtual Local Area Network

Obtaining Documentation

Cisco documentation and additional literature are available on Cisco.com. Cisco also provides several ways to obtain technical assistance and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

Cisco.com

You can access the most current Cisco documentation at this URL:

http://www.cisco.com/univercd/home/home.htm

You can access the Cisco website at this URL:

http://www.cisco.com

You can access international Cisco websites at this URL:

http://www.cisco.com/public/countries_languages.shtml

Ordering Documentation

You can find instructions for ordering documentation at this URL:

http://www.cisco.com/univercd/cc/td/doc/es_inpck/pdi.htm

You can order Cisco documentation in these ways:

- Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Ordering tool:

  http://www.cisco.com/en/US/partner/ordering/index.shtml

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

You can send comments about technical documentation to bug-doc@cisco.com.

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

For all customers, partners, resellers, and distributors who hold valid Cisco service contracts, Cisco Technical Support provides 24-hour-a-day, award-winning technical assistance. The Cisco Technical Support Website on Cisco.com features extensive online support resources. In addition, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not hold a valid Cisco service contract, contact your reseller.

Cisco Technical Support Website

The Cisco Technical Support Website provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The website is available 24 hours a day, 365 days a year at this URL:

http://www.cisco.com/techsupport

Access to all tools on the Cisco Technical Support Website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register at this URL:

http://tools.cisco.com/RPF/register/register.do

Submitting a Service Request

Using the online TAC Service Request Tool is the fastest way to open S3 and S4 service requests. (S3 and S4 service requests are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Service Request Tool automatically provides recommended solutions. If your issue is not resolved using the recommended resources, your service request will be assigned to a Cisco TAC engineer. The TAC Service Request Tool is located at this URL:

http://www.cisco.com/techsupport/servicerequest

For S1 or S2 service requests or if you do not have Internet access, contact the Cisco TAC by telephone. (S1 or S2 service requests are those in which your production network is down or severely degraded.) Cisco TAC engineers are assigned immediately to S1 and S2 service requests to help keep your business operations running smoothly.

To open a service request by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)
EMEA: +32 2 704 55 55
USA: 1 800 553 2447

For a complete list of Cisco TAC contacts, go to this URL:

http://www.cisco.com/techsupport/contacts

Definitions of Service Request Severity

To ensure that all service requests are reported in a standard format, Cisco has established severity definitions.

Severity 1 (S1)—Your network is “down,” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Severity 3 (S3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Severity 4 (S4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- Cisco Marketplace provides a variety of Cisco books, reference guides, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

  http://www.cisco.com/go/marketplace/

- The Cisco *Product Catalog* describes the networking products offered by Cisco Systems, as well as ordering and customer support services. Access the Cisco Product Catalog at this URL:

  http://cisco.com/univercd/cc/td/doc/pcat/

- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:

  http://www.ciscopress.com
- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet
- *iQ Magazine* is the quarterly publication from Cisco Systems designed to help growing companies learn how they can use technology to increase revenue, streamline their business, and expand services. The publication identifies the challenges facing these companies and the technologies to help solve them, using real-world case studies and business strategies to help readers make sound technology investment decisions. You can access iQ Magazine at this URL:

  http://www.cisco.com/go/iqmagazine
- *Internet Protocol Journal* is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:

  http://www.cisco.com/ipj
- World-class networking training is available from Cisco. You can view current offerings at this URL:

  http://www.cisco.com/en/US/learning/index.html

1

Introduction

As a network administrator, you should configure the Cisco Network Connectivity Center (CNCC) Network Connectivity Monitor (NCM) software to meet the needs of your operations. To accomplish this, you may need to perform one or more of the following tasks.

- Modify NCM files
- License NCM software
- Start and stop NCM programs when applicable
- Set up NCM programs as services or manual processes
- Set up and maintain the security of NCM software
- Understand the operation of the NCM Broker
- Manage log files

The following sections summarize the contents of this guide, and direct you to specific chapters that include the information you need to perform the various tasks.

Locating and Modifying Files

Some files can be modified to meet the needs of your network environment. *Locating and Modifying Files* on page 5 describes the directory structure of the software. It also discusses the files that can be modified and how to locate the files. It then provides procedures for modifying those files. Additionally, the chapter describes the files created by NCM software (for example, logs, repositories, and saved consoles), and details how to control their location.

Licensing NCM Software

NCM software requires a license to function. To evaluate the software, you can obtain a time-limited evaluation license that provides full access to all features. After you purchase the software, you will need to install a permanent license. *Licensing NCM Software* on page 19 provides procedures for obtaining both evaluation and permanent licenses for your software. Topics include installing the license, starting the FLEXlm license server, and pointing the software to the license server. It also discusses licensing additional NCM programs and adding licenses.

Controlling the Start-up of NCM Programs

NCM programs can either be installed as services or as manual processes. *Controlling the Startup of NCM Software* on page 33 provides procedures using the *sm_service* utility to install programs as services, and to start and stop services and processes with the utility. It also explains how to modify the run-time parameters of services with the utility.

Securing Access to NCM

Security is a critical concern in the world of large-scale distributed networks. Therefore, NCM provides a means by which network administrators can set up security to control access to the software. *Securing Access to NCM* on page 55 describes how security works, as well as the authentication processes used by the software. It then provides procedures for setting up security and controlling access to NCM programs. It also discusses how to set up encrypted connections between NCM clients and servers.

Operation of the NCM Broker

NCM client programs, such as a console or an adapter, use the NCM Broker to determine where CNCC Managers are running. When a CNCC Manager starts, it registers the host name of the machine it is running on, as well as the TCP port it is listening to, with the broker. Thereafter, NCM clients retrieve this information from the broker so that they can communicate with the manager. *Operation of the NCM Broker* on page 85 details the functions of the NCM Broker, provides procedures for viewing the broker's registry, and explains how to change the broker's environment variables.

Managing Log Files

Log files grow indefinitely, though slowly, under normal conditions, and need to be managed. In addition, NCM software can maintain up to 1000 backup log files. You can configure the number of log files retained by NCM software. *Managing Log Files* on page 91 discusses the management of log files, and details how to "roll over" the files.

2

Locating and Modifying Files

Product Suites and Directories

NCM software is distributed as product suites. The suites include the:

- CNCC NCM IP Management Suite
- CNCC NCM Service Assurance Management Suite
- InCharge Application Management Suite
- InCharge Security Infrastructure Management Suite
- InCharge Software Development Kit

During the installation process, each suite of products is installed to a separate directory under the root installation directory (users can specify the root directory but not the root subdirectories). In this document, the term ***BASEDIR*** represents the location where software is installed.

- For UNIX, this location is: */opt/InCharge<n>/<productsuite>*.
- For Windows, this location is: *C:\InCharge<n>\<productsuite>*.

The <*n*> represents the software platform version number. The <*productsuite*> represents the product suite that the product is part of.

Table 3 defines the <*productsuite*> directory for each product.

PRODUCT SUITE	INCLUDES THESE PRODUCTS	DIRECTORY
CNCC NCM IP Management Suite	• IP Availability Manager • IP Performance Manager • IP Discovery Manager • CNCC NCM Adapter for HP OpenView NNM • CNCC NCM Adapter for IBM/Tivoli NetView • CNCC NCM Adapter for CiscoWorks LMS and ITEM	/IP
CNCC NCM Service Assurance Management Suite	• Service Assurance Manager • Global Console • Business Dashboard • Business Impact Manager • Report Manager • SAM Failover System • Notification Adapters • Adapter Platform • SQL Data Interface Adapter • SNMP Trap Adapter • Syslog Adapter • XML Adapter • Adapter for Remedy • Adapter for TIBCO Rendezvous • Adapter for Concord eHealth • Adapter for InfoVista • Adapter for NetIQ AppManager	/SAM
InCharge Application Management Suite	• Application Services Manager • Beacon for WebSphere • Application Connectivity Monitor	/APP
InCharge Security Infrastructure Management Suite	• Security Infrastructure Manager • Firewall Performance Manager • InCharge Adapter for Check Point/Nokia • InCharge Adapter for Cisco Security	/SIM
InCharge Software Development Kit	• Software Development Kit	/SDK

Table 3: **Product Suite Directory for NCM Products**

In accordance with the preceding table, version 1.1 of CNCC NCM IP Availability Manager for UNIX operating systems is, by default, installed to */opt/InCharge6/IP/smarts*. This location is referred to as ***BASEDIR/****smarts*.

Additionally, version 1.1 of CNCC NCM Service Assurance Manager is, by default, installed to */opt/InCharge6/SAM/smarts*. This location is also referred to as ***BASEDIR/****smarts*.

Note: ***BASEDIR/****smarts* can refer to different product suite directories. Whenever you need to find or modify a file, always check the full path to the file to ensure that you are in the correct directory.

Optionally, you can specify the root of ***BASEDIR*** to be something other than */opt/InCharge6* (on UNIX) or *C:\InCharge6* (on Windows), but you cannot change the <*productsuite*> location under the root directory.

Common Subdirectories

Table 4 lists the subdirectories that are common to all product suites and provides a brief description of their contents.

INSTALLATION PATH	SUBDIRECTORY	DESCRIPTION
BASEDIR/*smarts*	*/bin*	Software executables for the installed product suite. The files vary by suite.
	/classes	Console related files such as .jar that are required by the software.
	/conf	Product suite configuration files. The files vary by suite.
	/doc	Documentation for software and third-party software such as FLEXlm.
	/jre	Java Runtime Environment files shipped with software and used by the NCM consoles.
	/lib	Shared libraries required by the software.
	/local	Top level directory where local copies of NCM configuration, rules, and script files are created and stored. Also the default location where NCM writes files such as repository and log files.
	/perl	Files used for the PERL API.
	/rules	ASL rule sets shipped with the software. The files vary by suite.
	/script	Scripts, typically shell scripts on UNIX and CMD scripts on Windows. These include the license installation scripts.
	/setup	Setup files used by the installation process.

Table 4: **Common Subdirectories**

In addition to the common subdirectories, the Service Assurance Management Suite (*/SAM*) includes two additional subdirectories: */actions* (client and server) and */consoles*.

Product Specific Subdirectories

When files are specific to a particular product or product module within a product suite, they are installed to subdirectories where the naming usually reflects the product or product module name.

For example, configuration files relevant to the CNCC NCM Adapter for IBM/Tivoli NetView and CNCC NCM Adapter for HP OpenView NNM are located at ***BASEDIR****/smarts/conf/NV* and ***BASEDIR****/smarts/conf/OV.* The rules files for the adapters are located at ***BASEDIR****/smarts/rules/NV* and ***BASEDIR****/smarts/rules/OV.*

Likewise, configuration files relevant to the SAM Adapter Platform are located at ***BASEDIR****/smarts/conf/icoi*. The rules files for the Adapter Platform are located at ***BASEDIR****/smarts/rules/icoi*.

The names used for subdirectories containing files relevant to a particular product or product module are defined in Table 5.

DIRECTORY NAMING CONVENTION	DESCRIPTION
/action	Service Assurance Management Suite. Files relevant to server and client tools.
/app-poller	Application Management Suite. Files relevant to the CNCC Application Connectivity Monitor.
/console	Service Assurance Management Suite. Files relevant to the CNCC NCM Global Console.
/discovery	IP Management Suite. Files relevant to the NCM IP discovery process.
/eHealth	Service Assurance Management Suite. Files relevant to the CNCC Adapter for Concord eHealth.
/icf	IP Management Suite. Files relevant to the IP Availability and Performance Managers.
/icoi	Service Assurance Management Suite. Files relevant to the Service Assurance Manager Adapter Platform.
/ics	Service Assurance Management Suite. Files relevant to the Service Assurance Manager.
/include	Software Development Kit. Contains files used by the SDK.

Directory Naming Convention	Description
/infovista	Service Assurance Management Suite. Files relevant to the CNCC Adapter for Infovista.
/jakarta-tomcat-5.0.16	Service Assurance Management Suite. Files relevant to the CNCC Business Dashboard.
/maps	Service Assurance Management Suite. Files relevant to custom map icons.
/notifier	Service Assurance Management Suite. Files relevant to the Service Assurance Manager Notification Adapters.
/NV	IP Management Suite. Files relevant to the CNCC NCM Adapter for IBM NV.
/OV	IP Management Suite. Files relevant to the CNCC NCM Adapter for HP OV.
/remedy	Service Assurance Management Suite. Files relevant to the CNCC Adapter for Remedy.
/sdi	Service Assurance Management Suite. Files relevant to the CNCC SQL Data Interface.
/sql	Service Assurance Management Suite. Files relevant to the CNCC SQL Data Interface.
/trapd	IP and Service Assurance Management Suites. Files used by the CNCC NCM Trap Forwarder to receive, process, and forward SNMP traps.
/xml-if	Service Assurance Management Suite. Files relevant to the CNCC XML Adapter

Table 5: **Product/Module Specific Subdirectories**

The subdirectories listed in Table 5 generally reside under:

- ***BASEDIR****/smarts/conf* and ***BASEDIR****/smarts/local/conf*
- ***BASEDIR****/smarts/rules* and ***BASEDIR****/smarts/local/rules*

Note: Product independent files also reside in the ***BASEDIR****/smarts/conf* directory. Examples of product independent files include the *brokerConnect.conf*, *clientConnect.conf*, and the *serverConnect.conf* files used for product security authentication.

User Modifiable Files

The installation includes system files, configuration files, ASL rule set files, scripts, templates, as well as third-party software files that the software uses.

You should never alter system files, such as executables, shared libraries, MODEL files, and setup files. Nor should you alter third-party software system files provided with the Java Runtime Environment.

As part of the configuration process, you will need to make changes to some of the user modifiable files. The following are examples of user modifiable files:

- Configuration files such as *ics.conf*, *clientConnect.conf*, and *serverConnect.conf* (you are required to edit these)
- ASL rule sets such as *syslog_mgr.asl*
- Sample actions such as *ics-ping* and *ics-telnet*
- Template files used to import data such as *service.data.template* and *topology-group.data.template*

Original versions of configuration files, ASL rule sets, scripts, and template files are installed to the ***BASEDIR****/smarts/conf,* ***BASEDIR****/smarts/rules,* and ***BASEDIR****/smarts/script* directories. When you need to alter a file in one of these directories, make those changes to a *local* copy of the file. By default, local copies of user modifiable files should reside in ***BASEDIR****/smarts/local* or one of its subdirectories.

To edit a file, invoke the *sm_edit* utility from ***BASEDIR****/smarts/bin* directory. When used, *sm_edit* automatically opens a local copy of the specified file and saves the modified file to its appropriate location. For more information about how to use the *sm_edit* utility, refer to *Modifying Files* on page 11.

You should *only* modify local copies of user modifiable files and should *always* retain, unedited, the original version of these files.

Table 6 identifies the default subdirectories that include user-modifiable files found under ***BASEDIR****/smarts/local*. The subdirectories vary by product suite.

INSTALLATION PATH	DIRECTORY	DESCRIPTION
BASEDIR*/smarts/local*	*/actions*	Tool scripts
	/conf	Configuration files
	/data	Import files
	/rules	ASL rules files

Table 6: **Subdirectories for User Modifiable Files**

The ***BASEDIR****/smarts/local*/conf directory (for each suite) also contains the following file, which is created during installation:

- *runcmd_env.sh*, a file for setting environment variables for an installed product suite, including the default location for the NCM Broker. You can edit this file to change the default location of the broker or to define any suite-wide environment variables. For information about setting environment variables, see *Methods for Setting Environment Variables for NCM Software* on page 97.

The product-specific naming convention used for subdirectories under ***BASEDIR****/smarts/local/conf* and ***BASEDIR****/smarts/local/rules* (as well as their nonlocal counterparts) typically reflects the product or product module within a product suite that uses the files. For more information, refer to *Product Specific Subdirectories* on page 8.

If desired, you can alter the location for rules files. For more information about how to do this, refer to *Controlling the Location for Rule Set Files* on page 14.

Note: If Cisco installs a site-specific patch at your location, then additional directories may be included under ***BASEDIR****/smarts/local*. You should never alter files in any of these additional subdirectories.

Modifying Files

After you install an application, you may need to modify files as part of the configuration process. For example, if you want to secure access to NCM, you need to change the password for the Admin account in the *serverConnect.conf* and *clientConnect.conf* files.

To modify files, use *sm_edit*, a utility that is installed with each product suite.

Note: You must use the *sm_edit* utility that is included with the installation of a product suite to modify files applicable to that suite. Do not, for example, edit IP files with the *sm_edit* that is installed with SAM.

When invoked, *sm_edit* opens the specified file in a text editor. This utility ensures that modified files are always saved to the appropriate local area and that nonlocal copies of all files remain unchanged. If an appropriate subdirectory does not exist for the file you are modifying, *sm_edit* creates the appropriate subdirectory before saving the modified file to that location. For files with header information set for encryption, *sm_edit* encrypts certain fields in the file. In addition, *sm_edit* preserves the file permissions of modified files, which helps ensure that important configuration files are not altered by unauthorized users.

Note: You can configure Windows to automatically invoke the *sm_edit* utility when you open a file through Windows Explorer, see *Associating Files With sm_edit on Windows* on page 13.

To use *sm_edit* from the command line, specify the file name and include the subdirectory under */local* where the file resides. For example, to edit the *ics.conf* enter:

```
# BASEDIR/smarts/bin>sm_edit conf/ics/ics.conf
```

In this example, *sm_edit* searches in the ***BASEDIR****/smarts/local/conf/ics* directory for the *ics.conf* file. If it finds the *ics.conf* file, it opens the file in a text editor. If *sm_edit* does not find the *ics.conf* file in the ***BASEDIR****/smarts/local/conf/ics* directory, it creates a local copy of the *ics.conf* file located in the ***BASEDIR****/smarts/conf/ics* directory.

If the appropriate subdirectory does not exist in the local area for the file you are modifying, *sm_edit* creates that subdirectory and saves the modified file there.

Specifying the Text Editor Used by the sm_edit Utility

The *sm_edit* utility is not a text editor. Instead, it uses the text editor specified by one of the following means:

- SM_EDITOR environment variable. For more information about SM_EDITOR, refer to *Environment Variables Used By NCM Software* on page 95.
- VISUAL environment variable

- EDITOR environment variable
- Finally, if these environment variables are not defined, *sm_edit* uses the edit program on UNIX and the WordPad program on Windows.

You can also specify the editor that *sm_edit* should use by providing the `--editor` argument when invoking the utility.

Associating Files With sm_edit on Windows

On Windows systems, you can configure the system so that *sm_edit* is automatically invoked when a file is opened through the Windows Explorer file browser. You do this by associating the appropriate file extensions with the *sm_edit* utility.

If you have more than one product suite installed on the same host, this association only works correctly with one product suite (the suite from which the associated *sm_edit* is run). For the other suites, you must run the *sm_edit* using a command prompt in the product suite containing the file you want to modify.

You can associate the following file types with *sm_edit*:

- *.conf* for configuration files
- *.asl* for ASL rulesets
- *.template* for data import files

Note: It is possible that another software program uses one or more of these file extensions. If an association between these file types and another program already exists, and you change the association, you may affect the operation of that software. In addition, the *sm_edit* utility will not open a file outside of the ***BASEDIR****/smarts* hierarchy. Attempting to open such a file will cause an error.

Associating File Types on Windows Systems

To associate a file extension with the *sm_edit* utility, use the following procedure:

1. From Windows Explorer, locate a user modifiable file within a product module or suite. For example, the *ics.conf* file located in the ***BASEDIR****/smarts/conf/ics* directory of a given suite.
2. Right-click on the file and select *Open With* > *Choose Program* from the menu.
3. From the Open With dialog box, select *Other..*.
4. Select the *sm_edit* utility located in the ***BASEDIR****/smarts/bin* directory. This adds *sm_edit* to the program selection list.

5. Choose *sm_edit* from the Open With program list.
6. Check the box beside “Always use this program to open these files.”
7. Click **OK**.

Repeat this procedure, except for steps 3 and 4, for each file type you associate with the *sm_edit* utility.

To remove a file type association from the *sm_edit* utility, uncheck the box beside “Always use this program to open these files”.

How NCM Locates User Modifiable Files

It is important to understand how NCM software locates user modifiable files at runtime. In general, when looking for a user modifiable file, NCM first searches in ***BASEDIR***/*smarts/local* or one of its subdirectories. If the file is not found, NCM then proceeds to look for the file in directories that are not intended to contain modified files. For example, when NCM searches for a configuration file, it first looks in ***BASEDIR***/*smarts/local/conf*. If the file is not found, then NCM proceeds to look in ***BASEDIR***/*smarts/conf*.

Note: ***BASEDIR***/*smarts* can refer to different product suite directories. Therefore the configuration file in the example could be an IP (Availability Manager or Performance Manager) configuration file, an APP file, or a SAM configuration file.

Controlling the Location for Rule Set Files

If you wish to locate your ASL rule sets in an area other than the default location (***BASEDIR***/*smarts/local/rules*), you should set the SM_RULESET_PATH environment variable. When you set SM_RULESET_PATH, NCM software first searches for rule set files in that path location or in one of its subdirectories. If the file is not found, NCM then proceeds to look, as described above, in the default location for modifiable rule set files.

To set SM_RULESET_PATH, add it to the *runcmd_env.sh*.

Note: You normally only define the SM_RULESET_PATH if you are writing custom ASL rule sets to work with your NCM software.

For more information about SM_RULESET_PATH, refer to *Environment Variables Used By NCM Software* on page 95.

Where NCM Writes Output Files

The ***BASEDIR****/smarts/local* directory is also the default location for files written by NCM software. Writeable files include logs, repository files, and saved consoles. If desired, you can alter the location for log files. For more information, refer to *Controlling Where NCM Writes Log Files* on page 16. Table 7 defines the default subdirectories found under ***BASEDIR****/smarts/local* that contain writeable files.

INSTALLATION PATH	DIRECTORY	DESCRIPTION
BASEDIR*/smarts/local*	*/consoles*	NCM consoles that have been saved.
	/logs	Server and adapter log files that the software has written. Also where archived notifications are saved.
	/repos	Repository files that the software has written. Unless otherwise specified, the InCharge Beacon and ACM save their repository files to a unique location: *BASEDIR/smarts/local/repos/beacon*

Table 7: **Default Subdirectories for NCM Writeable Files**

For Service Assurance Manager, the ***BASEDIR****/smarts/local/logs* directory includes a <servername>.audit file. This file contains audit entries that are added to all notifications in Service Assurance Manager. The entries, which are in time order, include the following information: Date/Time, Notification Name, Source, User, and Action.

The directory also includes a <servername>-statistics.log. This file contains entries for every invocation of the following operations:

```
dmctl -s servername invoke ICS_ActionManager::ICS-
ActionManager dumpStatistics <fileName>

dmctl -s servername invoke ICS_AutoActionManager::ICS-
AutoActionManager dumpStatistics <fileName>
```

The fileName is optional; otherwise it is <servername>-statistics.log.

The information in this file enables network personnel to monitor the execution of custom actions, and to monitor the performance of administrator-defined escalations.

Controlling Where NCM Writes Output Files

NCM software defines the location of writeable files using the SM_WRITEABLE environment variable. By default, SM_WRITEABLE is defined as ***BASEDIR****/smarts/local*. This means that saved consoles are written to ***BASEDIR****/smarts/local/consoles,* log files are written to ***BASEDIR****/smarts/local/logs,* and repository files are written to ***BASEDIR****/smarts/local/repos.*

If you want all writeable files to be written elsewhere, you should set the path using the SM_WRITEABLE environment variable.

To set SM_WRITEABLE, add it to the *runcmd_env.sh* file.

Controlling Where NCM Writes Log Files

If you want NCM software to write log files to an area other than the default location, you should set the path using the SM_LOGFILES environment variable.

To set SM_LOGFILES, add it to the *runcmd_env.sh.*

SM_LOGFILES takes precedence over SM_WRITEABLE. For more information about SM_LOGFILES, refer to *Environment Variables Used By NCM Software* on page 95.

sm_logerror

On Solaris and Linux systems, every process creates a child process, *sm_logerror,* at startup. This process has one purpose: to print a stack trace of its parent (by invoking the *pstack* program) should the parent request it (which usually happens only when the parent encounters a fatal error).

Every process also starts a child running an external authenticator, *sm_authority* (in the default configuration). This program either gets client credentials to send to a server, or checks credentials received by a server. Since the authenticator itself is an NCM program, it starts its own sm_logerror. Thus, if you follow the tree of processes under an sm_server, you might see:

```
sm_server
    sm_logerror
    sm_authority
        sm_logerror
```

Both *sm_logerror* and *sm_authority* spend their time reading a pipe that connects them to the process that created them. However, this process is not necessarily their parent. If the server is started with `--daemon`, it daemonizes after starting *sm_logerror* and *sm_authority*. During daemonizing, the original *sm_server* process is replaced by a different one. The child processes, *sm_logerror* and *sm_authority*, become orphans and are then inherited by the *init* process, which is always process 1. The new *sm_server* process retains its connection to the pipes that connected the original to the *sm_logerror* and *sm_authority* processes, and the processes work as intended.

These processes must not be "killed". Killing *sm_logerror* will make it impossible to print a stack trace, and make debugging more difficult. Killing *sm_authority* will cause connection attempts to fail when credentials cannot be obtained or checked.

Using the smgetinfo Utility to Save Modifications

Each product suite includes an *smgetinfo* utility. The utility enables you to maintain a backup copy of all the essential files and customizations from your installation for that product suite. It also enables you to collect data about your current installation so that you can send the data to Cisco TAC for support purposes. The *smgetinfo* utility is located in the ***BASEDIR****/smarts/script* directory.

When run, the *smgetinfo* utility stores the following directories in a .tar.gz or .tar.Z file on UNIX, or in a .zip file on Windows:

- */conf*
- */local*
- */rules*
- */setup*

It also collects the following data and stores it in the file.

- The versions and locations of all installed products
- The version of smartsd
- The version of dmctl
- The versions of all executables in ***BASEDIR***/smarts/bin/system
- The libraries in the ***BASEDIR***/smarts/lib directories
- Information about the host, operating system, and CPU

To run the *smgetinfo* utility, change to the ***BASEDIR/****smarts/script* directory and execute the following command.

On UNIX:

```
smgetinfo
```

or

```
sh smgetinfo
```

On Windows:

```
smgetinfo.cmd
```

After the utility collects and stores the data and files, it displays the name and location of the .tar (.gz or .Z) or .zip file. If run from the Service Assurance Manager on Windows, for example, the utility displays:

```
All custom configuration info has been saved in

C:\TEMP\smgetinfoSAM.zip
```

Note: To save information on Windows, WinZip is required. WinZip is commonly shipped with Windows systems. If WinZip is not installed in its default directory, set the variable ZIPPER to point to the WinZip directory. To collect information about installed products and components on all platforms, the Perl interpreter (version 5.0 or later) must be installed.

3

Licensing NCM Software

NCM 1.1 applications use the FLEXlm License Server 9.2. You install and configure the license server after the NCM installation process. You will need a license file, which you obtain from Cisco, in order to complete this process.

This chapter explains how to obtain and install a temporary (sometimes referred to as evaluation or trial) and a permanent license for NCM software. Both types of licenses provide access to specific applications and features as well as control the number of systems discovered by the software. This chapter also discusses migration of licensing for customers upgrading to NCM 1.1. Licensing the replacement of older versions of the software with the new is different than running the different versions concurrently.

NCM licensing includes a volume component that controls the number of devices a CNCC NCM Availability Manager can discover. For information about volume licensing see *Volume Licensing* on page 27.

License Administration Tools

Cisco uses version 9.2 of the FLEXlm licensing software from Macrovision. If you are not familiar with FLEXlm, you may want to read the FLEXlm documentation. A PDF version of the *FLEXlm End User Manual* is automatically installed with the documentation into the ***BASEDIR****/smarts/doc/pdf* directory. The name of the file is *flex_ug.pdf*.

FLEXlm provides several utilities for managing licenses. These include utilities for stopping the license server, diagnosing licensing problems, and forcing the license server to reread the license files. Instructions on using these utilities can be found in Chapter 7 of the *FLEXlm End User Manual*.

On UNIX, the utilities are automatically installed into the ***BASEDIR****/smarts/bin* directory. On Windows, they are installed into the ***BASEDIR****\smarts\bin\system* directory, and must be run from that directory.

Obtaining a Temporary License for NCM Software

When your CD of NCM software is shipped, Cisco sends a temporary license via e-mail on the same day. A temporary license for the software is time-limited. It enables you to use all of the features provided by the software for a limited period of time. At the end of the trial period, the software will no longer start.

To obtain a trial license, contact your Cisco Technical Assistance Center (TAC). After approving your request, Cisco TAC will send you an e-mail message with an attached trial license. You should copy the license file, *trial.dat*, to the ***BASEDIR****/smarts/local/conf* directory. Once in place, you can begin using the software.

If you installed the software on multiple systems, you can do one of the following to license the software on each system:

- Copy the *trial.dat* file to each system where the software is installed.
- Edit the SM_LICENSE variable in *runcmd_env.sh* on each system where the software is installed so that it specifies the full path name of a single *trial.dat* file (by default runcmd_env.sh specifies a path to a local copy of *trial.dat*). When multiple product suites are installed on the same system, each suite will include a *runcmd_env.sh* in its ***BASEDIR****/smarts/local/conf*.

Procedure for Permanently Licensing NCM Software

The permanent license controls access to applications, features, and controls the number of systems NCM can discover. Permanent licenses are not issued with the delivery of the NCM CD-ROM. Cisco will only send permanent licenses to approved customers who have provided information about their production environment.

Installing and using a permanent license for the software requires three steps:

1. Obtaining a permanent license from Cisco
2. Installing the FLEXlm license server and license
3. Starting the FLEXlm license server

Obtaining a Permanent License

The Cisco permanent license is not included on the NCM CD-ROM. You will receive the license by e-mail after you purchase the software.

To generate a valid license, Cisco needs the following information:

- The host ID of the computer where the FLEXlm license server is running. For more information, see *Obtaining the Host ID of a System* on page 22.
- The installation path to ***BASEDIR****/smarts/bin* on your broker (or to the installation from which you want to run the license server).
- The operating system of this computer.

Send this information to Cisco TAC. For more information, see the *Supplement and Read Me First for Cisco Network Connectivity Center.*

Note: Additional information may be required if NCM is installed in multiple data centers.

Once Cisco receives this information, a license will be generated and sent to you as a MIME attachment to an e-mail message. When you receive the license file, save the attachment to a file named *smarts.lic* in your ***BASEDIR*** directory on the host where the license server will run. Typically this location should be the same host and ***BASEDIR*** as the NCM Broker.

Do not make any changes to the license file unless you need to change the port as described in *Running the License Server on a Different Port* on page 25.

WARNING: Do not save the *smarts.lic* file to the ***BASEDIR****/smarts/local/conf* directory. The **install_license** script will fail if it finds a *smarts.lic* file in this directory.

Obtaining the Host ID of a System

To find the host ID of the system where you want to license the software, use the *lmhostid* option to the **lmutil** command. The lmutil command is included with Cisco software.

On UNIX, the command line is:

```
BASEDIR/smarts/bin/lmutil lmhostid
```

On Windows, it is:

```
BASEDIR\bin\system\lmutil lmhostid
```

The output should resemble the following:

```
# /opt/InCharge6/SAM/smarts/bin/lmutil lmhostid
lmutil - Copyright (c) 1989-2003 by Macrovision Corporation.
All rights reserved.
The FLEXlm host ID of this machine is "80cfc16b"
```

For best results, send Cisco the exact output of the command by copying it from your screen and pasting it, along with your other system information.

Installing a Permanent License

Install the license using the **install_license** script. By default, this file is located in the ***BASEDIR****/smarts/script* directory. We recommend that the license server should be on the host where the NCM Broker is running. You must run the *install_license* script on the same host where the license server runs. This script requires superuser privileges (user ID 0) for UNIX, and administrator privileges for Windows.

For example, the command on UNIX systems is:

```
# BASEDIR/smarts/script/install_license.sh install <path to
license file>/smarts.lic
```

On Windows systems, for example, the command is:

```
BASEDIR\smarts\script\install_license.cmd install <path to
license file>\smarts.lic
```

You must specify the full path to the *smarts.lic* file.

The *install_license* script performs two functions:

- Installs the license into the ***BASEDIR****/smarts/local/conf* directory.
- Configures the system so that the FLEXlm license server starts automatically when the system boots.

If applicable, edit the SM_LICENSE variable in *runcmd_env.sh* on the system(s) where the software is installed so that it reflects the <port>@<hostname> of the FLEXlm license server.

Note: Do not follow these steps if you are adding additional licenses. Instead you should follow the steps in *Adding Additional Licenses* on page 30.

Starting the License Server

The FLEXlm license server (lmgrd) is automatically installed in the ***BASEDIR****/smarts/bin* directory. The FLEXlm license server runs as a daemon on UNIX and as a service on Windows and automatically starts the Cisco vendor license server.

Run the FLEXlm license server run at all times to ensure continuous operation of NCM. A CNCC server will not start if it is unable to contact the license server. Adapters and the Global Console will run if they are unable to contact the license server but they will display an error message.

The command for starting and stopping the license server varies according to the operating system on which it runs. When you install the software, this command is added to the proper system-specific directory so that it is invoked when the system starts. The procedure for manually invoking this command is described below.

One of two possible options must be supplied with the command. The *start* option starts the process; the *stop* option ends the process.

Solaris:

```
# /etc/init.d/SMARTS-License start
```

HP-UX:

```
# /sbin/init.d/SMARTS-License start
```

AIX:

```
# /etc/smarts.d/S80SMARTS-License start
```

Linux:

```
# /etc/init.d/SMARTS-License start
```

Windows:

To manually start the license server on Windows, do the following as administrator:

1. Select *Settings* > *Control Panel* > *Administrative Tools*.
2. Select *Services*.
3. Select the FLEXlm License Server service.
4. Right-click and choose **Start**.

Licensing NCM Software on Additional Systems

When you install NCM software on more than one system, you must ensure that the software on each system can connect to the FLEXlm license server in order to be able to check out licenses. This is done automatically for the system where you run the *install_license* script, but has to be done for each other installation. Follow these steps to change the port and host information:

1. Edit the SM_LICENSE variable in *runcmd_env.sh* so that it specifies the <port>@<lic_host> of the FLEXlm license server.
2. Repeat Step 1 for each system running the software.

Note: Failure to point the software to the FLEXIm license server will result in an error message for remote users each time they attempt to start the application after the evaluation license has expired. If you have received a permanent license to replace the evaluation license for the software, you must follow the above procedures for each installation of a product suite.

Licensing NCM Software Across Multiple Sites

In general, you should run a license server on each host running an NCM Broker. For each license server, you should follow the directions for installing a permanent license.

Each license server will need its own license file. Each license file controls the applications, features, and the number of blocks of licenses for discovered devices. Excess licenses on one system cannot be allocated to another.

Running the License Server on a Different Port

The license server can run on a different port---for example, if the default port, 1744 is used by another process. The first line of the permanent license file, labelled SERVER, contains the hostname, hostID, and port of the license server, as shown in the following example:

```
SERVER this_host 8323fcbf 1744
```

To run the license server on a different port:

1. Change the port number ("1744") on the SERVER line.
2. Force the license server to restart.

Invoke these commands from the ***BASEDIR****/smarts/bin* directory (***BASEDIR****\smarts\bin\system* in Windows):

```
    # lmutil lmdown -c <port>@<lic_host>
    # lmutil lmreread -c <port>@<lic_host>
```

All NCM systems must be configured to use the new port. For directions to do this see *Licensing NCM Software on Additional Systems* on page 24.

Migrating From Earlier Versions of Software

There are two basic scenarios to consider when migrating to InCharge 6.2: where version 6.2 is an upgrade from InCharge 6.0 and where two different versions run in parallel.

Upgrading from Version 6.0 to Version 6.2

During upgrades, where older versions of InCharge are replaced with version 6.2, the installation will upgrade the license server automatically. Before you upgrade the software, you must obtain a new license from Cisco and place it in the ***BASEDIR*** on the host running the license server.

The upgrade process determines where the old license server used to run and looks for InCharge 6.2 license in that ***BASEDIR***. If it does not find the new license, named *smarts.lic*, the installation stops. If it does find the new license, it renames the old license to *legacy-license.lic* and places a copy of the new license into the ***BASEDIR****/smarts/local/conf* directory.

You will need to change the port and hostname of each NCM product suite as described in *Licensing NCM Software on Additional Systems* on page 24 to point to the new license server.

Running Older Versions in Parallel with Version 6.2

For some upgrades and installations, it is necessary that the license server accept the new license file for the version 6.2 InCharge software as well as older software. In this instance, a single license server manages both the old and the new license files.

Note: If you want to run version 6.2 and pre-version 5.0 InCharge software, you will need to run a separate license server for each version.

The steps to run InCharge version 6.2 and an earlier version are:

1. Install the 6.2 software (in a different location than the older software).
2. Copy and rename the old license file *license.dat* from the ***BASEDIR****/smarts/local/conf* directory of the old installation to *legacy_license.lic* and place the file in the new (6.2) installation directory, ***BASEDIR****/smarts/local/conf*.
3. Open the old license file using **sm_edit**.
4. Change the following in the first two lines:
   - hostname to this_host
   - port from 744 to 1744 (or the port for the license server if different)
   - daemon path from *lmgrd* to ***BASEDIR****/smarts/bin/system/lmgrd* (***BASEDIR*** refers to the location of the old installation)

     (use *lmgrd.exe* for Windows)
5. Stop the old license server. Use the procedure in *Starting the License Server* on page 23 except substitute stop for start.
6. Uninstall the old license server using the **install_license** script. Replace the parameter install with uninstall. See *Installing a Permanent License* on page 22 for more information.

7. Run the install license script to configure the new license server. See *Installing a Permanent License* on page 22 for more information.

8. Start the license server. See *Starting the License Server* on page 23 for more information.

To check that the license server loaded the old and new license files, look at the license server's log file ***BASEDIR****/smarts/local/logs/flexlm.log*.

Verifying the FLEXlm License Server Version

The FLEXlm License Server must be version 9.2 in order to work with an InCharge version 6.2 installation. To verify the version of the FLEXlm license server, specify `-v` option to the FLEXlm license server command:

```
% /opt/InCharge6/SAM/smarts/bin/lmgrd -v
lmgrd v9.2 - Copyright (c) 1988-2003 by Macrovision
Corporation. All rights reserved.
```

Volume Licensing

The permanent license controls the number of discovered systems. The license server maintains a pool of licenses that get checked out in blocks to NCM applications on an as needed basis. A block consists of fifty devices. Each application can check out one block at a time.

For more information about volume licensing, see the *Network Connectivity Monitor IP Discovery Guide*.

Determining the Total Number of Licenses

Each license file contains a line that defines the licensed number of systems. The line begins with the phrase INCREMENT AP_SYSTEM_VOLUME. The total volume of system licenses is determined by the sum of licenses granted through AP_SYSTEM_VOLUME in all of the license files. An example of line in the license file is:

```
INCREMENT AP_SYSTEM_VOLUME sm_lmgrd92 6.2 permanent 10
DE07382C7891
    VENDOR_STRING=BlockSize=50
```

This example shows the volume information of system licenses. The number 10 indicates that there are ten blocks of licenses. Each block contains 50 licenses. Multiplying the two produces the total number of systems licensed by the file---500.

Determining the Number of Checked-Out Licenses

Two utilities help to keep track of the number of checked-out licenses. The lmstat option of the **lmutil** utility shows the total number of license blocks checked out from the license server. The **sm_tpmgr** utility shows the number of system licenses checked out and the number used for a single Availability Manager.

Total Number of Checked-Out Licenses

The lmstat option of the **lmutil** utility shows the total number of licenses checked out from the license server. The output of the utility can be confined to the number of license blocks checked out for devices.

Invoke this command from the ***BASEDIR****/smarts/bin* directory (***BASEDIR****\smarts\bin\system* in Windows):

```
# lmutil lmstat -c <port>@<lic_host> -f AP_SYSTEM_VOLUME
```

The following example (entered on one line) shows the number of device license blocks checked out from the license server running on port 1744 on host inst-sol8:

```
# $BASEDIR/bin/system/lmutil lmstat -c 1744@inst-sol8 -f
AP_SYSTEM_VOLUME
```

This line returns the following information:

```
lmutil - Copyright (c) 1989-2004 by Macrovision Corporation.
All rights reserved.
Flexible License Manager status on Mon 4/26/2004 16:59

Users of AP_SYSTEM_VOLUME:  (Total of 5 licenses issued;
Total of 5 licenses in use)

  "AP_SYSTEM_VOLUME" v6.2, vendor: sm_lmgrd92
  floating license

    root qa-gga /dev/tty (v3.00) (inst-sol8/2004 2502), start
Mon 4/26 14:19
    root qa-gga /dev/tty (v3.00) (inst-sol8/2004 2804), start
Mon 4/26 16:06, 2 licenses
    root inst-sol8 /dev/tty (v3.00) (inst-sol8/2004 1301),
start Mon 4/26 12:19, 2 licenses
```

This output shows that the license server is managing a total of 5 license blocks (or 250 devices). Three Availability Managers (two running on qa-gga and the other running on inst-sol8) have together checked out all 5 license blocks.

Checked-Out Licenses and License Blocks For a Single Availability Manager

To determine the total number of checked-out licenses for a single Availability Manager, use the sizes option to the **sm_tpmgr** command. This shows the total number of system licenses.

Invoke this command from the ***BASEDIR****/smarts/bin* directory (***BASEDIR****\smarts\bin\system* in Windows):

```
# sm_tpmgr --sizes
```

The output should include information about the number of checked-out system licenses and the current number of counted systems. The difference between the two numbers shows the number of unused licenses for a single license block. The following example shows that 150 system licenses (3 blocks) have been checked out, 108 of the licenses have been used, and the maximum limit of discovered systems is 200:

```
Total System Volume License Checked Out:         150
Total Systems in Topology:              108
Remaining Blocks of System License in License Server: 5
Remaining Server Licenses in License Server: 3
Maximum Number Of Systems: 200
```

Discovery and Licenses

Volume licensing applies to Availability Managers and permits the application to discover a predetermined number of systems.

As more devices are added to the topology, the block fills to its limit of 50. When this limit is reached, the application checks out another block of licenses. Blocks are checked out one at a time and a new block will not get checked out until all of the licenses in the previous block are used. Any unused blocks are checked back in at the end of the discovery process.

For example, an IP Availability Manager with no topology starts an autodiscovery process. It checks out a block of system licenses. When it adds the 50th device to its topology, the Availability Manager checks out a new block of system licenses. Supposing, however, that it does not find any other device, it will check the empty block back in to the license server when discovery ends.

Limiting Discovery

There are a couple of options to consider for sizing constraints. Discovery filters can limit the devices found during the discovery process. Availability Managers can also be contained with an upper limit of discovered systems.

You can limit the number of systems automatically discovered by an Availability Manager by setting autodiscovery filters. See the *Network Connectivity Monitor IP Discovery Guide* for more information about configuring discovery filters.

You can limit the number of systems that are added to the topology by setting a system limit on the discovery. By default, the system limit is set to 50 systems. If you change the system limit, you should consider making the limit a multiple of 50, the block size for device licenses.

Exceeding License Limit

If the number of discovered devices exceeds the licensed number of systems, you will receive an OutOfLicense event notification and all excess systems are placed in the pending list.

Returning Unused License Blocks

When an application shuts down with **sm_service** or **dmquit**, any license blocks checked out to the application are returned to the license pool. If the application does not shut down gracefully, the license blocks get returned to the license pool, but not as quickly as a gracefully shut down application.

Adding Additional Licenses

Adding features or increasing the number of licenses with FLEXlm is easy. Cisco will send you an e-mail with the license attached. You simply copy the license to the ***BASEDIR*** on the host with the license server and force the license server to reread the license files.

To add license files, follow these steps:

1. Copy the new file to the ***BASEDIR*** directory for the license server.
2. Modify the new file so that the port number and path match those in the original license.

3. Copy the new file to ***BASEDIR****/smarts/local/conf*.
4. Force the license server to reread the license files. Invoke this command from the ***BASEDIR****/smarts/bin* directory (***BASEDIR****\smarts\bin\system* in Windows):

   ```
   # lmutil lmreread -c <port>@<lic_host>
   ```

4

Controlling the Startup of NCM Software

NCM programs can be installed as services. A service is a program that, once started, is intended to run continuously. On both Windows and UNIX, a service is administered by the *sm_service* utility. NCM programs installed as services start automatically upon system reboot; those not installed as services (manual processes or disabled processes) require that you issue commands to start and stop them as necessary.

We recommend that you install NCM programs as services. You are prompted to make this choice during the software installation. When you choose to install a program as a service, the installation process automatically sets up the program accordingly.

The following are examples of NCM programs that should be installed as services:

- NCM Broker
- Service Assurance Global Manager
- Service Assurance Business Impact Manager
- Service Assurance Adapter Platform (including the SNMP Trap Adapter and Syslog Adapter)
- IP Availability and Performance Managers

Although services start automatically upon system reboot, there will be occasion for you to manually start (and stop) a program that was installed as a service. For information on how to do this, refer to *Starting and Stopping Services with sm_service* on page 40.

If you choose not to install a program as a service, the program is installed as a manual process. If you install a program as a manual process, you can later change it to run as a service. For more information about how to do this, refer to *Installing Programs as Services with sm_service* on page 37.

In addition, you can modify a service's settings. For more information, refer to *Modifying Service Parameters with sm_service* on page 42

Note: The *sm_service* utility cannot be used with the Failover System. For specific information about starting, stopping, and configuring the Failover System scripts, refer to the *InCharge Service Assurance Manager Failover System User's Guide*.

About the sm_service Utility

sm_service is a cross-platform utility (for UNIX and Windows platforms) that sets up the environment for standard NCM applications (for example, the broker, servers, and adapters), and that installs those applications as services.

Additionally, the *sm_service* utility can start non-NCM applications.

The implementation of the *sm_service* utility includes two separate programs: *sm_serviced*, a long-running, system-level program, and *sm_service*, a command-line tool that communicates user requests to *sm_serviced*.

The *sm_service* utility is installed with the software (one per system), and can be used from either the ***BASEDIR****/smarts/bin* directory on UNIX systems or the DOS prompt on Windows systems. Select the ***BASEDIR*** with the latest version of software. The utility can only be used to control services on the local machine.

sm_serviced and ic-serviced

sm_serviced, the component of the *sm_service* utility that manages programs installed as services (either at installation, or by way of the *sm_service* command line), is installed with the software on UNIX systems during the installation process, and automatically starts its operations.

Thereafter, a script, *ic-serviced*, can be used to start and stop *sm_serviced*. The script is stored in a system-specific location: */etc/init.d* on Solaris and Linux; */sbin/init.d* on HP-UX; and */etc/smarts.d* on AIX. The control of services can be affected through *sm_service*. For more information see *The sm_service Command Line* on page 35.

The *ic-serviced* script includes several variables. Default settings are assigned to the variables during the installation process. If necessary, the value for the SM_SERVICE_STARTDIR variable can be edited. The default setting for this variable is / and specifies the directory into which core files are written. If it is changed, the setting must point to a directory on a file system with enough free space to hold a core file of the system's largest server.

Note: Contact Cisco TAC for information about editing the remaining variables at the beginning of the *ic-serviced* script.

The ic-serviced script enables you to start and stop *sm_serviced* on UNIX systems and to check the status of *sm_serviced*.

If *sm_serviced* is not already running, issue the following command to start it:

```
# <system-dependent path>/ic-serviced start
```

To stop *sm_serviced* and all of the services managed by it, issue the following command:

```
# <system-dependent path>/ic-serviced stop
```

To check that *sm_serviced* is running and responding, issue the following command:

```
# <system-dependent path>/ic-serviced status
```

Note: If you uninstall the first software suite you installed, the ic-serviced script will be removed from the system.

The sm_service Command Line

The *sm_service* command line is the tool through which you communicate user requests to *sm_serviced*. The basic syntax for the command line is:

```
sm_service <action> <options> <arguments>
```

Standard Options

Table 8 lists the standard options that can be used with the *sm_service* utility.

OPTIONS	DESCRIPTION
`--help`	Print the help text and exit.
`--version`	Print the program version and exit.
`--logname=<name>`	Use <name> to identify sender in the system log. Default: The program's name.
`--loglevel=<level>`	Minimum system logging level. Default: Error.
`--errlevel=<level>`	Minimum error printing level. Default: Warning.
`--tracelevel=<level>`	Minimum stack trace level. Default: Fatal. <level>: One of None, Emergency, Alert, Critical, Error, Warning, Notice, Informational, or Debug. Fatal is a synonym for Critical.
`--output[=<file>]`	Redirect output (stdout and stderr). The file name is <file>, or the - -logname value if <file> is omitted. Log files are always placed in $SM_LOGFILES or $SM_WRITEABLE/logs.
`--`	Stop scanning for options.

Table 8: **Standard Options for sm_service**

The following example highlights the use of a standard option.

```
# BASEDIR/smarts/bin/sm_service --help
```

sm_service Actions

Table 9 lists the actions that can be performed with the *sm_service* utility.

ACTIONS	DESCRIPTION
`install`	Installs a program as a service. It can also be used to modify the parameters of a service.
`show`	Displays the status of the installed services. It can also display the command lines of specified services so that they can be replicated on other systems.

ACTIONS	DESCRIPTION
`start`	Starts one or more installed services.
`stop`	Stops one or more installed services.
`isstopped`	Exits with a non-zero status if the requested set of services is in a state of not running.
`remove`	Removes one or more installed services.

Table 9: **sm_service Actions**

The following example highlights an action with several install options.

```
# BASEDIR/smarts/bin/sm_service install --startmode=runonce
ic-broker /opt/InCharge6/SAM/smarts/bin/brstart
--port=1234
```

Note: The command must be typed as one line.

Installing Programs as Services with sm_service

If, at installation, you installed a program to run as a service, the service will automatically start up when your system reboots. If you did not install a program as a service, you can use *sm_service* to install it as a service at any time.

The install action enables you to install a program as a service. After you define the name, description, startmode, and path of the program, *sm_service* stores the information in an *sm_service* database located in the */var/smarts* directory (on UNIX). *sm_service* then uses the information to start and run the program as a service.

The syntax for the *sm_service* install action is:

```
# sm_service install [<install options>] <name> path/to/exe
[<args>...]
```

In addition to the common install options, such as name, description, and startmode, the command line must include an absolute path to the program you want to the program that you want to install as a service. Arguments that apply to the program that you are installing as a service can follow the path.

For example:

```
# BASEDIR/smarts/bin/sm_service install --startmode=runonce
ic-broker /opt/InCharge6/SAM/smarts/bin/brstart
--port=1234
```

Install Options

Table 10 lists the install options that can be used with the *sm_service* utility.

OPTIONS	DESCRIPTION
`--force`	Overwrite an existing service with the same name. The option is used to update or modify the parameters of an existing service.
`"--description=<DESC>"`	A short description of the service. Enclose the option with double quotation marks. On UNIX platforms, the description information will be printed along with the name of the service when the 'show' action is invoked. On Windows platforms, the description will appear in the first column of the Service Control Manager window (the so-called "Descriptive Name").
`--startmode=<mode>`	The service start policy. Where <mode> is one of **runonce** (start automatically when *sm_serviced* starts), **automatic** (starts automatically when *sm_serviced* does not detect that it is running), **manual** (requires an explicit start request), or **disable** (cannot be started). Default: **runonce**
`--env=<NAME>=<VALUE>`	A <NAME>=<VALUE> pair which will be placed in the process environment of the launched service. As many `--`env pairs as necessary can be specified. The syntax <NAME>= (with no specified value) has the effect of unsetting <NAME> in the environment of the launched program. `--`env arguments are applied left to right as they appear on the command line, and this ordering is preserved in the database.

Table 10: **Install Options for sm_service**

Note: Do not use the `--daemon` option with the install action.

Examples of the sm_service Install Action

The following provides examples of *sm_service* install action command lines.

```
# BASEDIR/smarts/bin/sm_service install --startmode=runonce
ic-broker /opt/InCharge6/SAM/smarts/bin/brstart
--port=1234
```

```
# BASEDIR/smarts/bin/sm_service install --startmode=runonce
--env=SM_BROKER=localhost:1234 trapadapter
/opt/InCharge6/SAM/smarts/bin/sm_trapd
```

Displaying Installed Services and Their Status

The *sm_service* show action displays previously installed services and their status.

The syntax for the action is as follows:

```
# sm_service show <name> [<name> ...]
```

The show action has one option:

```
--cmdline
```

When the action is used with its cmdline option, it displays the *sm_service* command line that installed the program as a service. The command line can be reproduced and used for installing the requested service(s) on another system.

In the absence of the cmdline option, the show action displays the status of the named service (or all services, when <name> is not given), including:

- Whether the service is running
- The name of the service
- A brief description of the service

Examples of the sm_service Show Action

The following provides an example of the *sm_service* show action.

For the status of a service:

```
# BASEDIR/smarts/bin/sm_service show ic-broker
```

```
RUNNING BROKER IC-Broker
```

For the command line of a service:

```
# BASEDIR/smarts/bin/sm_service show --cmdline ic-broker
```

```
--startmode=runonce ic-broker
/opt/InCharge6/SAM/smarts/bin/brstart --port=1234
```

Starting and Stopping Services with sm_service

The *sm_service* utility can be used to start and/or stop programs when necessary. However, the programs, which can be either manual processes or services, must be installed with *sm_service* before you can use this action.

The syntax for the *sm_service* start action is:

```
# sm_service start [options]
```

The syntax for the *sm_service* stop action is:

```
# sm_service stop [options]
```

Start Options

Table 11 lists the options for the start action.

OPTIONS	DESCRIPTION
`<name> [<name>...]`	Start specified processes
`--all`	Start all automatic or runonce services that are not running. (UNIX only).
`--pattern`	Start all processes with absolute paths that match the wildcard pattern.

Table 11: **Start Options for sm_service**

Stop Options

Table 12 lists the options for the stop action.

Options	Description
`<name> [<name>...]`	Stop specified processes
`--all`	Attempt to stop all running services (UNIX only).
`--pattern`	Stop all processes with absolute paths that match the wildcard pattern.
`--force`	Do not consider it an error if the service is not running.

Table 12: **Stop Options for sm_service**

Examples of the sm_service Start and Stop Actions

The following provide examples of the *sm_service* start and stop actions.

For the start action:

```
# BASEDIR/smarts/bin/sm_service start ic-broker
```

For the stop action:

```
# BASEDIR/smarts/bin/sm_service stop ic-broker
```

Checking Whether Services Are Stopped

The isstopped action for *sm_service* exits with a non-zero status if any of the requested set of services are not running.

Table 12 lists the options for the isstop action.

Options	Description
`<name> [<name>...]`	Return status for specified processes
`--all`	Return the status of all processes.
`--pattern`	Return the status of all processes with absolute paths that match the wildcard pattern.

Table 13: **Isstopped Options for sm_service**

Examples of the sm_service isstopped Action

The following provide examples of the *sm_service* isstopped actions. These examples also show the use of wildcards. For more information about wildcard syntax, see *Wildcards Used By NCM Software* on page 113.

This checks to see whether all of the services started from the */opt/InCharge6* directory are stopped:

```
sm_service isstopped --pattern '/opt/InCharge6/*'
```

This example checks whether all CNCC servers are stopped:

```
sm_service isstopped --pattern '*sm_server*'
```

Starting and Stopping Services from the Windows Desktop

Programs that are installed as services can also be started and/or stopped from the Windows desktop.

To start a service from the Windows desktop, perform the following:

1. Select *Settings* > *Control Panel* > *Administrative Tools*.
2. Select *Services*
3. Select the service.
4. Choose **Start**.

Note: To stop a service, choose **Stop** instead of **Start**.

Modifying Service Parameters with sm_service

The parameters for a service can be modified with the *sm_service* install action. (See *Installing Programs as Services with sm_service* on page 37 for additional information.) The force option, which overwrites the existing parameters, must be used with the install action when you modify a service.

The syntax for the *sm_service* install action (to modify service parameters) is:

```
% sm_service install --force
```

To modify service parameters, perform the following steps.

1. Stop the existing service with the *sm_service* stop action.
2. Use the *sm_service* show action with the `--cmdline` option to display the command line that was used to install the existing service.

3. Use the *sm_service* install action with the force option, copy the displayed command line from step 2 after the force option, and modify the parameters that need to be changed.
4. Start the modified service.

Removing Services with sm_service

One or more existing services can be removed from the system, when necessary, with the *sm_service* remove action.

The syntax for the *sm_service* remove action is:

```
# sm_service remove <name> [<name>...]
```

Example of the sm_service Remove Action

The following provides an example of the *sm_service* remove action.

```
# BASEDIR/smarts/bin/sm_service remove trapadapter
```

Default Parameters for Services

During installation, NCM products can be installed as services. When installed as services, default values are specified for the parameters that are associated with the services.

This section lists the default parameters associated with NCM products, and briefly describes them.

If you need to modify the default parameters of a currently installed service, use the steps found in *Modifying Service Parameters with sm_service* on page 42.

Common sm_server Options

Several *sm_server* options are common to all servers that are installed as services (the Availability Manager, for example, and the Service Assurance Manager). Table 14 lists these *sm_server* options.

Exceptions to these parameters are noted and described with the default parameters for the individual products.

OPTIONS	DESCRIPTION
`--name=<name>`	The name of the NCM Domain Manager (Domain Manager or Global Manager).
`--config=<cfg>`	Specifies the *sm_server* configuration file to use. Configuration files are loaded from the directories ***BASEDIR****/smarts/local/conf/<cfg>* and ***BASEDIR****/smarts/conf/<cfg>*.
`--port=<port>`	The alternate Domain Manager port.
`--bootstrap=<file>`	Specifies the alternate bootstrap configuration file. The file name is relative to the configuration directory (see `--`config). The default is bootstrap.conf.
`--ignore-restore-errors`	Ignore errors encountered while restoring the saved topology. This option is required when starting an IP Server (am, pm, am-pm, or dm) that will use a 4.x repository file
`--subscribe=<sub>`	Start a subscription adapter that automatically subscribes to the specified notifications.
`--output[=<file>]`	Redirect the output (stdout and stderr). The file name is <file>, or the `--`logname value if <file> is omitted. Log files are always placed in ***BASEDIR****/smarts/local/logs*.

Table 14: **Common sm_server Options**

To display a complete list of the *sm_server* options on your system, use:

```
# BASEDIR/smarts/bin/sm_server --help
```

Common sm_adapter Options

Several *sm_adapter* parameters are common to all adapters that are installed as services (the Syslog Adapter, for example, and the Trap Adapter). Table 15 lists these *sm_adapter* parameters or options.

Exceptions to these parameters are noted and described with the default parameters for the individual products.

OPTIONS	DESCRIPTION
`--name=<name>`	The name of the adapter.
`--model=<name>`	The name of model library to load.
`--tail=<path>`	Read input by tailing a file.
`--rserver=<name>`	Auto-reconnect parser to CNCC Manager.
`--server=<name>`	Connect parser to CNCC Manager.
`--output=<file>`	Redirect the output (stdout and stderr). The file name is <file>, or the --logname value if <file> is omitted. Log files are always placed in ***BASEDIR/****smarts/local/logs*.

Table 15: **Common sm_adapter Options**

To display a complete list of the *sm_adapter* options on your system, use:

```
# BASEDIR/smarts/bin/sm_adapter --help
```

Common sm_beacon Options

Several *sm_beacon* options are common to all beacons that are installed as services (for example, the Application Connectivity Monitor). Table 16 lists these *sm_beacon* options.

Exceptions to these parameters are noted and described with the default parameters for the individual products.

OPTIONS	DESCRIPTION
`--name=<name>`	The domain name of the beacon. Also -n <name>.
`--config=<cfg>`	Specifies the *sm_beacon* configuration file to use. Configuration files are loaded from the directories ***BASEDIR/****smarts/local/conf/<cfg>* and ***BASEDIR/****smarts/conf/<cfg>*.
`--port=<port>`	The alternate beacon listening port. Also -p <port>.
`--bootstrap=<file>`	Specifies the alternate bootstrap configuration file. The file name is relative to the configuration directory (see --config). The default is bootstrap.conf.
`--broker=<location>`	The alternate broker location as host:port. Also -b <location>.

OPTIONS	DESCRIPTION
`--model=<model>`	Load a model library. Also -M <model>.
`--savedir=<directory>`	Specifies an alternate directory for restoring object topology. The default is SM_WRITEABLE/repos/icf.
`--norestore`	Do not attempt to restore any saved objects.
`--noregister`	Do not register the beacon with the NCM Broker.

Table 16: **Common sm_beacon Options**

To display a complete list of the *sm_beacon* options on your system, use:

```
# BASEDIR/smarts/bin/sm_beacon --help
```

Common sm_sdi Options

Several *sm_sdi* options are common to the components of the SQL Data Interface Adapter that are installed as services. Table 14 lists these *sm_sdi* options.

Exceptions to these parameters are noted and described with the default parameters for the individual products.

OPTIONS	DESCRIPTION
`--broker=<location>`	The alternate broker location as host:port. Also -b <location>.
`--name=<name>`	The name of the SQL Adapter component. Also -n <name>.
`--config=<cfg>`	Specifies the directory of the *sm_sdi* configuration files: *sdi_ics.conf*, *sdi_sql.conf* and *sdi_odbc.conf*. Configuration files are loaded from the directories ***BASEDIR****/smarts/local/conf/<cfg>* and ***BASEDIR****/smarts/conf/<cfg>*.
`--bootstrap=<file>`	Specifies the alternate bootstrap configuration file. The file name is relative to the configuration directory (see `--`config). The default is bootstrap.conf.
`--nopriv`	Allow *sm_sdi* to run under a non-privileged user ID. This option must be specified in order for a non-privileged user to start *sm_sdi*.

Table 17: **Common sm_sdi Options**

To display a complete list of the *sm_sdi* options on your system, use:

```
# BASEDIR/smarts/bin/sm_sdi --help
```

NCM Broker

Broker

A client application, such as a console or an adapter, utilizes the broker to determine where CNCC servers are running.

The default options for the broker and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service install
--startmode=runonce
--description="CNCC NCM Broker"
--env=SM_CLIENTCONNECT=brokerConnect.conf
ic-broker
  /opt/InCharge6/SAM/smarts/bin/brstart
    --port=426
    --output
    --restore=/opt/InCharge6/SAM/smarts/local/repos/broker
/broker.rps
```

The value for <PORT> is configured during installation.

Service Assurance Manager

SAM Server (Global Manager)/BIM

The Service Assurance Manager Server (that includes CNCC Business Impact Manager, if licensed) monitors and manages multiple distributed domains.

The default options for the SAM Server and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service
install --startmode=runonce
--description="CNCC NCM Service Assurance Manager Server"
ic-sam-server
  /opt/InCharge6/SAM/smarts/bin/sm_server
    --name=NCM-SA
    --config=ics
    --port=0
    --ignore-restore-errors
    --output
```

SDI Adapter

The default options for the SDI Adapter component of the SQL Data Interface Adapter and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service
install --startmode=runonce
ic-sdi-server
--description="CNCC SDI Manager Server"
/opt/InCharge6/SAM/smarts/bin/sm_sdi
   --name=INCHARGE-SDI
    --config=sdi/sdi
    --output
```

SDI Summary Adapter

The default options for the SDI Summary Adapter component of the SQL Data Interface Adapter and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service
install --startmode=runonce
ic-sdi-summary
--description="CNCC SDI Summary Adapter"
/opt/InCharge6/SAM/smarts/bin/sm_sdi
   --name=INCHARGE-SDI
    --config=sdi/summary
    --output
```

Summary Device Adapter

The default options for the Summary Device Adapter component of the SQL Data Interface Adapter and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service
install --startmode=runonce
ic-sdi-device-summary
--description="CNCC SDI Summary Device Adapter"
/opt/InCharge6/SAM/smarts/bin/sm_sdi
     --name=INCHARGE-SUM-DEVICE
     --nopriv
    --config=sdi/device
    --output
```

Business Dashboard

The default options for the Business Dashboard and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service
install --startmode=runonce
ic-business-dashboard
```

```
--description="CNCC NCM Servlet Engine"
/opt/InCharge6/SAM/smarts/bin/sm_tomcat
    --output
      start
```

SAM Adapter Platform

The SAM Adapter Platform provides open integration with third-party applications.

The default options for the SAM Adapter Platform and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service install
--startmode=runonce ic-icoi-server
--description="CNCC NCM SAM Adapter Platform Server"
  /opt/InCharge6/SAM/smarts/bin/sm_server
    --name=NCM-OI
   --config=icoi
   --port=0
   --ignore-restore-errors
   --output
```

Syslog Adapter

The CNCC NCM Syslog Adapter tails or parses the contents of any system log file, and generates notifications to the Global Manager based on the contents of the file.

The default options for the Syslog Adapter and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service install
--startmode=runonce
--description="CNCC NCM Syslog Adapter" ic-syslog-adapter
  /opt/InCharge6/SAM/smarts/bin/sm_adapter
   --name=SYSLOG-NCM-OI
   --rserver=NCM-OI
   --tail=/var/log/syslog
   --model=sm_system
   --model=sm_actions
   --output
    icoi-syslog/syslog_mgr.asl
```

Trap Adapter

The CNCC NCM SNMP Trap Adapter collects and parses SNMP traps, and generates notifications to the Global Manager based on the contents of the traps.

The default options for the Trap Adapter and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service install
```

```
--startmode=runonce
--description="CNCC NCM SNMP Trap Adapter" ic-trapd-receiver
  /opt/InCharge6/SAM/smarts/bin/sm_trapd
    --name=TRAP-NCM-OI
    --server=NCM-OI
    --config=icoi
    --port=162
    --ascii
    --model=sm_actions
    --output
   --rules=icoi-trapd/trap_mgr_parse.asl
```

NetIQ Adapter

The CNCC NetIQ Adapter imports event data generated by the NetIQ AppManager. The default options for the NetIQ Adapter and their values are:

```
/opt/InCharge6/SAM/smarts/bin/sm_service install
--startmode=runonce
--description="CNCC Adapter for NetIQ"
  ic-netiq-adapter
  /opt/InCharge6/SAM/smarts/bin/sm_adapter
    --name=NETIQ-AM
    --port=100
   --broker=localhost:426
    --output=NETIQ-AM
    -M sm_system
    -M sm_sdi
    -D "samServerName=NCM-SA"
     ics/niqam/main.asl
```

Concord eHealth Adapter

The CNCC Adapter for Concord eHealth provides an integration point between CNCC and the Concord eHealth Suite.

```
/opt/InCharge6/SAM/smarts/bin/sm_service install
--startmode=runonce
--description="CNCC Concord eHealth Adapter"
ic-ehealth-adapter
  /opt/InCharge6/SAM/smarts/bin/sm_ehealth
  -DMONITOR_SERVER_RESTART=TRUE
    --output
```

CNCC NCM IP

Availability Manager-only Server

CNCC NCM IP Availability Manager monitors and diagnoses availability problems within networks.

The default options for the Availability Manager Server and their values are:

```
/opt/InCharge6/IP/smarts/bin/sm_service install
ic-am-server
--startmode=runonce
--description="CNCC NCM IP Availability Manager Server"
  /opt/InCharge6/IP/smarts/bin/sm_server
    --name=NCM-AM
    --config=icf
   --bootstrap=bootstrap-am.conf
    --port=0
    --subscribe=default
    --ignore-restore-errors
     --output
```

Note: The `--ignore-restore-errors` option is required when starting an IP Server (am, pm, am-pm, or dm) that will use a 4.x repository file.

Performance Manager-only Server

CNCC IP Performance Manager monitors devices within networks, and diagnoses performance problems that occur.

The default options for the Performance Manager and their values are:

```
/opt/InCharge6/IP/smarts/bin/sm_service install
ic-pm-server
--startmode=runonce
--description="CNCC Performance Manager Server"
  /opt/InCharge6/IP/smarts/bin/sm_server
    --name=CNCC-PM
    --config=icf
   --bootstrap=bootstrap-pm.conf
    --port=0
    --subscribe=default
    --ignore-restore-errors
     --output
```

Availability Manager and Performance Manager Server

The default options and values for the server that runs the Availability and Performance Managers together are:

```
/opt/InCharge6/IP/smarts/bin/sm_service install
ic-am-pm-server
--startmode=runonce
--description="CNCC Availability and Performance Manager
Server"
  /opt/InCharge6/IP/smarts/bin/sm_server
    --name=CNCC-AM-PM
    --config=icf
   --bootstrap=bootstrap-am-pm.conf
    --port=0
    --subscribe=default
    --ignore-restore-errors
     --output
```

Discovery Manager Server

CNCC Discovery Manager provides automated discovery of Layer 2 and 3 elements in IP networks.

The default options for the Discovery Manager Server and their values are:

```
/opt/InCharge6/IP/smarts/bin/sm_service install
ic-dm-server
--startmode=runonce
--description="CNCC Discovery Manager Server"
  /opt/InCharge6/IP/smarts/bin/sm_server
    --name=CNCC-DISCMGR
    --config=icf
   --bootstrap=bootstrap-dm.conf
    --port=0
    --ignore-restore-errors
     --output
```

Application Services Manager

Application Services Manager

InCharge Application Services Manager manages application services and correlates service performance with the state of the network infrastructure.

The default options for Application Services Manager and their values are:

```
/opt/InCharge6/APP/smarts/bin/sm_service install
--startmode=runonce
--description="SMARTS Application Services Manager Server"
ic-asm-server
  /opt/InCharge6/APP/smarts/bin/sm_server
    --name=INCHARGE-ASM
```

```
              --config=asm
              --port=0
              --nonpriv
              --ignore-restore-errors
              --output
```

Application Connectivity Monitor

InCharge Application Connectivity Monitor discovers TCP application services, and measures their availability and response time.

The default options for the ACM Adapter and their values are:

```
/opt/InCharge6/APP/smarts/bin/sm_service install
--startmode=runonce
--description="SMARTS Application Connectivity Monitor
Server" ic-app-poller
  /opt/InCharge6/APP/smarts/bin/sm_beacon
   --name=IC-ACM
        --config=app-poller
        --port=0
        --output
```

Note: For more information about default parameters for the Report Manager processes (ic-sdi-server and ic-sdi-summary), refer to the *InCharge Service Assurance Manager User’s Guide for Report Manager*. For information about the default parameters for the Service Assurance Manager Failover System, refer to the *InCharge Service Assurance Manager Failover System User’s Guide*.

5

Securing Access to NCM

This section describes the security features of NCM. The software provides several means by which administrators can set up security and control access to the system. This includes:

- Client authentication and other user rights and privileges
- Encryption of passwords in files
- Encryption of communication channels

Client-server connections are controlled on both the client and server sides of the system. The system is secured using authentication records and by assigning connection privileges on the server side. When a client initiates a connection to a server, the client must supply appropriate authentication to the server before the connection (as defined by the connection privileges) is permitted.

Note: An administrator can place access restrictions on certain console operations by applying user profiles. This ability to restrict users to certain operations is described in the *Network Connectivity Monitor Service Assurance Manager Configuration Guide*.

For added protection, authentication and other passwords are encrypted in the files that store them.

Communication channels (that is, TCP connections made via SMARTS Remote API) between servers, brokers, and adapters can also be encrypted. Instead of passing information as clear text, these components' communications can be encrypted used either a site secret, the Diffie Helman-Advanced Encryption Standard (DH-AES), or both. For new installations, encryption by DH-AES is enabled by default between processes that support encryption.

The security mechanism is enabled when you install the software, but you should change the default settings to further secure the software. The initial security settings are thoroughly documented and permit access to the system. As a result, you should change the default user names, passwords, and the secret phrase if you want to enforce access control to the software. In addition, you should restrict access to the security files, as described in *Limiting Access to the Security Files* on page 70.

How NCM Security Works

NCM's security mechanism applies to authentication and communication. Authentication occurs whenever a client program initiates a connection to a server program. The client passes a username and a password to the server. The server determines whether the client is allowed to connect and, if the connection is allowed, what privileges the client is granted. The communication component allows for the encryption of certain communications between NCM components.

To properly configure the security system, you must understand how the security system works. The answers to the following questions will help:

- Which programs are servers and which are clients?
- How does a server authenticate a client?
- How does a client obtain a user name and password to pass to a server?
- How does encryption apply to authentication as well as communication?

Server and Client Programs for NCM

Recognizing which programs function as servers and which function as clients will help you understand how to configure security. Simply stated, a client is any program that initiates a connection to another program. Programs can act both as a client and as a server, for example a CNCC NCM Service Assurance Manager connects as a client to an underlying CNCC NCM IP Availability Manager by invoking a data exchange adapter.

Table 18 lists some programs and shows how they can interact. Programs that have a client-server interaction have a dot at the intersection of the client column with the server row. For example, a CNCC NCM Availability Manager connects as a client to NCM Broker.

		Acts as a Client								
		Broker	Availability/Performance Man.	Service Assurance Manager	Global Console	Application Service Manager	Application Connectivity Monitor	SAM Adapter Platform	InCharge Beacon	SNMP Trap and Syslog Adapters
Acts as a Server	Broker		●	●	●	●	●	●	●	●
	Availability/Performance Man.	●		●	●		●	●		
	Service Assurance Manager	●		●	●					
	Global Console									
	Application Service Manager	●		●	●			●		
	Application Connectivity Monitor	●		●	●	●				
	SAM Adapter Platform	●		●	●	●				●
	InCharge Beacon	●			●	●				
	SNMP Trap and Syslog Adapters	●								

Table 18: **Server and Client Programs of NCM Application**

Note: The NCM Broker functions as both a server and a client. In addition, when Service Assurance Managers are deployed in a multi-tiered environment, a Service Assurance Manager may also function as both a server and a client.

Remember that the server and client programs listed in Table 18 might be installed on the same host. For example, it is not uncommon for the NCM Broker and a Service Assurance Manager to run on the same host.

How a Server Authenticates Client Connections

When a client program initiates a connection to a server, it must provide a username and password. A server authenticates a connection request by comparing the username and password it receives from a client to the authentication records of its configuration file, *serverConnect.conf*. The server uses the first authentication record that matches the information sent by the client. The authentication records are reread for each attempted connection so any changes to the file will take effect immediately.

The Global Console always displays a log on dialog box where a user must type a username and password. Other NCM applications can prompt for a username and password or be configured to automatically send the user name and password.

For example, when a Global Console connects to a Service Assurance Manager, it sends a username and password to the Service Assurance Manager. The Service Assurance Manager compares these credentials to the authentication record listed in *serverConnect.conf* file. If the first matching record allows the connection, the Service Assurance Manager accepts the connection and the client grants the user the privileges specified by the authentication record. If the user name and password do not match an authentication record, the connection is refused.

Automating Client Authentication

Other than the Global Console, most clients, by default, automatically send authentication information to the server. This information is stored in a *clientConnect.conf* file. A client that uses automatic authentication reads the records in the order that they appear, selecting the first record whose login user name matches the user running the client and whose target matches the name of the server being connected to. Once a it finds a match, the client sends the username and password to the target server as authentication credentials. If the authentication fails, the server does not allow the connection and the failure is recorded in the server’s log file. If the authentication succeeds the server tells the client its access privilege, which is enforced by the client.

The NCM Broker uses its own client connection file, *brokerConnect.conf*.

Client authentication files are reread for each attempted connection. As such, you can edit the configuration files at any time and the changes take effect immediately.

For example, when a CNCC NCM IP Availability Manager registers with NCM Broker, the Availability Manager sends a username and password from its *clientConnect.conf* to the broker. The broker checks the user name and password against the records in its *serverConnect.conf*. Based on the results, it will grant or deny a connection.

However, when the broker checks whether a registered Availability Manager is alive (by pinging the manager), it must authenticate with that manager. To do this, it finds a username and password from its *brokerConnect.conf* to send to the Availability Manager. The manager checks for the user name and password in its *serverConnect.conf* and either grants of denies the right for the broker to ping it.

Securing Authentication Records

Security for the authentication records consists of limiting access to the files and encrypting passwords contained in those files.

Limiting access uses the permissions feature of the operating system to allow select users access to the files. These users must include those who launch the NCM applications requiring access to the authentication records.

Encryption is based on a secret phrase, common to all of the applications that must interact. The phrase is used to encrypt password fields in the authentication records.

Securing Communication Between NCM Applications

You can enable encrypted communications between NCM applications. This encryption can help secure NCM against spoofing and man-in-the-middle attacks.

Configuring Authentication

This section describes the components of authentication. It covers the content and syntax of the security files, the default authentication records provided in each file, and how to protect the contents of the files through file access restriction and encryption.

Syntax of the Security Files

This section describes the syntax of the *serverConnect.conf*, *clientConnect.conf,* and *brokerConnect.conf* files.

Each file consists of one or more authentication records, each of which contains four fields. A line that starts with two forward slashes (//) or a pound sign (#) is considered a comment and ignored. The fields of each authentication record are separated by colons (:). Any white space before, after, or between fields is ignored. If the field value contains a space, you need to add an escape character, the backslash (\), before the space. When a backslash is encountered, the following character loses any special significance and is used as is.

Note: The first line of each of the configuration files contains encryption information. Do not change this line unless you want to disable encryption. To comment out the line add a second pound sign. For more information see *Encrypted Files* on page 76.

The *clientConnect.conf*, *brokerConnect.conf*, and *serverConnect.conf* are read from top to bottom. For example, when a client selects a record for automatic authentication, it uses the first record with a matching target and username. If that user name is denied a connection by the server, the client does not try again. The ordering of authentication records is important because you can use wildcards for certain fields. In general, more specific authentication records should be listed first. For more information about wildcards, see *Wildcards Used By NCM Software* on page 113.

Server Authentication

Server authentication involves setting up authentication records in *serverConnect.conf* file. Each record contains information to control access to particular servers, a username and password, and a connection privilege. To modify this file, use the **sm_edit** utility as described in *Modifying Files* on page 11.

Note: Always use **sm_edit** utility to edit the *serverConnect.conf*. This is particularly important if the passwords are encrypted.

Connection Privileges

Connection privileges are specified in the *serverConnect.conf* file. NCM's authentication mechanism provides for four levels of privileges:

- *All* provides full access to all server functions. It is required for all adapter-to-server and server-to-server connections and by all command line utilities, with a few exceptions. This level of authorization is also required by administrative consoles, however this level of access can be further restricted through user profiles. For more information about using user profiles to restrict access, console operations see the *Network Connectivity Monitor Service Assurance Manager Configuration Guide*.
- *Monitor* access supports a monitoring console. A monitoring console cannot change server database or configuration parameters except in special circumstances such as acknowledging notifications. Only consoles support Monitor access. If you run a secure broker, consoles need Monitor privileges to access the secure broker.
- *None* prevents access to the server. This privilege can be used to explicitly prevent a user from accessing the server.
- *Ping* is a special access privilege that allows connections to a server, but only to ping the server and check whether it is functioning. An NCM Broker requires Ping access to check the status of servers and is sufficient to allow the dmctl command line utility to connect to a server and execute the "ping" command.

Description of serverConnect.conf

The *serverConnect.conf* file defines who can connect to which server and what privileges they are granted. By default, two versions of the file are located in the ***BASEDIR****/smarts/conf* and the ***BASEDIR****/smarts/local/conf* directories on the system where the server is running. The first version does not contain encrypted passwords—so the default values are accessible by anyone who can read the file. The version in ***BASEDIR****/smarts/local/conf* contains encrypted passwords. The **sm_edit** utility saves changes to the file in ***BASEDIR****/smarts/local/conf*, encrypts the passwords, and does not modify the default non-encrypted version of the file.

Table 19 describes the four fields of an authentication record in the *serverConnect.conf* file.

The format of a record in *serverConnect.conf* is:

```
<target>:<NCM user name>:<password>:<privilege>
```

FIELD	DEFINITION	VALUE
target	Name of the server for which this connection is intended. A server will only read this line if its name matches the value of the target field.	Can be a matching pattern with wildcards or one of the following special values: • <BROKER> indicates that this record only applies to the broker. • ~<BROKER> indicates this record applies to all servers except the broker.
NCM user name	Username for the client requesting a connection.	Can be a matching pattern with wildcards or the following special value: • <DEFAULT> is provided for legacy clients that cannot send a user name. • <AUTO> is provided for site-specific credentials.
password	Password for the user requesting a connection.	Can be a password or one of the following special values: • <SYS> indicates that the username must be a valid login name on the local system. The server passes the credentials to the host operating system for validation. • <DEFAULT> is provided for legacy clients that cannot send a password. • <AUTO> is provided for site-specific credentials.
privilege	Access privileges of the client.	Valid values include: • All • Monitor • None • Ping

Table 19: **Field Descriptions for serverConnect.conf**

Recall that during the authentication process the server receives from a client its connection target, username, and password. The server checks each of its records looking for a match. When it finds the first match, it sends the appropriate privilege back to the client. Otherwise the server logs the failed authentication.

Automatic Client Authentication

Most clients connect to servers without requiring a user to enter a username or password. When this happens, a client parses a configuration file for a user name and password to send to the server for authentication instead of prompting a user. Most clients will use the *clientConnect.conf*, however the broker uses its own file, *brokerConnect.conf*.

The broker only uses *brokerConnect.conf* to send authentication to other processes so that it can ping them.

Description of clientConnect.conf and brokerConnect.conf

By default, these files are located in the ***BASEDIR****/smarts/conf* directory. The **sm_edit** utility saves changes to the file in ***BASEDIR****/smarts/local/conf* and does not modify the original version of the file.

The format of a record in the *clientConnect.conf* or *brokerConnect.conf* is:

```
<login user>:<target>:<CNCC NCM user name>:<password>
```

Table 20 describes the four fields of an authentication record in the *clientConnect.conf* and *brokerConnect.conf* files.

FIELD	DEFINITION	VALUE
login user	System login name of the person or process attempting a connection.	Can be a matching pattern with wildcards.
target	Name of the server to which the client is trying to connect.	Can be a matching pattern with wildcards or one of the following special values: • <BROKER> indicates that this record only applies to the broker. • ~<BROKER> indicates this record applies to all servers except the broker.
NCM user name	Username that is sent to server for authentication.	Can be a user name or one of the following special values: • <USERNAME> indicates that the user name under which the current process is logged in as is sent as the username. • <PROMPT> indicates that the client program asks the user to provide a username. • <AUTO> is provided for site-specific credentials. • <DEFAULT> mimics legacy client authentication.
password	Password that is sent to the server for authentication.	Can be a password or one of the following special values: • <PROMPT> indicates that the client program asks the user to provide a password. • <AUTO> is provided for site-specific credentials. • <DEFAULT> mimics legacy client authentication.

Table 20: **Field Descriptions for clientConnect.conf and brokerConnect.conf**

It is important to remember that a program runs under the login name of the user who started it. This has several implications:

- A user account must provide sufficient privileges for the program to function properly. For example, a CNCC server may need to run with root privileges because it sends ICMP pings or receives SNMP traps.
- A user's system login name must correspond to a username in the *clientConnect.conf* file or a username and password will not be sent to a server for authentication. In the *clientConnect.conf* record a user's login name and username do not have to be identical.

Special Authentication Values

Special authentication describes the configuration of the *serverConnect.conf*, *clientConnect.conf* and *brokerConnect.conf* using the special values <SYS>, <PROMPT>, <AUTO>, and <DEFAULT>. These values control system, site-specific and legacy authentication.

System Authentication

System authentication uses the operating system username and password to authenticate clients. This method is configured in *serverConnect.conf*. Using this method, you give every console operator an account on the host on which the server they access runs; they log in using the username and password defined for that account. (The account can be disabled to prevent actual interactive access to the system.) In this way, each console operator has a unique username and password. Accesses to the system can be traced to a particular user, and access can be individually revoked. The use of <SYS> for the password allows the use of common password administration across applications and avoids having the password appear in plaintext in the file in unencrypted installations.

This mechanism can readily be extended to provide similar controlled access for administrators. For example, you could add the following records to *serverConnect.conf*:

```
* : fred|george : <SYS> : All
* : * : <SYS> : Monitor
```

This would give the users “fred” and “george”, if they provide the passwords for their accounts on the host, All access. Other users providing the correct password are granted Monitor access.

You could even define a class of administrative users, for example, with usernames that start with ADM and provide all other users with Monitor access.

```
* : ADM* : <SYS> : All
* : * : <SYS> : Monitor
```

Console applications automatically prompt for a user name and password and do not use *clientConnect.conf*. In order for non-console applications to prompt, the value <PROMPT> must be used for the username and password in *clientConnect.conf*.

For Unix systems, the system authenticates a user name based on its password. Under Windows a response to the prompt will be checked against the local domain. If the system must check with any other domain manager, the format of the user name must include the correct Windows domain as well as the user name:

```
user_name@domain_name
```

Prompting for Authentication

In order for non-console applications to prompt for a user name and password, the corresponding fields in *clientConnect.conf* must contain <PROMPT>. The client must be attached to a terminal, in order for the system to prompt the user. Lines containing a <PROMPT> are skipped by programs that are not attached to a terminal, even if they would otherwise be selected.

Responses to <PROMPT> are checked against user name and password fields in *serverConnect.conf*.

Site-Specific Authentication

Site-specific authentication uses a site's secret to authenticate connections. When corresponding records in *serverConnect.conf* and *clientConnect.conf* contain the <AUTO> value for both the username and password, the client generates a password using its secret. The server validates the password to accept or reject the connection.

Note: Do not use this method unless you have changed the default site secret. The default site secret is known to all NCM installations.

So for connections to a particular Service Assurance Manager you could define a record in *serverConnect.conf*:

```
GM-Company-1:<AUTO>:<AUTO>:All
```

Then define a corresponding record in the *clientConnect.conf* files on the different hosts connecting to the server:

```
*:GM-Company-1:<AUTO>:<AUTO>
```

As long as both the client and server use the same secret phrase, the server will authenticate each connection to the Service Assurance Manager.

Note: Do not used <AUTO> for the *brokerConnect.conf* if it is used for other connections because the broker does not need All access, only Ping.

Legacy System Authentication

To provide the ability to interoperate with older NCM software, which may not support authentication, NCM provides a "default account" mechanism. When an incoming connection does not provide any authentication information, a server substitutes the standard user name and password with <DEFAULT>. After that substitution, the <DEFAULT>/<DEFAULT> authentication information is validated in exactly the same way any other user name/password combination is validated.

An incoming connection that explicitly specifies <DEFAULT>/<DEFAULT> is permitted. It is treated in exactly the same way as a connection that supplied no authentication information.

Default Authentication Records

This section describes the authentication records enabled by default. These default settings provide examples of user records in *serverConnect.conf* and how automatic authentication can be set up in *clientConnect.conf* and *brokerConnect.conf*.

Authenticating Legacy NCM Software

The NCM Broker uses the following record in *serverConnect.conf* to authenticate legacy client connections and provide full access. The target field, with a value of <BROKER>, identifies that the broker is the sole target for authentication. There are no corresponding entries in the *clientConnect.conf* or *brokerConnect.conf*. For more information on the value, <DEFAULT>, see *Legacy System Authentication* on page 66.

```
<BROKER> : <DEFAULT> : <DEFAULT> : All
```

In this configuration, the Global Console will not prompt for a username and password when connecting to the broker. The default configuration defines a non-secure broker, equivalent in security to NCM software prior to version 1.0.

Automating Client Authentication to the Broker

NCM programs connect to the NCM Broker under the following conditions:

- When a server starts, it connects to the broker as a client to register itself.
- When a client, including a console, needs to connect to a server, it connects to the broker to query it for the server's location.

The record in *serverConnect.conf* specifies that the broker should grant any client that sends the username BrokerNonsecure and the password Nonsecure full access to the broker.

```
<BROKER> : BrokerNonsecure : Nonsecure : All
```

The automatic authentication record in *clientConnect.conf* applies to any client that supports authentication and needs to connect to the broker. This record specifically selects the broker as its target.

```
* : <BROKER> :BrokerNonsecure : Nonsecure
```

These two records define a non-secure configuration for the broker. In this configuration, consoles do not prompt for a username and password when connecting to the broker. For information regarding a secure broker configuration, see *Configuring a Secure Broker* on page 78.

Automating Broker Authentication to Servers

The NCM Broker periodically pings all registered servers to determine their status. When the broker does this, it acts as a client. This record in *serverConnect.conf* permits the broker to ping a server to check if it is running.

```
* : BrokerPing : Ping : Ping
```

The *brokerConnect.conf* file contains the automatic authentication record—the only record in the file.

```
* : * : BrokerPing : Ping
```

For more information about the Ping access permission, see *Connection Privileges* on page 60.

Authenticating Administrative Users

The default administrative record provides full access to any server using the *serverConnect.conf* file.

```
* : admin : changeme : All
```

This record authenticates clients that provide the username “admin” and the password “changeme”. This account grants administrative privileges or full access, which is denoted by the value of *All* in the privilege field.

Note: We recommend that you change the password for this account after installation. You must make any changes to the corresponding records in both the *serverConnect.conf* and *clientConnect.conf* files. Only replace the admin user after you have set up a corresponding administrative account in the Global Console Manager.

The corresponding automatic authentication record in *clientConnect.conf* grants any client, which uses the file, full access to any server using the corresponding record in *serverConnect.conf* file.

```
* : * : admin : changeme
```

Because of this record appears later in the *clientConnect.conf* than the interactive users record (`* : * : <PROMPT> : <PROMPT>`), only noninteractive clients will be granted full access—unless interactive users enter the correct user name ("admin") and password ("changeme").

For more information about the All access permission, see *Connection Privileges* on page 60.

Authenticating Global Console Users

Two records in *serverConnect.conf* provide the console with monitoring access to servers. The first authenticates console users who provide the username “maint” and the password “maint”.

```
* : maint : maint : Monitor
```

The second console record authenticates console users who provide the username “oper” and a password of “oper”.

```
* : oper : oper : Monitor
```

Consoles do not use *clientConnect.conf* for authentication—there is no automatic authentication, the console prompts the user for a name and password and passes the information to the server for authentication.

Note: User profile restrictions can be used to further limit certain Global Console operations, for more information see *Network Connectivity Monitor Service Assurance Manager Configuration Guide*.

We recommend that you create separate accounts for each operator. The easiest way to do this is to comment out these records and use the authentication record that uses the system login facility.

Authenticating Users with a Valid System Account

The last default authentication record in *serverConnect.conf* permits users with a valid login account on the system where the server is running to connect. A client that uses this authentication record must type a valid system user name and password. For authentication, the system user name becomes the username. This record provides monitoring privileges.

```
* : * : <SYS> : Monitor
```

Authenticating Users Interactively

This authentication record in the *clientConnect.conf* prompts users to type username and password. This does not guarantee that the connection will be successful. The server must be able to validate the username and password using its *serverConnect.conf* file.

```
* : * : <PROMPT> : <PROMPT>
```

The position of this authentication record in the default *clientConnect.conf* file is important. Because it uses wildcard patterns to match for both login user and target, this record is always selected for interactive connections, even if a following record also matches. Non-interactive programs skip this record because they cannot prompt for information.

This record could match the following *serverConnect.conf* authentication records described in *Authenticating Global Console Users* on page 69:

```
* : maint : maint : Monitor
* : oper : oper : Monitor
```

and the record described in *Authenticating Users with a Valid System Account* on page 69:

```
* : * : <SYS> : Monitor
```

Limiting Access to the Security Files

Prior to NCM version 1.1, the security mechanism used by the software requires that usernames and passwords appear in plaintext files. Because of this, you should configure security in such a way that each security file is only readable by those programs or users that require it. The security files installed with NCM, which should be edited after installation, are readable by anyone.

Because CNCC servers and the NCM Broker typically run as root on UNIX systems and as Local Administrator on Windows systems, the s*erverConnect.conf* and *brokerConnect.conf* files should only be readable by the root or Local Administrator users and no one else.

The simplest method for creating a secure setup for users and client programs is to provide two separate *clientConnect.conf* files. One *clientConnect.conf* file, which can remain readable by anyone, should only contain entries that make client programs prompt for passwords. This *clientConnect.conf* file will not contain passwords.

For client programs, create a separate *clientConnect.conf* file that contains the authentication information necessary for non-prompting programs to access CNCC servers. This *clientConnect.conf* should only be readable by the user(s) under which these programs run. Client programs use the SM_CLIENTCONNECT environment variable to find this *clientConnect.conf* file. You can specify SM_CLIENTCONNECT in the service startup file for each service. For clients that are installed as services, you can use the `--env` option to the **sm_service** utility to edit the parameters of a service.

Password Encryption

You can encrypt the password values in *serverConnect.conf*, *clientConnect.conf*, and *brokerConnect.conf*. This encryption centers around a secret phrase. The secret phrase is modified using the **sm_rebond** utility. A default secret phrase is set by installation. The default secret phrase is known to all NCM installations and should not be considered secure.

When you modify files using **sm_edit**, part of the post processing encrypts passwords if this feature is activated. Special authentication values are never encrypted. An encrypted password is preceded by <E-1.0>.

For more information about password encryption see *Encrypted Files* on page 76.

Modifying Security Files

To modify the security files included with NCM, use the **sm_edit** utility as described in *Modifying Files* on page 11.

The utility will encrypt password fields in each of the security files as part of the post-editing process. In order to do so, the first line of each file must be of the format:

```
#<encrypted field>:1.0:n
```

The value n refers to the position of the password field in the security file's record and should not be changed. For *serverConnect.conf*, this value is 3. For *clientConnect.conf* and *brokerConnect.conf*, this value is 4. By default, this line appears in each of the security files. To disable encryption, add a second pound sign (#) to the front of the of the line.

To encrypt the files without editing them use **sm_edit** with the no-edit option:

```
sm_edit --noedit
```

Using **sm_edit** on the plain text files will apply the changes to the **BASEDIR/***local/conf* directory and not affect the text versions in **BASEDIR/***conf*, which will remain unaltered with the default values.

For more information about encryption see *Encryption* on page 75.

Note: Installation encrypts the security files using the default secret. The encrypted security files reside in **BASEDIR/***local/conf*.

Specifying Alternate Security Files

You can define separate *serverConnect.conf* and *clientConnect.conf* files on hosts where multiple servers or clients are running. This enables you to configure a system where certain users and/or servers use one file and other users and/or servers reference a different file. Some installations can share the same ***BASEDIR***, but have requirements that stipulate that the servers and/or clients operate differently.

You can change the default behavior and use environment variables to force the servers and clients to use distinct authentication files. You can use the SM_SERVERCONNECT or SM_CLIENTCONNECT environment variables for each server to specify different security files. There is no default file path or name, specify the full absolute path to the alternate security file. Because you specify the file name, you can have multiple files in the same directory.

For client programs, you can specify an alternate *clientConnect.conf* file by setting the SM_CLIENTCONNECT variable in the user's login script on UNIX systems or as a user variable on Windows systems. For example, you could add the following line to a user's login script:

```
SM_CLIENTCONNECT=export /home/user2/clientConnect_user2.conf
```

For server programs, you can specify an alternate *serverConnect.conf* file, by setting the SM_SERVERCONNECT variable in the command line that starts the server.

Encrypting Connections

NCM components communicate over TCP connections using the SMARTS Remote API. Clients using Remote API connections authenticate themselves to servers by sending credentials, nominally a username and password. When the credentials are passed as clear text, they can be snooped from the network or accessed via man-in-the-middle configurations.

You can encrypt certain connections using different keys for the Advanced Encryption Standard based on a combination of the Diffie-Helman standard and the site secret associated with the SMARTS suite installation. For more information about the site secret see *Encryption* on page 75.

Encrypted connections do not work with the following products:

- Pre-1.1 versions of the software
- Version 6.2 Global Console

- Perl API and Remote Java API
- CNCC Adapter for NetIQ AppManager

Levels of Encryption

NCM version 1.1 provides three levels of security above clear text communication: Diffie Helman-Advanced Encryption Standard (DH-AES), encryption based on the site secret, and DH-AES used in conjunction with the site secret.

Any encryption based on the site secret should only be used once the secret phrase has been changed using **sm_rebond** described in *Changing the Secret* on page 76.

Table 21 lists the four types of encryption connections.

SECURITY LEVEL	DESCRIPTION	ADVANTAGES	DISADVANTAGES
0, CLEAR, or CLEARTEXT	no encrypted communication	Backwards compatibility, no configuration (default behavior)	No security, passwords passed to servers as clear text
1	DH-AES	No site secret needed, no configuration (default behavior for new installations), protects against eavesdroppers	Slower connection than clear text or level 2 security, not secure against active attacks
2	Encryption based on site secret	Protects against eavesdropping and active attack, almost as fast as clear text	Must set site secret and keep it common across all communicating entities
3	DH-AES and site secret	Protects against eavesdropping and active attack, even by those who know the site secret	Slower connection than clear text or level 2 security, must set site secret and keep it common across all communicating entities

Table 21: **Encryption Levels for Connections**

Establishing Connections

Encrypted connections are configured using two environment variables:

- SM_OUTGOING_PROTOCOL— controls the connections that a NCM program acting as a client is allowed to initiate.
- SM_INCOMING_PROTOCOL— controls the connections that a NCM program acting as a server is allowed to accept.

Each of these can contain a list of security levels. If you specify more than one, separate them with commas. If the variable is not set, it is the same as specifying “0”.

When two NCM version 1.1 applications that support encryption try to connect, they exchange the levels of encryption each supports. Then a connection is attempted at the highest level they have in common. If that connection fails (for example, the two applications try to connect using level 3 but do not share the same site secret), they will not connect—-even if both have a lower level of encryption in common.

Applications that do not support encryption are treated as only having a CLEARTEXT level of encryption. If a Global Console must connect to a server, the server must allow clear-text connections.

When the two ends of the connection do not match, for example, SM_OUTGOING_PROTOCOL is 3 at server A and SM_INCOMING_PROTOCOL is 2 at server B, a connection cannot be formed. Both client and server will report errors.

The default value for the SM_OUTGOING_PROTOCOL and SM_INCOMING_PROTOCOL is "1,0".

Suggested Encrypted Connections

We recommend that you configure your system to use encrypted connections wherever possible.

- Brokers should be able to communicate with clear text as well as encryption in both SM_INCOMING_PROTOCOL and SM_OUTGOING_PROTOCOL only if they need to monitor the status of pre-version 1.1 software, such as CNCC Managers. This is required in this configuration since the Broker acts as both a client and a server, and must be able to communicate with every component in the system. For brokers that only support version 1.1 software, only its SM_INCOMING_PROTOCOL needs to support clear text in order to communicate with a Global Console.
- Servers to which consoles are connected with should include clear text in the SM_INCOMING_PROTOCOL. The SM_OUTGOING_PROTOCOL does not necessarily have to include clear text. This allows consoles to connect in clear-text mode, but requires any outgoing connections to be encrypted. If a server must connect to a pre-version 6.2 application (for example, if a version 6.2 Service Assurance Manager must connect to a version 6.0 Domain Manager) then set SM_OUTGOING_PROTOCOL to clear text as well as encryption.

- Configure adapters with SM_OUTGOING_PROTOCOL set to require encryption. Only adapters that register with the broker (`--name` option) can accept incoming connections. If you have adapters that accept incoming connections, setting SM_INCOMING_PROTOCOL to require encryption is appropriate.

  Also, if the adapter must connect to, or be connected to, by pre-version 1.1 NCM software, add the clear text option to the appropriate variable.

- Configure any version 1.1 components that must run on networks outside the management domain (especially Beacons running on machines in the DMZ) with both SM_INCOMING_PROTOCOL and SM_OUTGOING_PROTOCOL set to encryption. Depending on the level of encryption, this will prevent snooping or man-in-the-middle attackers. You will not be able to connect directly to such a component using a console, nor will they be able to communicate with pre-version 1.1 software.

Encryption

By default, encryption is enabled during the version 1.1 installation process. All NCM suites and applications that will interact with each other must share a common secret phrase. The connection configuration files and files containing passwords to SNMP v3 devices all contain encrypted information.

The basis for encryption is a secret phrase that gets transformed into the file *imk.dat*. This file is the basis of authentication for NCM programs. The programs can use this site secret to encrypt passwords in the authentication files and other files as well as to encrypt connections between NCM programs.

At installation, the encryption is enabled with a default secret phrase. This phrase is **Not a secret** and the *imk.dat* file is copyable. The *imk.dat* is found in the ***BASEDIR****/local/conf* directory. To improve security, you should change the secret phrase using the **sm_rebond** utility.

Note: The secret phrase should be treated with the same care as a root password or highest level system administration password.

Changing the Secret

The **sm_rebond** utility changes the secret phrase and re-encrypts the files affected by the secret. This utility prompts for the old secret phrase and the new phrase then generates an *imk.dat* file and updates all the files containing encrypted information. This utility affects all of the applications running on the same host that use the same *imk.dat* file.

Note: For multi-host sites, **sm_rebond** must be run on each suite. The phrase must exactly match on each host in order for the applications to make a connection.

The secret phrase can consist of a combination of printable characters, integers and special characters. The secret phrase cannot be more than 1,024 characters long.

sm_rebond shuts down all of the processes run from the suite that were started using **sm_service** or ***s*****m_serviced** and that use the same *imk.dat*, re-encrypts the security files and the seed files, then restarts the processes once the phrase and encryption changes are made. All other processes should be shut down manually before running the utility then restarted once the utility is finished.

Note: **sm_rebond** only encrypts files that reside in ***BASEDIR****/local/conf* and three levels of subdirectories below that. To encrypt files outside of that directory area, contact Cisco TAC.

Locking the Secret

If access to encrypted data is a critical issue for your organization, Cisco provides the option to lock the *imk.dat* file that disable copies of the *imk.dat* file, even by restoration from a backup process. For more information about this level of security, contact Cisco TAC.

Encrypted Files

The encryption process uses the secret to encrypt passwords contained in the security files and seed files. The first line of each file marks the appropriate field or keyword for encryption. The original files are located in the ***BASEDIR****/smarts/conf* directory but local copies get written to ***BASEDIR****/smarts/local/conf*. The files are *serverConnect.conf*, *clientConnect.conf*, *brokerConnect.conf*, and seed files.

The first line of each of the files contains information labelling fields and keywords to be encrypted by the **sm_edit** process. The first line must be in place in order for encryption to work.

To change a password in the file, delete the current password and replace it with the new password in plain text. If the current value is encrypted, be sure to also delete the <E-1.0> tag that marks the password as encrypted as well as the current password. The special values <DEFAULT>, <PROMPT>, <SYS>, and <AUTO> in the security files do not get encrypted.

If you use **sm_edit** to modify the password, the appropriate values are encrypted as part of the process of closing and saving the file. If you use another method to edit the files, you need to encrypt the files by running the **sm_edit** utility as follows:

```
BASEDIR/smarts/local/conf/sm_edit --noedit <filename>
```

Security File Encryption

Password is the only field in the security files that gets encrypted. Passwords are located in the fourth field of a record in the *clientConnect.conf* and *brokerConnect.conf*. Passwords are located in the third field of the *serverConnect.conf*. For more information about modifying these files see *Password Encryption* on page 71.

Seedfile Encryption Details

The encryption for a seed file only applies to certain information that applies to SNMP v3 devices. For these devices a keyword that contains the authentication password can be encrypted. The first line of the seedfile must be:

```
#<encrypted field>:1.0:AUTHPASS
```

The AUTHPASS keyword is the authentication password for an SNMP v3 device.

For this line, the pound sign (#) must be the first character of the line. To comment out these lines, add a second pound sign to the front of the line.

Encrypted Connections

NCM 1.1 software uses two environment variables to set levels of encryption. There are three levels of encryption and a clear-text state. When two applications try to make a connection, they do so at the highest common level.

The encryption levels 2 and 3 depend on the site secret and level 1does not. Only NCM 1.1 suites that use the same secret phrase will be able to establish level 2 or 3 encrypted connections. For more information about encrypted connections see *Establishing Connections* on page 73.

Configuring a Secure Broker

You can configure the NCM Broker to run in a secure manner. Using a secure broker results in the following changes to how the software runs:

- Consoles prompt for a username and password to connect to the broker. Without a secure broker, consoles connect to the broker without authenticating.
- Other servers and clients use their respective *clientConnect.conf* files to determine what credentials to send to the broker, just as they use *clientConnect.conf* to determine what credentials to send to a server. In particular, you can configure the *clientConnect.conf* files so that clients and servers prompt for connections to the broker, as the console does, or specify the password in *clientConnect.conf*.

To configure and run a secure broker, complete the following steps:

1. Choose a unique username and password to replace the BrokerNonsecure/Nonsecure credentials. The new username and password will be used by both servers and clients:
   - Servers will use these credentials to register with the broker.
   - Clients will use these credentials to connect to the broker and determine the location of a server.

   For example, you could use the username “SecureBroker” and the password “Secure”. We recommend that you choose a unique username and password.
2. Use the **sm_edit** utility to open a local copy of the *clientConnect.conf* file, located in ***BASEDIR****/smarts/local/conf*. Edit this file, used by all clients and servers, so that NCM programs send the SecureBroker/Secure credentials when connecting to the broker. For example:

   ```
   *: <BROKER> : SecureBroker : Secure
   ```

   Conversely, you can configure *clientConnect.conf* so that clients and servers prompt for connections to the broker, as well as other servers. This involves replacing the password, “Secure” in this example, with <PROMPT>.

   ```
   *: <BROKER> : SecureBroker : <PROMPT>
   ```
3. Use **sm_edit** to make the following changes to the local *serverConnect.conf* file used by the broker:
   - Delete the line granting <DEFAULT>/<DEFAULT> access to the broker.
   - Change the BrokerNonsecure/Nonsecure line to grant Ping access rather than All access. Do not, however, delete this authentication record.

- Add a new authentication record that grants All access to the SecureBroker/Secure credentials. For example:

```
<BROKER> : SecureBroker : Secure : All
```

Other Security Features

Two environmental variables provide other methods of limiting security. One can be used to turn security on and off. The other limits access to servers to specific hosts.

Controlling Authentication

The environment variable SM_AUTHORITY controls the authentication security mechanism provided by NCM. Beginning with version 1.0 of NCM, the authentication mechanism is enabled by default. If necessary, you can set the value of this variable to disable it. This environment variable must be set on each system where the software is running.

Note: InCharge software prior to version 4.1 did not provide an authentication mechanism and the software functions as though SM_AUTHORITY were set to <NONE>. For InCharge software version 4.1, the default value of SM_AUTHORITY is <NONE>, meaning that clients and servers do not send authentication information and will be unable to register with or query a secure broker. Beginning with NCM version 1.1, the security mechanism is enabled by default.

The name of the environment variable is SM_AUTHORITY. It can have one of two values:

- <STD> enables security.
- <NONE> disables security.

When no value is specified, which is the default, NCM software versions 1.0 and later treat this the same as <STD>.

Note: The angle brackets (<>) are required.

When SM_AUTHORITY is set to <NONE>, NCM software behaves as if no security mechanism is in effect:

- Clients do not read *clientConnect.conf* and never prompt for a username or password. They always send <DEFAULT>/<DEFAULT> as their credentials.
- Servers ignore any incoming username or password and grant any connection All access.

A server running with SM_AUTHORITY set to <NONE> is different from a server that grants <DEFAULT>/<DEFAULT> All access. The former server ignores any authentication information that a client might send, while the latter, if it receives authentication information, attempts to validate it using its *serverConnect.conf* file. It is possible that the validation process may fail.

Add the SM_AUTHORITY environment variable to the *runcmd_env.sh* file in the ***BASEDIR****/smarts/local/conf* directory.

For example, define the following line in the *runcmd_env.sh* file:

```
export SM_AUTHORITY="<STD>"
```

All NCM programs started from this install area will invoke the *runcmd_env.sh* file. Any programs that are already running must be restarted.

Limiting Access to Servers

The `--accept` option to the **sm_server** command, described in *SM_MAIN_OPTIONS* on page 100, provides another method of access control. This option limits the hosts that can connect with the server. Before other security measures occur, incoming connections must pass the `--accept` option before authenticating.

Example Security Configurations

You can configure NCM installations with a variety of levels of security. Higher levels of security require more administrative effort and extra effort on the part of users of the system, but give you better control and visibility into the use of the software. The appropriate level of security is a decision you will have to make for your site. This section describes a two security levels and describes their advantages and disadvantages. You can choose one of these levels, or combine characteristics of several.

Default Security

After a new installation, default security uses the security files with the passwords encrypted based on the default secret phrase and connection protocols are set to use Diffie Helman encryption or clear text. With the default security, you should change the passwords for the usernames in the security files. This setup prevents casual inspection, but does not pose much of a deterrence from more strenuous efforts.

Note: It is essential that you change the default passwords for these usernames before using the system in a production environment.

While this setup provides basic authentication, it does not provide much manageability, visibility, or security. The oper/oper credentials must be known to all console operators; the admin/changeme credentials to all administrators. This makes it difficult to know who is actually connected to a server, based on their username. In addition, you cannot revoke the rights of one user without changing the username and password for all users at the same access level.

The *serverConnect.conf* and *clientConnect.conf* files include authentication records that improves user administration. First, besides the oper/oper credentials, there are maint/maint credentials that also provide Monitor access. This illustrates how one might define shared usernames that are still differentiated on the basis of role. Second, *serverConnect.conf* contains the following line:

```
* : * : <SYS> : Monitor
```

Anyone who can provide a username and password that the operating system considers valid is also granted Monitor access. In this scenario, you give every console operator an account on the host on which the server they access runs; they log in using the username and password defined for that account. (The account can be disabled to prevent actual interactive access to the system.) In this way, each console operator has a unique username and password. Accesses to the system can be traced to a particular user, and access can be individually revoked. The use of <SYS> for the password avoids having to potentially create a number of records in *serverConnect.conf*.

This mechanism can readily be extended to provide similar controlled access for administrators. For example, you could add the following records to *serverConnect.conf*:

```
* : fred|george : <SYS> : All
* : * : <SYS> : Monitor
```

This would give the users “fred” and “george”, if they provide the passwords for their accounts on the host, All access. You could even define a class of administrative users, for example, with usernames that start with ADM.

```
* : ADM* : <SYS> : All
* : * : <SYS> : Monitor
```

Controlled Security

Controlled security uses security files (with encrypted passwords), a new secret phrase, and passwords for the usernames in the security files that are different than their default values. Under controlled security, connection protocols are set to use the site secret and clear text. This setup protects against eavesdropping and active attacks as well as casual inspection.

The initial steps you should take to configure controlled security are:

1. Create a new secret phrase using **sm_rebond**. For more information see *Changing the Secret* on page 76.
2. Change the username and passwords in *serverConnect.conf*, *clientConnect.conf*, and *brokerConnect.conf*.

   These should provide control and visibility of access by individual users. Consider using system authentication to authenticate Global Console users as described in *System Authentication* on page 65. The new usernames and passwords should also address connections between servers and servers or servers and clients.

   If your software resides on a single host, a single username and password for automatic client authentication is usually sufficient. However, if you have clients, such as adapters, on multiple hosts, or generally in different security domains, it may be desirable to use different usernames and passwords for manageability and visibility. For example, if there is reason to believe that the username and password from one machine have been compromised, you can change them without having to change the configuration on any other machine.

   To achieve such a configuration, you need to assign unique usernames and passwords to clients that you consider to reside in separate security domains. Add a line granting access to each such username/password credential to the *serverConnect.conf* file of each server the client will access. Add a corresponding line to the *clientConnect.conf* file that the client will use.

Note: If you have multiple security domains on a single host, use the SM_CLIENTCONNECT environment variable so that different clients read different *clientConnect.conf* files. You can use the `--env` option of the **sm_service** utility to set the environment value when the service starts.

3 Configure the different NCM components to communicate with encrypted connections. The types of allowed connections depend on the versions of the software in use.

If your site solely uses version 1.1 software, most connections should be encrypted. In order to enable communication with the Global Console, SM_INCOMING_PROTOCOL for the broker and any server to which consoles connect should allow clear text as well as encryption.

If your site uses version 1.1 with older versions, both SM_OUTGOING_PROTOCOL and SM_INCOMING_PROTOCOL values should include clear text as well as encryption.

For more information see *Encrypting Connections* on page 72.

4 At the highest security level, you should run with a secure broker as well. To run in this mode, all your software must be at version 4.1 or higher. Setting up a secure broker is described in *Configuring a Secure Broker* on page 78.

6

Operation of the NCM Broker

A client application, such as a console or an adapter, utilizes the broker to determine where CNCC servers are running. When a CNCC server starts, it registers the host name of the machine it is running on and the TCP port it is listening on with the broker. CNCC clients retrieve this information from the broker so that they can communicate with the CNCC server.

The broker registry maintains the following information:

- The name of the CNCC server, including the host and TCP port it is running on.
- The status of each CNCC server.

  The broker checks the status of each CNCC server every five minutes by connecting to the host on which NCM is running and determining if the NCM process is running properly. If the broker is unable to connect or the process is not running then the broker changes the status of the CNCC server to Dead.

  - *Running* indicates that the broker is able to communicate with the CNCC server.
  - *Dead* indicates that the CNCC server exited unexpectedly or is unreachable. When a CNCC server properly shuts down, it notifies the broker and the broker removes it from its registry.
  - *Unknown* indicates that the broker was restarted and that it is querying its list of CNCC servers to determine their state.
- The process ID of each CNCC server. This is the process ID assigned by the host's operating system. In some cases when the broker is restarted, the process ID of each CNCC server is set to zero to indicate the broker does not know the process ID of the CNCC server.

- The last time the state of the CNCC server changed. This value is set when the CNCC server registers with the broker and is updated if the broker determines that the CNCC server is dead. When the broker restarts, it changes the status of CNCC servers marked Unknown to Running or Dead.

As noted above, the broker changes the status of the CNCC server to *Dead* when it cannot connect to the CNCC server. However, the broker continues to try to connect to the CNCC server every five minutes. If the broker succeeds in connecting to the CNCC server, it changes its state back to *Running*.

Viewing the Registry of the NCM Broker

The broker is a component of a NCM application. The broker is designed to automatically start each time the host on which it is installed is started.

You can use the *brcontrol* utility to view the list of registered programs.

Broker Command Line Option

To view the contents of the broker from a command prompt, use the following command:

```
% BASEDIR/smarts/bin/brcontrol
```

How the CNCC Clients Find the Broker

The NCM Broker facilitates communication between CNCC clients and CNCC servers. During installation, you are prompted for the host location of the broker and its port number. This information is stored in the SM_BROKER_DEFAULT environment variable. This variable is used by all programs to find the broker. The default host setting for the variable is *localhost*, and the default port setting is 426.

A CNCC client follows the steps listed below to determine the broker's location.

1 Checks to see if the broker's location was specified as an option at startup. When this option is specified, no other options are checked.

2 Checks the value of the SM_BROKER_DEFAULT environment variable. If this variable is set, no other options are checked.

3 Checks if the broker is running on the host *smarts-broker* and listening on TCP port 426.

4 If *smarts-broker* is not defined, the program checks port 426 on localhost.

Note: The host name *smarts-broker* is usually an alias, such as a DNS CNAME.

How to Change the Broker Environment Variable

The default location of the broker is specified at installation and stored in the SM_BROKER_DEFAULT environment variable.

There are two scenarios in which you may need to change the value of the environment variable after NCM software is installed.

- During installation, the default broker host and/or port settings were changed, but the entered values are incorrect.
- The broker is automatically installed when you install a NCM application or the adapters. Because only one broker should be active on a network, the broker on one of the hosts must be disabled, and the broker environment variable on that host must point to the host where the broker is running. This typically occurs when a NCM application is installed on one host and the NCM adapters are installed on a second host.

Correcting the Broker Settings

In the first scenario, you must edit the broker host and/or port value in two places. First, change the SM_BROKER_DEFAULT environment variable in the *runcmd_env.sh* file, which is located in the ***BASEDIR/****smarts/local/conf* directory:

```
SM_BROKER_DEFAULT=<host>:<value>
```

In the above script ***<host>*** is the location of the broker, and ***<value>*** represents the port of the broker. (The defaults are localhost for the location, and 426 for the port.) Edit the location and port as needed.

This variable becomes the default for all programs and any subsequent installations for this host.

Additionally, you must change the port value in the sm_service install broker command line to match the port value you set in the SM_BROKER_DEFAULT environment variable.

Use the sm_service show action with the cmdline option to display the existing install broker command line.

```
BASEDIR/smarts/bin/sm_service show --cmdline ic-broker
```

```
BASEDIR/smarts/bin/sm_service install
--startmode=runonce
--env=SM_CLIENTCONNECT=brokerConnect.conf ic-broker
   BASEDIR/smarts/bin/brstart
    --port=<PORT>
     --output
     --restore=BASEDIR/smarts/local/repos/broker/broker.rps
```

Then use the sm_service install action with the force option to change the port value for the broker.

```
BASEDIR/smarts/bin/sm_service install --force
--startmode=runonce
--env=SM_CLIENTCONNECT=brokerConnect.conf ic-broker
   BASEDIR/smarts/bin/brstart
    --port=<NEW VALUE>
     --output
     --restore=BASEDIR/smarts/local/repos/broker/broker.rps
```

Disabling the Broker

In the second scenario, you must change the SM_BROKER_DEFAULT host variable in the *runcmd_env.sh* file, which is located in the ***BASEDIR****/smarts/local/conf* directory:

```
SM_BROKER_DEFAULT=<host>:<value>
```

In the above script ***<host>*** represents the host location of the NCM Broker, and ***<value>*** represents the port of the broker.

This variable becomes the default for all programs and any subsequent installations.

Additionally, you must change the startmode value in the sm_service install broker command line from *runonce* to *disable*.

Use the sm_service show action with the cmdline option to display the existing install broker command line.

```
BASEDIR/smarts/bin/sm_service show --cmdline ic-broker
```

```
BASEDIR/smarts/bin/sm_service install
--startmode=runonce
--env=SM_CLIENTCONNECT=brokerConnect.conf ic-broker
   BASEDIR/smarts/bin/brstart
    --port=<PORT>
     --output
     --restore=BASEDIR/smarts/local/repos/broker/broker.rps
```

Then use the sm_service install action with the force option to change the startmode value for the broker to disable.

```
BASEDIR/smarts/bin/sm_service install --force
--startmode=disable
--env=SM_CLIENTCONNECT=brokerConnect.conf ic-broker
   BASEDIR/smarts/bin/brstart
    --port=<PORT>
     --output
     --restore=BASEDIR/smarts/local/repos/broker/broker.rps
```

Securing the Broker

Securing access prevents unauthorized usage of the broker, and protects it from being modified (for example, servers being deleted from the broker). Authenticating users and determining their privileges is accomplished through two files: *serverConnect.conf* and *brokerConnect.conf*. These files ensure that only authorized users access the broker. You must modify the local copies of these files, which are located in ***BASEDIR****/smarts/local/conf*. For more information see *Configuring a Secure Broker* on page 78.

7

Managing Log Files

NCM servers such as Domain Managers, Global Managers, and Open Integration servers, and adapter processes write log files containing status information and error reports. This chapter describes the naming convention for log files, how to start a new log file, and how to control the number of old log files that are saved.

Overview of Log Files

Log files are, by default, written to the ***BASEDIR****/smarts/local/logs* directory. This is the default location specified by the SM_WRITEABLE environment variable. For more information about SM_WRITEABLE, see *SM_WRITEABLE* on page 111.

All log files have a *.log* file type. The name of a CNCC server's log file is based on the server's name. For example, if the name of the CNCC server is NCM, the log file is named *NCM.log*. The names of log files written by adapters are described in the *InCharge IP Adapters User's Guide.*

NCM programs can maintain up to 1,000 different copies of backup log files. The number of saved log files is determined by the value of the SM_BACKUP_FILE_LIMIT environment variable. When a server starts up, it renames a file that matches its log file name, adding a *.bak* to the name; for example, *NCM.log* to *NCM.log.bak*. If a file with this name already exists, it is renamed *NCM.log.NNN*, and a new *NCM.log.bak* is created. By default, NCM programs save two log files, which does not include the active log file.

Starting a New Log File

You can request that a CNCC server or adapter create a new log file, often referred to as *rolling over* a log file. Log files grow indefinitely, though slowly, under normal conditions.

The roll_log command is invoked through the *dmctl* utility, which requires that you attach to the server or adapter with administrative privileges. The syntax of the command is as follows:

```
roll_log [file-name]
```

The *file-name* option enables you to specify a new name for log files. If you omit this option, the current naming convention is retained. If you specify a name, the new log file uses that name. Any new log files created with roll_log will also use this naming convention if a different name is not specified. The new name specified by file-name is handled in exactly the same manner as the `--output` option to the sm_server command.

For example, on Windows you would invoke the following command to start a new log file for a server named NCM:

```
C:\InCharge6\SAM\smarts\bin\dmctl -s NCM exec roll_log
```

On UNIX systems, the equivalent command is:

```
% /opt/InCharge6/SAM/smarts/bin/dmctl -s NCM exec roll_log
```

When roll_log is invoked, the server or adapter writes an informational message to the end of the current log file, then repeats the steps it executed at startup, moves the current *.log* file to the *.log.bak* file, and opens a new *.log* file. All subsequent logging information is recorded to the new file.

You can repeat the log file roll-over process as many times as you like. Note, however, that the number of log files that are saved is controlled by the value of SM_BACKUP_FILE_LIMIT. If you need to retain log files beyond this limit, you should copy or at least rename the *.log.bak* files as soon as they are created. For more information changing the number of saved log files, see *SM_BACKUP_FILE_LIMIT* on page 111.

Controlling the Number of Saved Log Files

NCM software can retain up to 1,000 rolled over log files. The number of log files retained is determined by the value of SM_BACKUP_FILE_LIMIT. When a CNCC server starts up, it renames a file that matches its log file name, adding a *.bak* to the name. If there is an existing *.bak* file, the *.bak* will be replaced with *.NNN*.

For example, if the SM_BACKUP_FILE_LIMIT environment variable is set to 3 the process is as follows:

1. A log file is created and it is named *NCM.log*.
2. The *NCM.log* file created in Step 1 is rolled over:
   - The *NCM.log* file created in step 1 is renamed to *NCM.log.bak*.
   - A new *NCM.log* file is created.
3. The *NCM.log* file created in step 2 is rolled over:
   - The *NCM.log.bak* file renamed in step 2 is now renamed to *NCM.log.001*.
   - The *NCM.log* file created in step 2 is renamed to *NCM.log.bak*.
   - A new *NCM.log* file is created.
4. The *NCM.log* file created in step 3 is rolled over:
   - The *NCM.log.bak* file renamed in step 3 is renamed to *NCM.log.002*.
   - The *NCM.log* file created in step 3 is renamed to *NCM.log.bak*.
   - A new *NCM.log* file is created.
5. The *NCM.log* file created in step 4 is rolled over:
   - The *NCM.log.bak* file renamed in step 3 is renamed to *NCM.log.003*.
   - The *NCM.log* file created in step 4 is renamed to *NCM.log.bak*.
   - A new *NCM.log* file is created.
6. Since the limit of *NCM.log.NNN* files that are retained is set to 3, when the *NCM.log* file created in step 5 is rolled over:
   - The oldest log file, *NCM.log.001* from step 3, is deleted.
   - The *NCM.log.bak* file renamed in step 5 is renamed *NCM.log.004*.
   - The *NCM.log* file created in step 5 is renamed *NCM.log.bak*.
   - A new *NCM.log* is created.

Alternative Method for Starting New Log Files on UNIX Systems

On UNIX systems you can send a SIGUSR1 signal to the process writing the log file. Upon receipt of the USR1 signal, a server or adapter process writes an informational message to the end of the current log file, then repeats the steps it executed at startup, moves the current *.log* file to the *.log.bak* file, and opens a new *.log* file. All subsequent logging information is recorded to the new file.

To send a SIGUSER1 signal to a process, you must first determine the process number of the CNCC server. For a Domain Manager, you can use the brcontrol utility. For an adapter, use the system's **ps** command. Once you obtain the process id, use the **kill** command of your shell:

```
% kill -USR1 <pid>
```

A

Environment Variables Used By NCM Software

This chapter describes the environment variables used by NCM software. These include environment variables specific to NCM, variables set by other software, and system variables.

How Variable Values Are Interpreted

This section describes how the software interprets the value of an environment variable. If a variable, such as SM_BROKER_DEFAULT, requires that its value be formatted in a particular manner, it is noted in the description of the variable.

How Integer Variables Are Interpreted

When the value of an environment variable is to be expressed as an integer, NCM interprets the value of the variable as follows:

- Any leading white space is skipped. If the next character is a plus sign (+) or a minus sign (-), it is also skipped. If the next character was a minus sign, the final numeric value is negated.
- If the value starts with a number other than zero, it may contain only the digits zero through nine, and it is considered a decimal number.

- If the value starts with "0x" or "0X", it may contain the digits zero through nine, as well as the letters "a" through "f" and "A" through "F". At least one digit or letter must follow the "x" or "X". The value is treated as a hexadecimal number.
- If the value starts with a zero and is not followed by a "x" or "X", it may contain only the digits zero through seven, and it is treated as an octal number.

Otherwise, the variable does not represent a numeric value. In most cases, an error message is printed. In a few cases, the numeric value is simply taken to be zero.

Table 22 provides examples of numeric values and how they are interpreted by NCM.

NUMERIC VALUE	INTERPRETED AS
`"0"`	The value zero.
`"  10"`	The value ten, the leading white space is ignored.
`"+010"`	The value eight, octal.
`"0x10"`	The value sixteen, hexadecimal.
`"-123"`	The value negative one hundred twenty three.
`"abc"`	An illegal value, may be treated as zero.
`"019"`	An illegal value, leading zero makes it octal, but octal numbers cannot contain a nine. This value may be treated as zero.
`"1  "`	An illegal value, leading white space is ignored but trailing white space is not. This value may be treated as zero.
`"0X"`	An illegal value, at least one digit or letter must follow the "OX". This value may be treated as zero.
`"- 23"`	An illegal value, embedded white space is not permitted. This value may be treated as zero.

Table 22: **Examples of Numeric Values**

How Boolean Variables Are Interpreted

When an environment variable is expressed as a Boolean, NCM interprets the value of the variable, when set, as follows:

- If the value starts with a letter, an uppercase "T" or a lowercase "t" or an uppercase "Y" or a lowercase "y", the variable is interpreted as true. If the value starts with any other letter, the variable is interpreted as false.

- If the value can be interpreted as an integer variable, and it has a value of zero, the variable is interpreted as false. If the value is anything other than zero, the variable is interpreted as true.
- For any other value the variable is interpreted as false.

Methods for Setting Environment Variables for NCM Software

You can use one of the following methods to set an environment variable. The difference between these methods is that one sets the environment variable across a product suite and the other sets the environment variable for a particular program.

Setting a an Environment Variable Suite-Wide

When necessary, you can set an environment variable so that a product suite installed in that location can use it. For example, the SM_BROKER_DEFAULT variable, the value of which is set during installation, is used in this manner.

To set an environment variable so that it can be used by the programs of a product suite, add it to the *runcmd_env.sh* file, which is located in the ***BASEDIR****/smarts/local/conf* directory of that product suite.

1. Use *sm_edit* to open the *runcmd_env.sh* file. Invoke *sm_edit* from the ***BASEDIR****/smarts/bin* directory:

   ```
   % sm_edit conf/runcmd_env.sh
   ```

   To open *runcmd_env.sh* on Windows systems, you must invoke *sm_edit* as follows:

   ```
   C:\>sm_edit conf\runcmd_env.sh
   ```

2. Add the environment variable and its value using the following syntax:

   ```
   export SM_AUTHORITY="<STD>"
   ```

3. Save the *runcmd_env.sh* file and close it.
4. Any NCM program within a product suite started after this point will use the applicable environment variables specified in the *runcmd_env.sh* file. NCM programs that are already running need to be restarted for any new environment variable to take effect.

Setting a an Environment Variable for a Program

You can also set an environment variable so that it only affects a particular program. Typically, this is done by adding the environment variable to the program's sm_service command line. For example, if you want a CNCC server to use a particular *serverConnect.conf* file, you would specify the SM_SERVERCONNECT variable in the sm_service command line for that program.

The following example sets the value of the SM_SERVERCONNECT environment variable to point to the *server_Connect_IP.conf* file.

1 Use the sm_service show action with the cmdline option to display the existing command line for the program.

```
% /opt/InCharge6/IP/smarts/bin/sm_service show --cmdline
ic-am-server
/opt/InCharge6/IP/smarts/bin/sm_service install
--startmode=runonce ic-am-server
--description="CNCC NCM Availability Manager Server"
--name=NCM-AM
    /opt/InCharge6/IP/smarts/bin/sm_server
        --name=NCM-AM
        --config=icf
        --bootstrap=bootstrap-am.conf
        --port=0
        --ignore-restore-errors
        --output
```

2 Use the sm_service install action with the force option to add the environment variable to the command line.

```
% /opt/InCharge6/IP/smarts/bin/sm_service install --force
--startmode=runonce
--env=SM_SERVERCONNECT=/opt/InCharge6/IP/smarts/conf
/serverConnect_IP.conf
ic-am-server
--description="CNCC NCM Availability Manager Server"
--name=NCM-AM
    /opt/InCharge6/IP/smarts/bin/sm_server
        --name=NCM-AM
        --config=icf
        --bootstrap=bootstrap-am.conf
        --port=0
        --ignore-restore-errors
        --output
```

3 Start, or stop and restart, the program.

Environment Variables for Users

The following list of environment variables can be applied to the software.

- SM_JAVA
- SM_JAVAHOME
- SM_DISPLAY
- SM_EDITOR
- SM_MAIN_OPTIONS
- SM_ENABLE_SNMP_SET
- SM_SNMP_BUG_COMPATIBLE
- SM_OUTGOING_PROTOCOL
- SM_INCOMING_PROTOCOL
- SM_BROKER_DEFAULT
- SM_BROKER
- LM_LICENSE_FILE
- SM_LICENSE
- SM_LMGRD_LICENSE_FILE
- SM_AUTHORITY
- SM_KEYFILE
- SM_CLIENTCONNECT
- SM_OKLOGIN_LEVEL
- SM_SERVERCONNECT
- TZ
- LC_TIME
- SM_FORMAT_DATE
- SM_FORMAT_TIME
- SM_FORMAT_DATETIME
- SM_FORMAT_HPTIME
- SM_FORMAT_INTERVAL
- SM_FORMAT_HPINTERVAL

- SM_BACKUP_FILE_LIMIT
- SM_WRITEABLE
- SM_LOGFILES
- SM_RULESET_PATH

SM_JAVA

This environment variable specifies a Java Runtime Environment (JRE) other than that in *BASEDIR/smarts/jre*. The contents are the paths to the libraries of the JRE. On UNIX, the value of SM_JAVA is added to the LD_LIBRARY_PATH (Solaris and Linux), LIBPATH (AIX), or SHLIB_PATH (HP-UX). On Windows, the value of SM_JAVA is added to the PATH.

SM_JAVAHOME

This environment variable is used in conjunction with SM_JAVA to specify a non-standard JRE. It specifies the location of java classes that are part of the JRE. SM_JAVA and SM_JAVAHOME must both be specified or neither are specified.

SM_DISPLAY

This environment variable specifies the location of a client's X-Windows display (typically for a console user). Output from a server tool that uses the X protocol is directed to the display specified by SM_DISPLAY. Only CNCC NCM Service Assurance Manager uses this variable.

SM_EDITOR

This environment variables enables you to set the text editor that is invoked by the *sm_edit* utility. When SM_EDITOR is not defined, the *sm_edit* utility uses the value of one of the following system environment variables: VISUAL then EDITOR. If these environment variables are not defined, the *sm_edit* utility uses the edit editor on UNIX systems and the WordPad editor on Windows systems.

SM_MAIN_OPTIONS

The environment variable SM_MAIN_OPTIONS enables you to set default values for the standard options supported by NCM software. You can override the values specified by SM_MAIN_OPTIONS by providing alternative values on the command line.

STANDARD OPTION	DEFINITION
`--accept=<hosts>`	Specify a list of hosts, by name or IP address, from which NCM programs accept connections. Multiple entries should be separated by commas. By default, NCM programs accept connections from any host. NCM programs that accept connections are dmstart, sm_server, sm_adapter, and sm_trapd. Note that sm_adapter and sm_trapd only accept connections when started with the `--name` option. If you wish to limit connections to the host on which the software is installed, specify the name or IP address of the host instead of localhost.
`--daemon`	Run the NCM program as a daemon (UNIX only).
`--errlevel=<level>`	Specify the minimum level at which error events are written to the standard error output. Possible values include: None, Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug. The default is Warning.
`--loglevel=<level>`	Specify the minimum level at which event messages are written to the system logging facility. Possible values include: None, Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug. The default is Error.
`--logname=<name>`	Specify the name used to identify the NCM program in the system log. On UNIX systems, the default is the program name; on Windows systems the default is the registered service name.
`--output[=<file>]`	Redirect the data sent to *stdout* and *stderr* to a log file. Log files are written to the ***BASEDIR/****smarts/logs* directory. If <*file*> is not specified, the value of `--logname` is used. If a log file with specified name already exists, it is moved to a backup file, ***BASEDIR/****smarts/local/logs/<file>.bak*. If the backup file already exists, it is deleted.
`--tracelevel=<level>`	Specify the minimum level at which a process stack trace is written to the standard error output. Possible values include: None, Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug. The default is Fatal.
`--useif=<IP address>`	Specifies the IP address that the NCM program should use as the source address for outgoing packets and the destination address for which it accepts incoming packets.
`--version`	Print the version of the NCM program.

Table 23: **Standard Options Supported by SM_MAIN_OPTIONS**

For example, if you set SM_MAIN_OPTIONS to `--errlevel=FATAL` then NCM programs will record all errors with a level of Fatal or higher.

SM_SNMP_BUG_COMPATIBLE

Cisco implementation of the SNMP protocol is strict in that the software only sends or receives SNMP messages that conform to the SNMP standard. Unfortunately, not all SNMP implementations are as strict. The result is that NCM programs may receive non-conforming SNMP messages. In some cases, the software can successfully interpret and handle non-conforming messages. There are two such cases:

- When the type of an SNMP variable is not the required type but the value can be unambiguously interpreted as the required type. An example is when a TimeTicks variable type is sent as an Unsigned32 or non-negative Integer32 variable.
- When the type of an SNMP variable is inconsistent with the version of the SNMP protocol that delivers the message. An example is when a Counter64 variable is sent in a version 1 (v1) SNMP message.

By default, NCM only accepts conforming SNMP messages. You can, however, set the SM_SNMP_BUG_COMPATIBLE environment variable, so that NCM accepts the non-conforming SNMP messages described above.

When set, any invalid SNMP message that is accepted by Cisco software is logged at a Warning level to pinpoint the source of the invalid SNMP message and the reason why it is invalid.

SM_SNMP_BUG_COMPATIBLE is a Boolean.

SM_ENABLE_SNMP_SET

This environment variable is disabled by default. When enabled (set to 1), it allows you to set the value of an agent or a snap device using SM_SNMP_Actions::Set method. When it is disabled, the method cannot be used to set the value of an oid.

SM_OUTGOING_PROTOCOL

This environment variable controls the kinds of connections an NCM program acting as a client is allowed to initiate. It can contain one or more numbers. These numbers specify the security level at which communications can occur. If more than one is specified, they must be separated with commas. If the variable is not set, it is the same as specifying "0". Software releases prior to NCM 1.1 cannot use encrypted connections.

SM_INCOMING_PROTOCOL

This environment variable controls the kinds of connections an NCM program acting as a server is allowed to accept. It can contain one or more numbers. If more than one is specified, they must be separated with commas. If the variable is not set, it is the same as specifying "0". Software releases prior to NCM 1.1 cannot use encrypted connections.

Variables That Control the NCM Broker

SM_BROKER_DEFAULT

This environment variable specifies the default location of the broker. When you install NCM software, you are prompted for the host location of the broker and its port number.

The broker's location can be specified in any one of the following formats: "`host:port`", "`host`" or "`:port`". When the host or port values are omitted, the default value is used. The default host value is "smarts-broker" or "localhost". The default port value is 426.

If the location of the broker changes, you must edit the *runcmd_env.sh* file on all systems where NCM software is installed. In addition, you may have to edit the broker startup script.

NCM software follows the steps listed below to determine the broker's location.

1. It checks to see if the broker's location was specified as an option at startup. When this option is specified, no other options are checked.
2. It checks the value of the SM_BROKER_DEFAULT environment variable. If this variable is set, no other options are checked.
3. It checks if the broker is running on the host *smarts-broker* and listening on TCP port 426.
4. If *smarts-broker* is not defined, the program checks port 426 on localhost.

Note: The host name *smarts-broker* is usually an alias, such as a DNS CNAME.

SM_BROKER

SM_BROKER should only be set in special cases when you want to override the default broker. For example, if you have to temporarily move your broker in an emergency or test situation you could set this variable.

Under normal circumstances you should use the SM_BROKER_DEFAULT environment variable to set the default location of the broker.

Variables That Control FLEXlm Licensing

LM_LICENSE_FILE

The LM_LICENSE_FILE environment variable is used by the FLEXlm license software. This variable specifies the location of the license file(s) or the license server(s) (`port@host`). In many cases, this variable has already been set for another software product that also uses the FLEXlm license software.

SM_LICENSE

The value of this environment variable is either the full pathname to the license file itself; or it is port@host, which means that there is a license server running at that port on that host.

SM_LMGRD_LICENSE_FILE

Similar to LM_LICENSE_FILE, this FLEXlm variable is the vendor-specific environment variable that specifies the location of the license file(s) or the license server(s). When both LM_LICENSE_FILE and SM_LMGRD_LICENSE_FILE are set, the SM_LMGRD_LICENSE_FILE environment variable takes precedence.

For more information about these environment variables, see the *FLEXlm End User Manual*, which is automatically installed with the documentation into the ***BASEDIR****/smarts/doc/pdf* directory. The name of the file is *flex_ug.pdf*.

Variables That Control NCM Security

SM_AUTHORITY

This environment variable determines the selection of the authority programs used by the NCM software for various levels of security. For example:

SM_AUTHORITY=IDENTIFY=program1,AUTHENTICATE=program2

Where Program1 is the program granting credentials, and Program2 is the program validating credentials.

Program1 and Program2 may refer to the same program.

- SM_AUTHORITY=<STD> is understood to mean:

SM_AUTHORITY=IDENTIFY=sm_authority,AUTHENTICATE=sm_authority

- SM_AUTHORITY=<NONE> is understood to mean:

SM_AUTHORITY=IDENTIFY=sm_authnone,AUTHENTICATE=sm_authnone

- SM_AUTHORITY=someAuthority is understood to mean:

SM_AUTHORITY=IDENTIFY=someAuthority,AUTHENTICATE=someAuthori
ty

Optional arguments can be specified after the authority name. A comma may not be included within the program name. A comma may appear in the arguments if the argument is quoted.

If SM_AUTHORITY is not specified, SM_AUTHORITY=<STD> is assumed.

SM_KEYFILE

This environment variable specifies the location of the *imk.dat* file. By default this variable refers to ***BASEDIR****/smarts/local/conf/imk.dat.*

SM_CLIENTCONNECT

This environment variable enables you to specify an alternate location for the *clientConnect.conf* client-side authorization file. You must specify the full path, including the file name. The default location is the ***BASEDIR****/smarts/local/conf/clientConnect.conf* directory.

SM_OKLOGIN_LOGLEVEL

This environment variable enables the logging of successful logins at FATAL severity, rather than INFORMATIONAL.

SM_OKLOGIN_LOGLEVEL is a Boolean.

SM_SERVERCONNECT

This environment variable enables you to specify an alternate location for the *serverConnect.conf* server-side authorization file. You must specify the full path, including the file name. The default location is the ***BASEDIR****/smarts/local/conf/serverConnect.conf* directory.

Variables That Control Date and Time Formatting

CNCC servers and adapters format dates and times for printing in log files, archive files, and various messages. You can control many aspects of this formatting with the following variables.

Most dates and times displayed in NCM Consoles are formatted by the consoles themselves and are not controlled by these variables.

Additional information can be found on many systems—in the man pages on UNIX systems, or the Microsoft Developer Network (MSDN) on Windows systems. If these are not available on your system, various sites on the Web publish this information.

- Each UNIX vendor generally makes its own documentation available on line at its site.
  - HP's documentation is available at http://www.docs.hp.com/.
  - Sun's documentation is available at http://docs.sun.com/.
  - IBM's is available at http://www.ibm.com/servers/aix/library/.
- Windows documentation is available at http://msdn.microsoft.com/library/.

TZ

This system environment variable determines what time zone is used when formatting a time. As such, it control the conversion from an internal time base (for example GMT, on UNIX systems) to the appropriate local time. The syntax is TZ=NNNshh:mmDDD, where:

- NNN is the time zone name
- s is an optional sign
- hh and mm are the offset in hours and minutes from GMT. Positive offsets represent time zones behind GMT, negative offsets represent those ahead of GMT.
- DDD is the corresponding name of Daylight Savings time when it is in effect

For example, in the Eastern United States, TZ is set to EST5EDT - Eastern Standard Time/Eastern Daylight Time, where EST is five hours behind GMT.

All UNIX systems use a system initialization file to set this variable. Windows systems time zone settings are usually set with the Date/Time functionality available from the Control Panel, however, the TZ variable can be used to change the lengthy format of the time zone on a Windows system. If you set TZ=EST5EDT on a Windows system, “Eastern Daylight Time” is replaced by “EDT”.

LC_TIME

This system environment variable controls the formatting of dates and times. By default, CNCC servers use time and date formats that are sensitive to locale settings. For example, if you chose a date format that includes the name of the day of the week, in the English locale, the first weekday is formatted as Monday, while in the French locale it is formatted as Lundi.

If LC_TIME is not set, the system variable LC_ALL is used. If LC_ALL is not set either, the system variable LANG is used. Finally, if LANG is not set, a default “C locale” (essentially US English) is used, except on Windows systems where the system default is used.

The locale names, and available locales, are system specific.

SM_FORMAT_DATE

This environment variable sets the date format used by the software. The contents of this variable constitutes an “strftime” format string. This type of format string contains characters other than “%”, which are copied as is, and conversion specifications that start with a “%” and usually consist of one additional character.

Table 24 lists common date conversion specifications:

Conversion Specification	Description
%a	Abbreviated weekday name.
%A	Full weekday name.
%b	Abbreviated month name.
%B	Full month name.
%d	Day of month [1 - 31]; single digits are preceded by 0.
%m	Month number [1 - 12]; single digits are preceded by 0.
%x	Appropriate date representation.
%y	Year within century [00 - 99].
%Y	Year including the century (for example 1993).
%Z	Time zone name or abbreviation, or nothing if no time zone information exists.

Table 24: **Common Date Conversion Specifications**

If SM_FORMAT_DATE is not defined, the format “%d-%b-%Y” is used. In the US English locale, this produces a date format such as “03-Jun-2002”.

There are many additional formats. While there are a few system-specific extensions, most UNIX systems support the same set of conversions. Beyond the basic conversions, however, the Windows implementation is unique.

SM_FORMAT_TIME

This environment variable controls the formatting of times to the nearest second. As with SM_FORMAT_DATE, the contents of this variable constitute an “strftime” format string.

Table 25 lists common time conversion specifications:

Conversion Specification	Description
%H	Format the hour using a 24-hour clock (00 - 24).
%I	Format the hour using a 12-hour clock (00 - 12).
%M	Minute (00 - 59)
%p	Locale's equivalent of either a.m. or p.m
%S	Seconds (00 - 61)
%X	Locale's appropriate time representation.
%Z	Time zone name or abbreviation.

Table 25: **Common Time Conversion Specifications**

NCM also supports the special conversion specification “%.”, which is described in the section *SM_FORMAT_HPTIME* on page 109.

If SM_FORMAT_TIME is not set, the format “%X%. %Z” is used. This displays the locale's appropriate time representation, followed by a high-precision time representation (if applicable), followed by the time zone.

Note: On most UNIX systems, %X in the default (“C”) locale will produce a time representation equivalent to “%H:%M:%S” - that is, hours, minutes, and seconds on a 24-hour clock. On Windows systems, the default representation is usually equivalent to “%I:%M:%S %p %Z”.

SM_FORMAT_DATETIME

This environment variable controls the formatting of combined dates and times. It is used when CNCC servers or adapters need to format a time representation.

SM_FORMAT_DATETIME is a combination of SM_FORMAT_DATE and SM_FORMAT_TIME, and is a “strftime” format string supporting the additional “%.” conversion operator.

If this variable is not set, the date and time format specifications (SM_FORMAT_DATE and SM_FORMAT_TIME, or their respective defaults) are concatenated with a single intervening space, and the result is used.

SM_FORMAT_HPTIME

Standard time formatting is limited to a precision of one second. When this is insufficient, NCM provides a mechanism for displaying times with a greater precision to the nearest nanosecond, with the actual resolution being about a millisecond.

CNCC servers and adapters determine, on a case-by-case basis, whether it makes sense to request that a high-precision field be included when formatting a time. If so, and if a “%.” conversion operator is present in the time or datetime format, then the high-precision time replaces the conversion specification. If the server or adapter does not request high-precision time formatting, the “%.” specification is removed; if no “%.” specification is present, the high-precision time is ignored.

SM_FORMAT_HPTIME controls the formatting of high-precision time. As in the case of SM_FORMAT_TIME, this variable contains characters other than “%” that are copied into the converted results unchanged, and conversion specifications that begin with “%”. The result of a conversion specification always produces a non-negative decimal value less than 1. A single digit that specifies the number of digits to appear after the decimal point may appear between the “%” and the conversion letter; if omitted, 3 digits will appear.

Table 26 lists the high precision time conversion specifications.

CONVERSION SPECIFICATION	DESCRIPTION
%%	Produces a “%”
%s	Produces fractional seconds; the result always has a leading decimal point. For example, with the default precision, a half a second would be represented as “.500”.
%m	Produces fractional milliseconds; the result always has the number of whole milliseconds right-justified with spaces in a 3-character field, followed by a decimal point and the requested number of digits (with trailing zeros). If the precision is 0, the decimal point is omitted. With the default precision, half a second would be represented as “500.000”.
%u	Produces fractional microseconds. This conversion is analogous to that for m except that there are 6 digits before the decimal point. With the default precision, half a second would be represented as “500000.000”.

Table 26: **High Precision Time Conversion Specifications**

In all of the above, a decimal point is taken from the LC_NUMERIC attribute of the current locale. This can be set for languages that, for example, use “,” to separate the integer and decimal portions of a number.

When SM_FORMAT_HPTIME is not set, the format “+%0mms” is used in its place. In the Eastern US time zone, with the US English locale, with all the default values, this might produce a full-precision timestamp such as “28-May-2002 15:30:45+670ms EDT”.

SM_FORMAT_INTERVAL

An interval is a length of time rather than a specific time. SM_FORMAT_INTERVAL controls the formatting of intervals, which are accurate to the nearest second. Like SM_FORMAT_DATETIME, this variable contains characters other than “%” that are copied into the converted results unchanged, and conversion specifications that begin with “%”. A single digit, specifying the number of digits to appear after the decimal point, may appear with “%” and the conversion letter; if omitted, 3 digits will appear.

The precision is only meaningful for the H, M, and S conversions; if specified with any other conversion, it is ignored. Note that with S, the digits after the decimal point are always zero.

Table 27 lists the interval conversion specifications.

Conversion Specification	Description
%%	Produces a “%”.
%d	Produces the number of full days.
%h	Produces the number of full hours, after subtracting out the number of full days. Always two digits.
%H	Produces the number of full hours. This is a decimal value, such as one and a half hours (which with the default precision would appears as “1.500”).
%m	Produces the number of full minutes, after subtracting out the full number of hours. Always two digits.
%M	Produces the number of full minutes. This is a decimal value.
%s	Produces the number of full seconds, after subtracting out the full number of minutes. Always two digits.
%S	Produces the number of full seconds. While this is nominally a decimal value, because interval formatting is applied to values with a resolution of one second, “%S” actually always formats integers.
%.	Defines the position where a high-precision time is to be substituted, if the program requests high-precision formatting.

Table 27: **Interval Conversion Specifications**

A leading "—" indicates a negative interval, which represents a moment of time in the past.

If SM_FORMAT_INTERVAL is not set, the format "%d %h:%m:%s%." is used.

SM_FORMAT_HPINTERVAL

This environment variable controls the format of high precision time intervals. It uses the same conversion specifications as SM_FORMAT_HPTIME; the only difference between these two variables is where they are used. This environment variable substitutes the high precision interval for the "%." conversion specification for a time interval, instead of a time or a datetime.

If SM_FORMAT_HPINTERVAL is not set, the format "%s" is used in its place. With the default settings, an interval of 131072.6 seconds—which works out to 1 day, 12 hours, 24 minutes, and 32.6 seconds—would be represented as "1 12:24:32.600".

Variables That Control How NCM Reads and Writes Files

SM_BACKUP_FILE_LIMIT

NCM programs create a new log file when they are restarted or when the log file is rolled over. This environment variable determines the number of log files that are saved by NCM programs. For more information, including an example, see the *Network Connectivity Monitor System Administration Guide*.

If this environment variable is set to 0, only the current log file and a backup of the previous log file are saved. If this variable is set to an invalid value, a warning message is written to the log file and the default value is used instead. Valid values are 0 through 99, the default value is 3.

SM_WRITEABLE

This environment variable specifies the location where NCM software writes output files such as repositories, log files, and saved consoles. By default the location is set to ***BASEDIR/****smarts/local*. If you set the SM_WRITEABLE variable, you must also create the underlying directories.

Table 28 lists the directories whose location is controlled by SM_WRITEABLE:

SUBDIRECTORY	CONTAINS
/consoles	Saved consoles
/logs	Log files
/repos	Repository files

Table 28: **Directories Controlled by SM_WRITEABLE**

The location where log files are saved can be separately controlled with the SM_LOGFILES environment variable.

Note: Currently, you cannot use the SM_WRITEABLE environment variable to define where the NCM Console writes saved consoles. Saved consoles are always written to the ***BASEDIR****/smarts/local/consoles* directory.

SM_LOGFILES

This environment variable is used to specify the location where log files should be written. Setting this variable is optional; if there is no value assigned to this variable, the log files will be written to the location SM_WRITEABLE*/logs*. If you set SM_LOGFILES, you must also create the */logs* directory at the location specified by the variable.

This variable takes precedence over the value of the SM_WRITEABLE environment variable.

SM_RULESET_PATH

By default, Cisco programs only invoke ASL scripts located in the ***BASEDIR****/smarts/rules* directory, the ***BASEDIR****/smarts/local/rules* directory, or a subdirectory of one of these directories. You can use this environment variable to specify additional locations from which Cisco programs can invoke ASL scripts.

When set, the directories specified by SM_RULESET_PATH are searched before the system defaults.

B

Wildcards Used By NCM Software

This chapter describes the wildcards used by NCM software. These wildcards can be used for pattern matching for the security configuration files and also utilities such as *sm_service*.

A wildcard pattern is a series of characters that are matched against incoming character strings. You can use these patterns when you define pattern matching criteria.

Matching is done strictly from left to right, one character or basic wildcard pattern at a time. Basic wildcard patterns are defined in Table 29. Characters that are not part of match constructs match themselves. The pattern and the incoming string must match completely. For example, the pattern *abcd* does not match the input *abcde* or *abc*.

A compound wildcard pattern consists of one or more basic wildcard patterns separated by ampersand (&) or tilde (~) characters. A compound wildcard pattern is matched by attempting to match each of its component basic wildcard patterns against the entire input string. For compound wildcard patterns, see Table 30.

If the first character of a compound wildcard pattern is an ampersand (&) or tilde (~) character, the compound is interpreted as if an asterisk (*) appeared at the beginning of the pattern. For example, the pattern ~*[0-9]* matches any string not containing any digits. A trailing instance of an ampersand character (&) can only match the empty string. A trailing instance of a tilde character (~) can be read as "except for the empty string."

Note: Spaces are interpreted as characters and are subject to matching even if they are adjacent to operators like "&".

CHARACTER	DESCRIPTION
Note: Spaces specified before or after wildcard operators are interpreted as characters and are subject to matching.	
?	Matches any single character. For example, *server?.cisco.com* matches *server3.cisco.com* and *serverB.cisco.com*, but not *server10.cisco.com.*
*	Matches an arbitrary string of characters. The string can be empty. For example, *server*.cisco.com* matches *server-ny.cisco.com* and *server.cisco.com* (an empty match).
[set]	Matches any single character that appears within [set]; or, if the first character of [set] is (^), any single character that is *not* in the set. A hyphen (-) within [set] indicates a range, so that [*a-d*] is equivalent to [*abcd*]. The character before the hyphen (-) must precede the character after it or the range will be empty. The character (^) in any position except the first, or a hyphen (-) at the first or last position, has no special meaning. Example, *server[789-].cisco.com* matches *server7.cisco.com* through *server9.cisco.com*, but not *server6.cisco.com.* It also matches *server-.cisco.com.* Example: *server[^12].cisco.com* does not match *server1.cisco.com* or *server2.cisco.com*, but will match *server8.cisco.com.*
<n1-n2>	Matches numbers in a given range. Both *n1* and *n2* must be strings of digits, which represent non-negative integer values. The matching characters are a non-empty string of digits whose value, as a non-negative integer, is greater than or equal to *n1* and less than or equal to *n2*. If either end of the range is omitted, no limitation is placed on the accepted number. For example, *98.49.<1-100>.10* matches a range of IP addresses from *98.49.1.10* through *98.49.100.10.* Example of an omitted high end of the range: <*50-*> matches any string of digits with a value greater than or equal to 50. Example of an omitted low end of the range: <*-150*> matches any value between zero and 150. A more subtle example: The pattern <*1-10*>* matches 1, 2, up through 10, with * matching no characters. Similarly, it matches strings like 9x, with * matching the trailing x. However, it does not match 11, because <1-10> always extracts the longest possible string of digits (11) and then matches only if the number it represents is in range.
|	Matches alternatives. For example, "*ab|bc|cd*" without spaces matches exactly the three following strings: "*ab*", "*bc*", and "*cd*". A | as the first or last character of a pattern accepts an empty string as a match. Example with spaces "*ab | bc*" matches the strings "*ab* " and " *bc*".
\	Removes the special status, if any, of the following character. Backslash (\) has no special meaning within a set ([set]) or range (<n1-n2>) construct.

Table 29: **Basic Wildcard Patterns**

Special characters for compound wildcard patterns are summarized below.

CHARACTER	DESCRIPTION
&	"And Also" for a compound wildcard pattern. If a component basic wildcard pattern is preceded by & (or is the first basic wildcard pattern in the compound wildcard pattern), it *must* successfully match. Example: *NY*&*Router* matches all strings which contain NY and also contain Router. Example: <1-100>&*[02468] matches even numbers between 1 and 100 inclusive. The <1-100> component only passes numbers in the correct range and the *[02468] component only passes numbers that end in an even digit. Example: *A*|*B*&*C* matches strings that contain either an A or a B, and also contain a C.
~	"Except" for a compound wildcard pattern (opposite function of &).If a component basic wildcard pattern is preceded by ~, it *must not* match. Example: 10.20.30.*~10.20.30.50 matches all devices on network 10.20.30 except 10.20.30.50. Example: *Router*~*Cisco*&*10.20.30.*~10.20.30.<10-20>* matches a Router, except a Cisco router, with an address on network 10.20.30, except not 10.20.30.10 through 10.20.30.20.

Table 30: **Compound Wildcard Patterns**

Index