EMC® Smarts®

Version 7.0

System Administration Guide

P/N 300-004-101
REV A01

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Copyright ©1996–2006 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com

All other trademarks used herein are the property of their respective owners.

Contents

Preface vii

Chapter 1 Introduction

Performing administration tasks 1-2
- Locating and modifying Smarts files 1-2
- Licensing Smarts software 1-2
- Controlling the start-up of Smarts programs 1-2
- Securing access to Smarts software 1-2
- Operation of the Smarts Broker 1-3
- Managing log files 1-3

Chapter 2 Locating and Modifying Smarts Files

Smarts product suites and directories 2-2
- Common subdirectories 2-3

User-modifiable files 2-4
- Modifying Smarts files 2-5
- Associating Smarts files with sm_edit on Windows 2-6

How Smarts software locates user-modifiable files 2-8
- Controlling the location for rule set files 2-8

Where Smarts software writes output files 2-9
- Controlling where Smarts software writes output files 2-9
- Controlling where Smarts software writes log files 2-10
- sm_logerror 2-10

Using the smgetinfo utility to save modifications 2-11

Chapter 3 License Administration

Before you begin 3-2
- Types of license files 3-2
- Next step 3-3

How to obtain a license file for Smarts software 3-4
- Information needed for a temporary license file 3-4
- How to renew a temporary license file 3-4
- Information needed for a permanent license file 3-4
- How to obtain additional permanent licenses 3-5

How to set up a temporary license file 3-6
- Rules for using a temporary license file 3-6
- Setup procedure for a temporary license file 3-6

Setting up and administering a permanent license 3-8
Components and their roles 3-8
Versions and compatibility 3-10
How products are licensed 3-10
Rules and other considerations 3-11
Setup procedure for a permanent license file 3-13
Adding more permanent licenses 3-17
Upgrading licensing from older versions of Smarts software 3-17
Running older versions of license files in parallel 3-18
Installing and uninstalling the FLEXnet license server 3-20
Installing the FLEXnet license server 3-20
Uninstalling the FLEXnet license server 3-21
Administering license servers 3-22
License administration tools 3-22
Verifying the FLEXnet license server version 3-22
Starting and stopping the license server 3-22
Running the license server on a different port 3-23
Running the license server with a firewall 3-24
Volume licensing for EMC Smarts IP Availability Manager 3-25
Determining the total number of licenses 3-25
Determining the number of checked-out licenses 3-25
Discovery and licenses 3-26
Examples of license files 3-28
Temporary license file example 3-28
Permanent license file example 3-28
Troubleshooting 3-30
Evaluation software no longer runs 3-30
Messages in the flexlm.log file 3-30
Need to determine the host ID for the FLEXnet license server 3-30
Need to determine the FLEXnet license server and all licenses served 3-31
Can't find license error 3-31
Changing an IP address and its effect on licensing 3-31
Error occurs when starting additional Domain Managers 3-31
License file corrupted when attempting to edit license file 3-32
Cannot restart FLEXnet license server after applying a SmartPack 3-32

Chapter 4 Controlling the Startup of Smarts Software

Overview of services 4-2
About the sm_service utility 4-3
sm_serviced and ic-serviced 4-4
The sm_service command line 4-5
Standard options 4-5
sm_service actions 4-5
Installing programs as services with sm_service 4-7
Install options 4-7
Examples of the sm_service install action 4-8
Displaying installed services and their status 4-9
Examples of the sm_service show action 4-9
Starting and stopping services with sm_service 4-10
Start options 4-10
Stop options 4-10
Examples of the sm_service start and stop actions 4-10
Checking whether services are stopped 4-11
Examples of the sm_service isstopped action 4-11

Starting and stopping services from the Windows desktop 4-11
Modifying service parameters with sm_service..... 4-12
Removing services with sm_service 4-13
Example of the sm_service remove action..... 4-13
Default parameters for EMC Smarts services..... 4-14
Common sm_server options 4-14
Common sm_adapter options 4-15
Common sm_beacon options..... 4-15
Common sm_sdi options..... 4-16

Chapter 5 Securing Access to Smarts Software

Smarts security features 5-2
How Smarts security works 5-3
Server and client programs for Smarts software 5-3
Server authentication 5-4
Client authentication..... 5-5
Encryption 5-5
Configuring authentication 5-6
Syntax of the security files..... 5-6
Server authentication records 5-6
Client authentication records..... 5-8
Special authentication values..... 5-9
Default authentication records 5-11
Specifying alternate security configuration files..... 5-14
Controlling authentication 5-16
Setting the SM_AUTHORITY variable..... 5-16
Encryption..... 5-17
Changing the secret..... 5-17
Using the secret for multiple suites and sites 5-18
Locking the secret 5-18
Encrypted passwords..... 5-19
Encrypting passwords 5-19
Encrypted seed files..... 5-21
Encrypted connections..... 5-22
Levels of encryption 5-22
Encrypting connections 5-23
Suggested encrypted connections 5-24
Configuring a secure broker..... 5-25
Additional safeguards..... 5-26
Using file permissions..... 5-26
Limiting access to the configuration files..... 5-26
Limiting access to servers..... 5-26
Example security configurations 5-27
Default security..... 5-27
Customized security 5-28

Chapter 6 Operation of the Smarts Broker

Overview of the Smarts Broker..... 6-2
Viewing the registry of the Smarts Broker 6-3
How the Smarts clients find the broker..... 6-4
How to change the broker environment variable 6-5
Correcting the broker settings 6-5
Disabling the broker..... 6-6

Securing the broker 6-7

Chapter 7 Managing Log Files

Overview of message logging 7-2
Log files 7-5
Starting a new log file 7-6
Controlling the number of saved log files 7-7

Appendix A Environment Variables Used By Smarts Software

How variable values are interpreted A-2
How integer variables are interpreted A-2
How Boolean variables are interpreted A-3
Methods for setting environment variables for Smarts software A-4
Setting a Smarts environment variable suite-wide A-4
Setting a Smarts environment variable for a program A-4
Environment variables for users A-6
Variables that control the Smarts Broker A-9
Variables that control FLEXnet licensing A-10
Variables that control Smarts security A-10
Variables that control date and time formatting A-11
Variables that control how Smarts reads and writes files A-16
Variables that control thread locks A-17

Appendix B Wildcards Used By Smarts Software

Types of wildcard patterns B-1

Index i-1

Preface

The purpose of this guide is to provide information about administering Smarts software. This includes information such as using the proper method to edit Smarts files, locating files, setting up security, managing log files, installing software licenses, and starting and stopping Smarts programs.

Note: This document provides information pertaining to the Smarts software platform. Such information is applicable to all Smarts products that are based on this platform. The version number on the cover of this document reflects the Smarts software platform version, not any specific Smarts product version or suite version. Smarts product version numbers may vary and do not always match the software platform version number.

Audience for this guide

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

Prerequisites

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

EMC Smarts installation directory

In this document, the term BASEDIR represents the location where EMC Smarts 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 EMC Smarts product suite to which the product belongs. This location is referred to as BASEDIR/smarts.

Optionally, you can specify the root of BASEDIR to be something other than:

- UNIX: /opt/InCharge7
- Windows: C:\InCharge7

However, you cannot change the <productsuite> location under the root directory.

For more information about the directory structure of EMC Smarts software, refer to the *EMC Smarts System Administration Guide*.

Related documentation

In addition to this document, EMC Corporation provides a Help system for command line programs as well as product documentation.

Help for command line programs

Descriptions of command line programs 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.

EMC Smarts documentation

Readers of this guide may find the following related documentation helpful. It can be found in the BASEDIR/smarts/doc/pdf directory.

Note: These documents are updated periodically. Electronic versions of the updated manuals are available on the Powerlink website: `http://Powerlink.EMC.com`.

- *EMC Smarts Documentation Catalog*
- *EMC Smarts System Administration Guide*
- *EMC Smarts ICIM Reference*
- *EMC Smarts Common Information Model Infrastructure Models Chart*
- *EMC Smarts Common Information Model Application/Business Models Chart*
- *EMC Smarts ASL Reference Guide*
- *EMC Smarts Perl Reference Guide*

Suggestions for searching PDF files

You may search across multiple PDF files using the Adobe Acrobat Reader software.

1. If the documentation is not accessible to all users of the EMC Smarts product suite, copy the contents of the *BASEDIR/smarts/doc/pdf d*irectory to a central location, such as a shared drive on your LAN, that operators and others may view.
2. To search throughout the documentation library, open the Acrobat Reader software.
   1. Choose **Edit** > **Search**, and enter a word or phrase.
   2. In the **Where would you like to search** option, select **All PDF Documents in** and type the pathname of the location where the PDF documents reside.

If you have more than one EMC Smarts product suite installed, you can set up cross-product document searches by copying files from the BASEDIR/smarts/doc/pdf directory for each product suite into this common documentation directory path.

Conventions used in this guide

EMC uses the following conventions for notes and cautions.

Note: A note presents information that is important, but not hazard-related.

IMPORTANT

An important notice contains information essential to the operation of the software.

CAUTION

A caution contains information essential to avoid data loss or damage to the system or equipment. The caution may apply to hardware or software.

Typographical conventions

EMC uses the following conventions in this guide:

Normal font	In running text: • Interface elements (for example, button names, dialog box names) outside of procedures • Items that the user selects outside of procedures • Java classes and interface names • Names of resources, attributes, pools, Boolean expressions, buttons, DQL statements, keywords, clauses, environment variables, filenames, functions, menu names, utilities • Pathnames, URLs, filenames, directory names, computer names, links, groups, service keys, file systems, environment variables (for example, command line and text), notifications
Bold	In procedures: • Names of dialog boxes, buttons, icons, menus, fields, and maps • Selections from the user interface, including menu items, and field entries • Key names • Window names • Code examples • Commands • User interface tabs
Italic	Used for: • Full publications titles referenced in text In procedures: • Directory paths • Filenames • Scripts • Important/unique words • URLs • Utilities • Variables
`Courier`	Used for: • System output • User entry
< >	Angle brackets used to enclose parameter or variable values supplied by the user.
[ ]	Square brackets used to indicate optional values.
|	Vertical bar used to indicate alternate selections. The bar means “or”.
{ }	Braces used to indicate content that you must specify (that is, x or y or z).
...	Ellipses used to indicate nonessential information omitted from the example.

%	Used to indicate a C shell prompt.
#	Used to indicate a C shell superuser prompt.
▼▲	Used to indicate a command is wrapped over one or more lines. The command must be typed as one line.

Pathname conventions

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

Graphical conventions

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.

Smarts Manager

Unless otherwise specified, the term Smarts Manager is used to refer to EMC Smarts programs such as Domain Managers, Global Managers, and adapters.

Where to get help

EMC support, product, and licensing information can be obtained as follows:

Product information — For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to the EMC Powerlink™ website (registration required) at:

http://Powerlink.EMC.com

Technical support — For technical support, go to EMC Customer Service on Powerlink. To open a service request through Powerlink, you must have a valid support agreement. Please contact your EMC sales representative for details about obtaining a valid support agreement or to answer any questions about your account.

Sales and customer service contacts — For the list of EMC sales locations, please access the EMC home page at:

http://EMC.com/contact

1

Introduction

As a network administrator, you should configure the Smarts software to meet the needs of your operations. To accomplish this, you may need to perform one or more tasks as described in:

- Performing administration tasks 1-2

Performing administration tasks

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

Locating and modifying Smarts files

Some Smarts files can be modified to meet the needs of your network environment. "Locating and Modifying Smarts Files" on page 2-1 describes the directory structure of the Smarts 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 Smarts software (for example, logs, repositories, and saved consoles), and details how to control their location.

Licensing Smarts software

Smarts software requires a license file to function. To evaluate the software, you can obtain a time-limited evaluation license that provides access to specified features. After you purchase the software, you will need to set up a permanent license file. "License Administration" on page 3-1 provides procedures for obtaining both evaluation and permanent license files for your Smarts software. Topics include setting up the license file, starting the FLEXnet license server, and pointing the Smarts software to the license server. It also discusses licensing additional Smarts programs and adding licenses.

Controlling the start-up of Smarts programs

Smarts programs can either be installed as services or as manual processes. "Controlling the Startup of Smarts Software" on page 4-1 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 Smarts software

Security is a critical concern in the world of large-scale distributed networks. Therefore, Smarts security provides a means by which network administrators can set up security to control access to the software. "Securing Access to Smarts Software" on page 5-1 describes how Smarts security works, as well as the authentication processes used by the software. It then provides procedures for setting up security and controlling access to Smarts programs. It also discusses how to set up encrypted connections between Smarts clients and servers.

Operation of the Smarts Broker

Smarts client programs, such as a console or an adapter, use the Smarts Broker to determine where Smarts Managers are running. When a Smarts 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, Smarts clients retrieve this information from the broker so that they can communicate with the Smarts Manager. "Operation of the Smarts Broker" on page 6-1 describes the functions of the Smarts 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, Smarts software can maintain up to 1000 backup log files. You can configure the number of log files retained by Smarts software. "Managing Log Files" on page 7-1 discusses the management of log files, and describes how to "roll over" the files.

2

Locating and Modifying Smarts Files

This chapter describes the directory structure of the Smarts software, files that can be modified, how to locate those files, and procedures for modifying those files. Additionally, the chapter describes the files created by Smarts software (for example, logs, repositories, and saved consoles), and details how to control their location.

- Smarts product suites and directories 2-2
- User-modifiable files 2-4
- How Smarts software locates user-modifiable files 2-8
- Where Smarts software writes output files 2-9
- Using the smgetinfo utility to save modifications 2-11

Smarts product suites and directories

Smarts software is distributed as product suites. A representative list includes:

- EMC® Smarts® IP Management Suite
- EMC Smarts Service Assurance Management Suite
- EMC Smarts Software Development Kit

During the installation process, each suite of products is installed to a separate directory under the *InCharge* root installation directory (users can specify the root directory but not the root subdirectories). In this document, the term *BASEDIR* represents the location where Smarts 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 Smarts software platform version number. The <productsuite> represents the Smarts product suite that the product is part of.

For example, Table 2-1 defines the <productsuite> directory for the EMC Smarts Service Assurance Management Suite.

Table 2-1 Example of product suite directory

Product suite	Includes these products	Directory
EMC Smarts Service Assurance Management Suite	• Service Assurance Manager • Global Console • EMC Smarts Business Dashboard • Business Impact Manager • SAM Failover System • Service Assurance Manager Notification Adapters (Email Notifier Adapter, Log File Notifier Adapter, SNMP Trap Notifier Adapter, Script Notifier Adapter, EMC Smarts Adapter for TIBCO Rendezvous) • Adapter Platform • SNMP Trap Adapter • Syslog Adapter • XML Adapter	/SAM

In accordance with Table 2-1, version 7.0 of Business Impact Manager for UNIX operating systems is, by default, installed to /opt/InCharge7/SAM/smarts. This location is 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/InCharge7 (on UNIX) or C:\InCharge7 (on Windows), but you cannot change the <productsuite> location under the root directory.

Common subdirectories

Table 2-2 lists the subdirectories that are common to all product suites and provides a brief description of their contents.

Table 2-2 Common subdirectories

Installation path	Subdirectory	Description
BASEDIR/smarts	/bin	Smarts software executables for the installed product suite. The files vary by suite.
	/classes	Console-related files such as .jar that are required by Smarts software.
	/conf	Product suite configuration files. The files vary by suite.
	/doc	Documentation for Smarts software and third-party software such as FLEXnet.
	/jre	Java Runtime Environment files shipped with Smarts software and used by the Smarts consoles.
	/lib	Shared libraries required by the Smarts software.
	/local	Top level directory where local copies of Smarts configuration, rules, and script files are created and stored. Also the default location where Smarts software writes files such as repository and log files.
	/perl	Files used for the Perl API.
	/rules	ASL rule sets shipped with Smarts 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	Log files created by the Smarts installation process.

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

User-modifiable files

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

You should never alter Smarts 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. Smarts software is designed to first search for user-modifiable files in BASEDIR/smarts/local or one of its subdirectories. If a modified version of a file is not found in the local area, Smarts software then searches appropriate non-local directories.

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 Smarts files" on page 2-5.

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

Table 2-3 identifies the default subdirectories that include user-modifiable files found under BASEDIR/smarts/local. The subdirectories vary by product suite.

Table 2-3 Subdirectories for user-modifiable files

Installation path	Directory	Description
BASEDIR/smarts/local	/actions	Tool scripts
	/conf	Configuration files
	/data	Import files
	/rules	ASL rules files

Note: The BASEDIR/smarts/local directory has been reserved for directories with local copies of user-modifiable files as well as writable files such as logs and repository files. It typically contains directories listed in Table 2-3. During installation, Smarts software may create empty subdirectories under the BASEDIR/smarts/local directory. These directories are reserved for use for future SmartPacks and Rolling Patches. The exception is the BASEDIR/smarts/local/bin directory that contains files necessary for the successful installation of EMC Smarts on Windows 2003 (Enterprise Edition) operating systems.

In addition, the BASEDIR/smarts/local/conf directory (for each suite) contains the runcmd_env.sh file, which is created during installation. The runcmd_env.sh is a file for setting environment variables for an installed product suite, including the default location for the Smarts Broker. You can edit this file to change the default location of the broker or to define any suite-wide environment variable. For information about setting environment variables, see "Methods for setting environment variables for Smarts software" on page A-4.

The product-specific naming convention used for subdirectories under BASEDIR/smarts/local/conf and BASEDIR/smarts/local/rules (as well as their non-local counterparts) typically reflects the Smarts product or product module within a product suite that uses the files.

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 2-8.

Note: If EMC Corporation 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 Smarts files

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

To modify Smarts files, use sm_edit, a Smarts utility that is installed with each Smarts 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 non-local 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 Smarts file through Windows Explorer, see "Associating Smarts files with sm_edit on Windows" on page 2-6.

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 and writes it to the BASEDIR/smarts/local/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. This ensures that the original version of the file remains unchanged.

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 for users" on page A-6.
- 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 Smarts files with sm_edit on Windows

On Windows systems, you can configure the system so that sm_edit is automatically invoked when a Smarts 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 Smarts 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. Select the checkbox 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, clear the checkbox beside "**Always use this program to open these files**".

How Smarts software locates user-modifiable files

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

Note: BASEDIR/smarts can refer to different product suite directories. Therefore, the configuration file in the above example could be a configuration file for any product suite such as the IP Management Suite or Service Assurance Management Suite.

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, Smarts software first searches for rule set files in that path location or in one of its subdirectories. If the file is not found, Smarts software 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 file.

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

For more information about SM_RULESET_PATH, refer to "Environment Variables Used By Smarts Software" on page A-1.

Where Smarts software writes output files

The BASEDIR/smarts/local directory is also the default location for files written by Smarts software. Writable 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 Smarts software writes log files" on page 2-10. Table 2-4 defines the default subdirectories found under BASEDIR/smarts/local that contain writable files.

Table 2-4 Default subdirectories for Smarts writable files

Installation path	Directory	Description
BASEDIR/smarts/local	/consoles	Smarts consoles that have been saved.
	/logs	Smarts Manager and adapter log files that Smarts software has written. Also where archived notifications are saved. Note that installation and uninstallation logs are written to the BASEDIR/smarts/setup/logs directory.
	/repos	Repository files that Smarts software has written. Unless otherwise specified, the EMC Smarts Beacon saves its repository file to a unique location: BASEDIR/smarts/local/repos/beacon

For Service Assurance Manager, the BASEDIR/smarts/local/logs directory includes a <managername>.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 <managername>-statistics.log. This file contains entries for every invocation of the following operations:

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

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

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

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

Controlling where Smarts software writes output files

Smarts software defines the location of writable 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 Smarts writable 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 Smarts software writes log files

If you want Smarts 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 Smarts Software" on page A-1.

sm_logerror

On Solaris, Linux, and HP-UX systems, every Smarts 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 Smarts 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 Smarts Manager, or checks credentials received by a Smarts Manager. Since the authenticator itself is a Smarts 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. When the Smarts Manager starts as a service, it runs as a daemon after starting sm_logerror and sm_authority. 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 Smarts product suite includes an smgetinfo utility. The utility enables you to maintain a backup copy of all essential files and customizations from your Smarts installation for that product suite. It also enables you to collect data about your current installation so that you can send the data to EMC Global Services for support purposes. The smgetinfo utility is located in the BASEDIR/smarts/script directory.

When run, the smgetinfo utility stores the following Smarts 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 Smarts 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. Note that to run this utility, you need superuser privileges (user ID 0) for UNIX and administrator privileges for Windows.

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

License Administration

This chapter describes license administration and should be used in conjunction with your deployment guide for the EMC Smarts Service Assurance Management Suite and the EMC Smarts IP Management Suite and the *FLEXnet Licensing End User Guide*. (A FLEXnet PDF version of the *FLEXnet Licensing End User Guide* (LicensingEndUserGuide.pdf) is automatically installed with the EMC Smarts documentation into the BASEDIR/smarts/doc/pdf directory.)

- Before you begin 3-2
- How to obtain a license file for Smarts software 3-4
- How to set up a temporary license file 3-6
- Setting up and administering a permanent license 3-8
- Installing and uninstalling the FLEXnet license server 3-20
- Administering license servers 3-22
- Volume licensing for EMC Smarts IP Availability Manager 3-25
- Examples of license files 3-28
- Troubleshooting 3-30

Before you begin

For licensing and license management, Smarts products rely on a third-party licensing software product, FLEXnet, from Macrovision Corporation. The FLEXnet licensing software is installed automatically during the Smarts installation process. You will need the following to complete the licensing process for your Smarts products:

- A license file from EMC Corporation.
- A license server (for permanent license files only). You can use the FLEXnet license server provided with Smarts software or use an existing FLEXnet license server that you already installed for other software applications.

  "Setting up and administering a permanent license" on page 3-8 includes an overall setup procedure for the license file and FLEXnet license server. In addition, this section presents the roles of components in the licensing process, a summary of software versions and compatibility, rules and considerations, and procedures for setting up licensing for an upgrade and for older versions of software.

Without the proper licensing, the Smarts products will not start correctly.

Note: In this discussion, a temporary license file is referred to as *smarts.lic* and a permanent license file is referred to as *<host_id>.smarts.lic.* These are the names of the license files when they are generated and sent to you by the EMC Corporation. If you install a license server with the *install_license* script described later, you should be aware that the script changes the permanent license file name from *<host_id>.smarts.lic* to *smarts.lic.*

Types of license files

Different types of license files for Smarts software exist. They are:

- Temporary (sometimes referred to as *evaluation* or *trial*): A temporary license file is time-limited. It enables you to use specified features for a limited period of time, according to an evaluation agreement. At the end of the trial period, the Smarts software will not start correctly.
- Permanent: A permanent license file is not time-limited. It controls access to applications and features, and controls the number of systems that can be discovered, according to your purchase order (PO).
- Travelling: A travelling license file is used for EMC Smarts Discovery Manager which allows you to deploy the IP Discovery Manager on a laptop. During the information gathering phase of deployment planning, to accurately determine the quantity of network devices, the laptop can be connected to an organization's network and be used to discover elements in the networks.

  See the *EMC Smarts IP Management Suite Deployment Guide* for planning for discovery and the *EMC Smarts IP Management Suite Discovery Guide* for information about the IP Discovery Manager.

Next step

The tasks you need to perform to complete licensing depend upon the type of license file you have and your deployment.

- If you do not have a license file, see "How to obtain a license file for Smarts software" on page 3-4.
- If you already obtained a temporary license file, you need to set up one or more copies of it. A FLEXnet license server is not used with a temporary license file; you do not need to install a license server. See "How to set up a temporary license file" on page 3-6.
- If you already obtained a permanent license file, you need to set up the permanent license file and possibly install a license server. For instructions, see "Setting up and administering a permanent license" on page 3-8.

Note: For the following procedures, you need administrative privileges for both the Smarts software and the systems on which the software is running.

How to obtain a license file for Smarts software

License files are not included with Smarts software.

To obtain a license file, send an email request to smartslicensing@EMC.com or contact your sales representative. Indicate whether you need a temporary (evaluation) or a permanent license file.

- If you request a temporary license file, a sales representative will contact you.
- If you request a permanent license file, EMC Corporation will send you an email with an attached Permanent License Information Request form to complete. The form is also available from your sales representative.

During the process, you will need to complete paperwork and supply information.

Information needed for a temporary license file

For a temporary license file, you need to complete an evaluation request and an evaluation agreement. EMC Corporation will send you an email with an attached temporary license file shortly after the software is shipped.

How to renew a temporary license file

To renew a temporary license file, send an email request to smartslicensing@EMC.com. EMC Corporation will send you an email with a new temporary license file attached.

Information needed for a permanent license file

For a permanent license file, you will need to complete a Permanent License Information Request form. On this form, you need to supply:

- The host ID of the computer where the FLEXnet license server is or will be running (To determine the host ID, see "Obtaining the host ID for a permanent license file" on page 3-4.).
- The host name.
- Smarts products to enable.
- The Smarts software version.
- The expected installation path to BASEDIR/smarts/bin for the Smarts installation from which you want to run the license server.
- The number of managed devices.

After you purchase the software, EMC Corporation will send you an email with an attached permanent license file.

Obtaining the host ID for a permanent license file

For a permanent license, you need to provide the host ID of the system where the FLEXnet license server is or will be running. The value of host ID is a hexadecimal value.

To determine the host ID of the system, use the lmhostid option for the lmutil command. The lmutil command is included with Smarts software.

On UNIX, the command line is:

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

On Windows, it is:

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

The output should resemble the following:

```
# /opt/InCharge7/SAM/smarts/bin/lmutil lmhostid
lmutil - Copyright (c) 1989-2004 by Macrovision Corporation. All
   rights reserved.
The FLEXlm host ID of this machine is "80cfc16b"
```

For best results, send EMC Corporation the exact output of the command by copying it from your screen and pasting it into the Permanent License Information Request form.

How to obtain additional permanent licenses

To add features or increase the number of licenses per feature, send an email request to smartslicensing@EMC.com with a completed Permanent License Information Request form. EMC Corporation will send you an email with a new permanent license file attached.

After you receive the new permanent license file, follow the procedure described in "Adding more permanent licenses" on page 3-17 to force the FLEXnet license server to re-read the license files.

How to set up a temporary license file

Typically, setting up a temporary license file is a task that is performed *after* Smarts software is installed. The following procedure assumes that Smarts software has been installed in a new directory.

Also, note that a FLEXnet license server is not used with a temporary license file; you do not need to install a license server.

Rules for using a temporary license file

Use the temporary license file as you received it from EMC Corporation. Use the attachment *as is*. Do not rename the file. Do not modify the file contents. Do not copy and paste its contents to another file.

A temporary license file is time-limited. It enables you to use specified features for a limited period of time. At the end of the trial period, the Smarts software will no longer start. If you need to, you can renew the trial period. To do so, see "How to renew a temporary license file" on page 3-4.

Setup procedure for a temporary license file

To set up a temporary license file, for each product suite on each system, copy the temporary license file, smarts.lic, to the BASEDIR/smarts/local/conf directory.

For example, if you are evaluating the EMC Smarts IP Management Suite and the EMC Smarts Service Assurance Management Suite, then you need to copy the smarts.lic file to:

- The /opt/InCharge7/IP/smarts/local/conf directory and the /opt/InCharge7/SAM/smarts/local/conf directory for UNIX.
- The C:\InCharge7\IP\smarts\local\conf directory and the C:\InCharge7\SAM\smarts\local\conf directory for Windows.

You do not need to edit the SM_LICENSE variable in the runcmd_env.sh file for each product suite, because by default the variable specifies the full path name to a local copy of the smarts.lic file.

Note: A Global Console does not require a temporary license file.

In Figure 3-1, notice there is no FLEXnet license server or vendor daemon for EMC Smarts, a temporary license file is copied for each product suite, and the SM_LICENSE specifies the full path name (UNIX, in this case) to a local copy of the temporary license file. For an evaluation, a Global Console does not require a temporary license file.

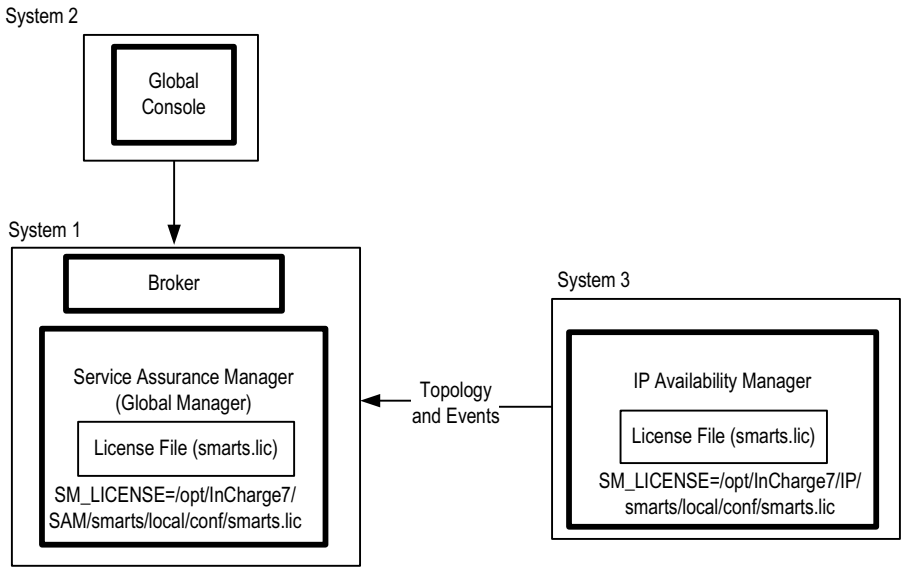


Figure 3-1 Temporary license file setup example

Setting up and administering a permanent license

Before you begin to set up a permanent license file and, if necessary, a license server, you should note that:

- For installation upgrades (such as upgrading entire product suites or adding products to an existing directory), copying a license file to its proper location is performed *before* the upgrade installation.
- For new installations or migrations (that use a new or different directory), setting up a license file is a post-installation task.

See the installation guide that accompanied your product suite for the order of installation tasks. See the migration guide that accompanied your product suite for the license migration procedure.

This section describes the roles of components in the licensing process and provides a summary of software versions and compatibility, rules and considerations such as an existing firewall, an existing FLEXnet license server, or multiple sites, and procedures for setting up licensing for a new installation, adding features, an upgrade, and older versions of software.

Components and their roles

Licensing involves the following components:

- FLEXnet license server—The FLEXnet license server authenticates Smarts products. The FLEXnet license server consists of two types of processes: a license manager daemon and one or more vendor daemons. On Windows, both processes run as services.
  - The license manager daemon receives the initial contact from products and passes the connections to a vendor daemon. It also manages all of the vendor daemons.
  - A vendor daemon is a process that controls how many licenses are in use, as defined in a permanent license file. There is a vendor daemon for each vendor who has a FLEXnet-licensed product on the network.
- Vendor daemon for EMC Smarts—The vendor daemon for EMC Smarts is the process that grants and records the licenses for Smarts software.
- Permanent license file—The permanent license file is generated by EMC Corporation. It controls access to applications and features as well as limits the number of systems discovered by Smarts products.
- Smarts products—These products communicate with the FLEXnet license server to request a license.

  These products also need to communicate with the Smarts Broker. The Broker maintains a registry of Smarts Managers (Global Managers and Domain Managers) and their status. Client applications, such as a console or an adapter, access the Broker to determine where the Smarts Managers are running. EMC Corporation recommends that the FLEXnet license server run on the same host as the Smarts Broker.

Process flow

In Figure 3-2, notice there is a FLEXnet license server, a vendor daemon for EMC Smarts, and a permanent license that resides on the same host as the license server. For all Smarts products, the SM_LICENSE variable specifies the port and host where the FLEXnet license server resides.

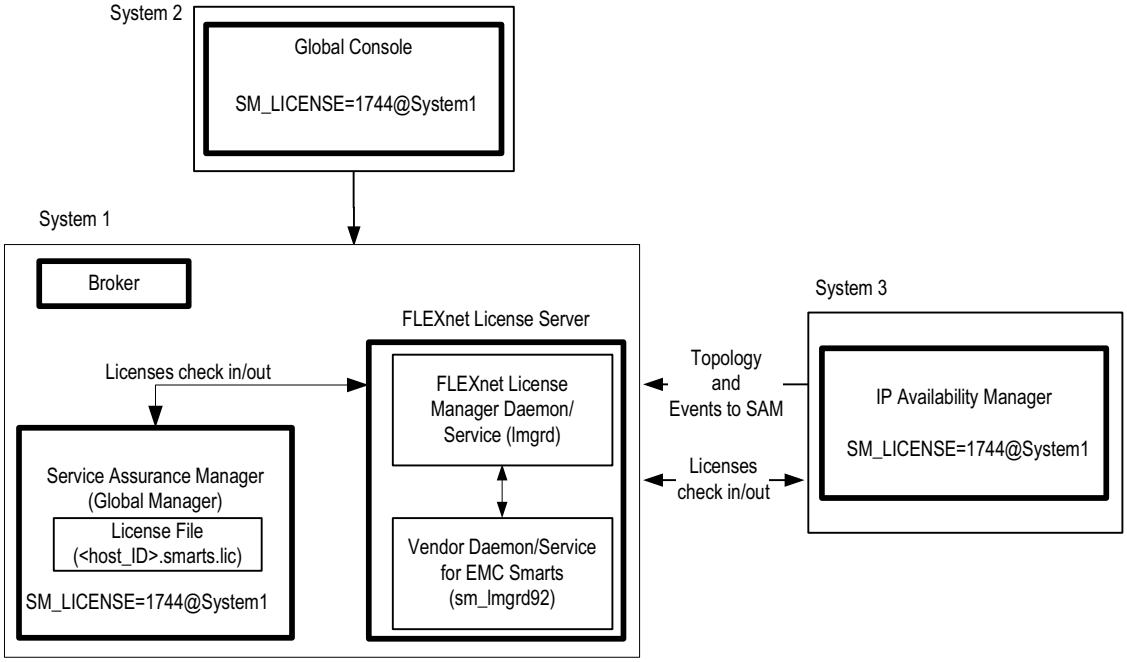


Figure 3-2 Typical licensing process flow

The process flow is as follows:

1. A Smarts product sends a request to the FLEXnet license server for a license to access an application or a feature. In the case of a Global Console, a console relies on its Smarts Manager to send a request for licenses to access console features.
2. The FLEXnet license manager daemon receives the request and passes it to the vendor daemon for EMC Smarts.
3. The vendor daemon for EMC Smarts reads the permanent license file and checks if a license is available.
   - If a license is available, the Smarts product is allowed to use the application or feature.
   - If a license is not available, the vendor daemon for EMC Smarts writes an error message to BASEDIR/smarts/local/logs/flexlm.log and the application or feature does not work. Also, an error message is written to the log file of the Smarts Manager that requested the license (for example, BASEDIR/smarts/local/logs/INCHARGE-SA.log).

Versions and compatibility

Table 3-1 lists recent Smarts software versions, corresponding FLEXnet versions from Macrovision Corporation, and daemon names. FLEXnet software is backward-compatible. For example, a newer license manager daemon (lmgrd) will work with older vendor daemons and older FLEXnet-licensed products. For more information, see the *FLEXnet Licensing End User Guide* (LicensingEndUserGuide.pdf) located in the BASEDIR/smarts/doc/pdf directory.

Note: If you intend to use a permanent license file for Smarts versions 6.0, 5.0, or 5.0.1 with Smarts version 6.2 (or later) software, those license files must be reformatted because of file format changes and FLEXnet encryption changes. Contact EMC Global Services for assistance.

Note: The current naming convention for a permanent license file is *<host_id>.smarts.lic* where <host_id> is the host ID of the system where the FLEXnet license server resides.

Table 3-1 Summary of software versions

License version	Smarts version	License manager daemon name	EMC Smarts vendor daemon name	Smarts permanent license file name
FLEXnet v10.1.3	7.0 (SAM, IP), AI 2.1.1, ACM 2.1.1, Storage Insight 1.0, VoIP 1.1, NPM 2.0	lmgrd	sm_lmgrd92	<host_id>.smarts.lic
FLEXlm v9.2	6.5.1 (SAM, IP, SDK), 6.2 (SAM, IP, SDK), ADM 5, AI 2.0, ATM/FR 2.0, MPLS 1.2, Multicast 1.1, NPM 1.1, OTM 2.0	lmgrd	sm_lmgrd92	<host_id>.smarts.lic
FLEXlm v7.2	6.0 (SAM, IP, ACM 1.1), 5.0 or 5.0.1(SAM), ASM 5.0.x	lmgrd	sm_lmgrd	license.dat

How products are licensed

Smarts software is licensed according to a purchase order (PO) or contractual agreement. The number of available licenses listed in a permanent license file varies accordingly.

A permanent license file defines the number of available licenses for each application or feature, and the number of devices that an EMC Smarts IP Availability Manager can discover. In a permanent license file, each application or feature is listed as one row. Each row includes the number of concurrent licenses. (For a temporary license file, the number of concurrent licenses is unlimited or "uncounted.") For an example of a permanent license file, see "Examples of license files" on page 3-28.

IP Availability Manager uses volume licensing. Volume licensing enables a Smarts product to check out blocks of licenses from the FLEXnet license server. For information, see "Volume licensing for EMC Smarts IP Availability Manager" on page 3-25.

Rules and other considerations

This section describes rules and considerations such as an existing firewall or an existing FLEXnet license server.

Rules for the permanent license file

In most cases, you can use the permanent license file as you received it from EMC Corporation. Do not rename the file or modify the file contents unless you are instructed to do so. Do not copy and paste its contents to another file. For file contents that you are permitted to modify, see one of the following procedures for your deployment and "Examples of license files" on page 3-28.

The current naming convention for a permanent license file is <host_id>.smarts.lic where <host_id> is the host ID of the system where the FLEXnet license server resides.

Rules and considerations for deployment

There are also rules and considerations for deployment:

- EMC Corporation recommends that the FLEXnet license server run on the same host as the Smarts Broker—all Smarts products need to be able to access the license server.
- The permanent license file needs to reside on the same system as the FLEXnet license server.
- If you already use an existing FLEXnet license server for other software applications, see "An existing FLEXnet license server" on page 3-12 for information.
- If a firewall is present, see "Firewalls" on page 3-12.
- If more than one FLEXnet license server is needed, see "Multiple sites" on page 3-13.

Typically, the setup procedure for a permanent license file involves two phases. Assuming that Smarts software has been installed, you need to do the following:

- Phase I—On the system where you wish to install the FLEXnet license server, copy the permanent license file to the proper directory, invoke the install_license script to install the FLEXnet license server, and start the FLEXnet license server.
- Phase II—Wherever Smarts software is installed, modify the SM_LICENSE variable in the runcmd_env.sh file.

Table 3-2 lists licensing scenarios and their corresponding procedures.

Table 3-2 Summary of procedures

If you need to...	Go to:
Set up licensing for a new installation	"Setup procedure for a permanent license file" on page 3-13
Add a product or increase the number of licenses	"Adding more permanent licenses" on page 3-17
Upgrade the license file as part of upgrading the product suite (for example, 6.5.1 from 6.2)	"Upgrading licensing from older versions of Smarts software" on page 3-17
Migrate the license file as part of migrating the product suite (for example, 7.0 from 6.5.1)	See the migration guide that accompanied your product suite for instructions.
Set up a deployment with different versions of Smarts software installed	"Running older versions of license files in parallel" on page 3-18
Run Smarts software version 7.0 and also concurrently run Smarts software version 6.0 (and earlier versions)	"Running older versions of license files in parallel" on page 3-18

Note: For Smarts software version 7.0, if you are migrating from version 6.2 or 6.5.1, you need to install a license file server, but you do not need to obtain a new license file provided that you have not: acquired more features, increased the number of licenses, or changed machines (host IDs).

Note: For Smarts software version 6.5.1, if you are upgrading from version 6.2 or need to run it concurrently with version 6.2, you do not need to change the license file server or obtain a new license file provided that you have not: acquired more features, increased the number of licenses, or changed machines (host IDs).

An existing FLEXnet license server

If you already use an existing FLEXnet license server for other software applications, you must ensure the following and consult your *FLEXnet Licensing End User Guide* for more complete information.

- The FLEXnet version of the license server must be version 10.1.3 in order to manage the vendor daemon for EMC Smarts.
- The permanent license file needs to be copied to the system where the FLEXnet license server resides.
- On other systems wherever Smarts software is installed, modify the SM_LICENSE variable in the runcmd_env.sh file to specify the port and host of the FLEXnet license server.

Firewalls

If a firewall is present, both the FLEXnet license server and the Smarts Broker need to be able to communicate with various Smarts products. This might not be possible if you have a firewall separating them from your other networks.

If there is no access, your firewall administrator might need to:

- Configure security policies (rules) to enable a one-way connection from the FLEXnet license server and the Smarts Broker to the various Smarts products.
- Specify ports while setting up the one-way connection for a greater level of security.

To enable communication through a firewall, you can either specify a specific port for the vendor daemon for EMC Smarts in the license file before you set up the FLEXnet license server as described in Step 3 in "Phase I: Set Up the Permanent License File and the FLEXnet License Server" on page 3-14 or, if the FLEXnet license server is already deployed, use the procedure described in "Running the license server with a firewall" on page 3-24.

The following table offers a listing of common port value defaults.

Table 3-3 Port defaults

Product or component	Default port
Smarts Broker	426
FLEXnet license server (supplied by EMC Corporation)	1744
Smarts Manager (Global Manager or Domain Manager)	one port each, which can be configured.
Smarts adapters, including SNMP Trap Adapter (Receiver) and Syslog Adapter	See the user's guide for your adapter.

Multiple sites

More than one FLEXnet license server may be required depending on your deployment. If there is more than one Smarts Broker for the installation, there may be good reason to use more than one license server.

In general, you should run a FLEXnet license server on each host running a Smarts Broker. For each license server, you should follow the directions for installing a permanent license. (See "Setup procedure for a permanent license file" on page 3-13.)

Each FLEXnet license server will need its own permanent license file. Each permanent 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.

In certain cases, you may use multiple FLEXnet license servers in your network for the same reasons that you split a topology or choose different locations for Smarts Managers: geographical requirements, corporate organizational requirements, or the network's security design. Multiple license servers require a corresponding number of permanent license files.

Also, when multiple FLEXnet license servers are used, each EMC Smarts IP Availability Manager will use blocks of volume licenses from a specific license server. Therefore, the license blocks must be divided among multiple licenses and the appropriate license must be deployed to each license server. Then, IP Availability Managers that rely on a specific license server will have the appropriate volume licenses to support the systems that they must manage. If multiple license servers are used, ensure that EMC Corporation is aware of the configuration details.

Setup procedure for a permanent license file

The following procedure assumes that Smarts software has been installed in a new directory. For Phase I, you will need to set up the permanent license file and the FLEXnet license server. For Phase II, you will need to specify the port and host of the FLEXnet license server in the runcmd_env.sh file of each product suite.

Phase I: Set Up the Permanent License File and the FLEXnet License Server

For Phase I, on the system where the Smarts Broker resides, perform these steps:

1. If you previously used a temporary license (copied *smarts.lic* to the *BASEDIR/smarts/local/conf* directory), you need to delete it, because it will cause the *install_license* script to fail later in Step 4.

2. When you receive the permanent license file, save the attachment to a file named *<host_id>.smarts.lic* in your *BASEDIR* directory on the system where the license server will run.

   Typically, this location should be the same host and *BASEDIR* as for the Smarts Broker. By default, *BASEDIR* is */opt/InCharge7/<productsuite>* for UNIX and *C:\InCharge7\<productsuite>* for Windows.

   WARNING

   Do not save the permanent license file (<host_id>.smarts.lic) to the BASEDIR/smarts/local/conf directory. The install_license script will fail if it finds a permanent license file in this directory. Also, do not save a permanent license file to the BASEDIR/smarts/script or BASEDIR/smarts/conf directory.

   Later, as the result of Step 4, the *install_license* script installs the permanent license file into the *BASEDIR/smarts/local/conf* directory.

3. If necessary, you may need to change the port for the FLEXnet license server (which will be installed in Step 4). For example, if the default port, 1744, is used by another process, the FLEXnet license server can run on a different port. Do not make any changes to the permanent license file unless you need to change the port.

   Use any text editor to open the *<host_ID>.smarts.lic* file. The first line of the permanent license file, labelled SERVER, contains the hostname, host ID, and port of the license server, as shown in the following example:

   ```
   SERVER this_host 8323fcbf 1744
   ```

   Change the port number from the default (“1744”) on the SERVER line and save the file.

   If the FLEXnet license server will be deployed on the opposite side of a firewall from your other Smarts software installations, add the PORT keyword and port number to the second line of the permanent license file, labelled VENDOR, as shown:

   ```
   VENDOR sm_lmgrd92 /opt/InCharge7/SAM/smarts/bin/system/sm_lmgrd92
   PORT=<port_number>
   ```

   Do not remove the 1744 default port value from the SERVER line.

   For more information, see “Running the license server with a firewall” on page 3-24.

4. Install the FLEXnet license server using the *install_license* script. By default, this file is located in the *BASEDIR/smarts/script* directory.

   You must run the *install_license* script on the same system where the license server runs. This script requires superuser privileges (user ID 0) for UNIX, and administrator privileges for Windows. You must specify the full path to the permanent license file.

For example, the command on UNIX systems is:

▼ `# /opt/InCharge7/SAM/smarts/script>install_license.sh install <path to license file>/<host_ID>.smarts.lic` ▲

On Windows systems, for example, the command is:

▼ `C:\InCharge7\SAM\smarts\script>install_license.cmd install <path to license file>\<host_ID>.smarts.lic` ▲

Note: ▼▲ indicates the command must be typed as one line.

In addition to installing the FLEXnet license server, the *install_license* script performs other functions such as configuring the license server to start when the system reboots. For more information, see "Installing and uninstalling the FLEXnet license server" on page 3-20.

5. Start the FLEXnet license server. The FLEXnet license server runs as a daemon on UNIX and as a service on Windows and automatically starts the vendor daemon for EMC Smarts. Issue the command appropriate for your platform. For more information, see "Starting and stopping the license server" on page 3-22.

   Solaris:

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

   HP-UX:

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

   Linux:

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

   Windows:

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

   - Select **Start > Settings > Control Panel > Administrative Tools**.
   - Select **Services**.
   - Select the **FLEXlm License Manager** service.
   - Right-click and choose **Start**.

6. Verify that the FLEXnet license server started by checking your Smarts Manager log file. To do so, start your Smarts Manager and view the Smarts Manager log file for two messages that indicate success:

   - ICF-N-LICENSEDFEATURE-Enabling
   - ICF-N-RESTOREFINISH-PersistenceManager: restore finished

   The log file name is based upon the Smarts Manager's name (for example, *INCHARGE-SA.log*) and the log file is located in the *BASEDIR/smarts/local/logs* directory.

   For instructions on starting your Smarts Manager, see "Starting and stopping services with sm_service" on page 4-10.

Note: On the system where the FLEXnet license server and Smarts Broker reside, for a permanent license file, you do not need to edit the SM_LICENSE variable in the *runcmd_env.sh* file for that product suite, because by default the *install_license* script adds the SM_LICENSE variable with the appropriate values.

However, if other product suites are deployed on the same system, you need to manually edit the SM_LICENSE variable in the *runcmd_env.sh* file for those product suites as described in "Phase II: Ensuring access to the FLEXnet license server using SM_LICENSE" on page 3-16. For example, if the Service Assurance Management Suite and IP Management Suite reside on System1 and the *install_license* script is invoked from the */opt/InCharge7/SAM/smarts/script* directory, the script modifies the variable, but you will have to modify the variable for the IP Management Suite.

Phase II: Ensuring access to the FLEXnet license server using SM_LICENSE

When you install Smarts software on more than one system, you must ensure that the Smarts software on each system can connect to the FLEXnet license server in order to be able to check out licenses. For the system where you installed the FLEXnet license server, the SM_LICENSE variable is added automatically by the install_license script. For other Smarts software installations, you need to modify the variable.

To set up licensing for other product suites or the Global Console, you need to set the SM_LICENSE variable to specify the <port>@<lic_host> of the FLEXnet license server.

Use the sm_edit utility to edit the SM_LICENSE variable in the runcmd_env.sh file. The runcmd_env.sh file resides in the BASEDIR/smarts/local/conf directory of the product suite.

Perform the following steps:

1. For UNIX, invoke *sm_edit* from the *BASEDIR/smarts/bin* directory:

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

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

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

   For example on Windows, type:

   ```
   C:\InCharge7\SAM\smarts\bin>sm_edit conf\runcmd_env.sh
   ```

   For more information about the sm_edit utility, see "Modifying Smarts files" on page 2-5.

2. Set the SM_LICENSE variable to *<port>@<lic_host>* of the FLEXnet license server.

   The *runcmd_env.sh* file should resemble the following:

   ```
   SM_BROKER_DEFAULT=localhost:426
   SM_LICENSE=1744@SYSTEM1
   SM_SNMP_BUG_COMPATIBLE=TRUE
   SM_INCOMING_PROTOCOL=1,0
   SM_OUTGOING_PROTOCOL=1,0
   ```

3. Save the *runcmd_env.sh* file and close it.

4. Any Smarts program within a product suite started after this point will use the applicable environment variables specified in the *runcmd_env.sh* file. Smarts programs that are already running need to be restarted for any new environment variable to take effect. To start services, see "Starting and stopping services with sm_service" on page 4-10.

Adding more permanent licenses

Adding features or increasing the number of licenses with FLEXnet is easy. EMC Corporation will send you an email with the permanent license file attached, as described in "How to obtain additional permanent licenses" on page 3-5. You copy the permanent license file to the BASEDIR on the system with the FLEXnet license server and force the FLEXnet license server to re-read the license files.

To add permanent license files, follow these steps:

1. Copy the new file to the *BASEDIR* directory for the license server.
2. Use any text editor to modify the new file so that the port number and path match those in the original license file.
3. Copy the new file to *BASEDIR/smarts/local/conf*.
4. Force the license server to re-read the license files. Invoke this command from the *BASEDIR/smarts/bin* directory (*BASEDIR\smarts\bin\system* in Windows):

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

   For example, on UNIX, issue a command similar to this:

   ```
   # /opt/InCharge7/SAM/smarts/bin/lmutil lmreread -c 1744@localhost
   ```

   For Windows, issue a command like this:

   ```
   ▼ C:\InCharge7\SAM\smarts\bin\system\lmutil lmreread -c
   1744@localhost ▲
   ```

   Note: ▼▲ indicates the command must be typed as one line.

   You do not need to restart Smarts Managers. There can be multiple permanent license files in the *BASEDIR/smarts/local/conf* directory. The FLEXnet license server will read all files with the .lic suffix.

For more information about the lmreread utility, see the *FLEXnet Licensing End User Guide*.

Upgrading licensing from older versions of Smarts software

This procedure is for upgrades only. For example, if you needed to upgrade from Smarts software version 6.2 to Smarts software version 6.5.1, then you should use this procedure.

If you are migrating the license file as part of migrating the product suite (for example, from 6.5.1 to 7.0), see the migration guide that accompanied your product suite for instructions.

Note: If you have an existing permanent license file for Smarts software version 6.2 and have not acquired more features, increased the number of licenses, or changed machines (host IDs), you do not need to obtain a new license file to upgrade to Smarts software version 6.5.1. Skip Step 1.

To upgrade licensing, perform these steps:

1. Before you upgrade the Smarts software, you must obtain a new permanent license file from EMC Corporation and place it in the *BASEDIR* on the host running the FLEXnet license server.

2. Run the installation program (select **Upgrade** for the installation type). See the installation guide that accompanied your product suite for upgrade instructions.

   During upgrades, where older versions of Smarts software are replaced, the installation program will upgrade the FLEXnet license server automatically.

   The upgrade process determines where the old FLEXnet license server used to run and looks for the permanent license file in that *BASEDIR*.

   - If it does not find the new license file, named *<host_ID>.smarts.lic*, the installation stops.
   - If it does find the new license file, it renames the old license file to *legacy-license.lic* and places a copy of the new license file into the *BASEDIR/smarts/local/conf* directory.
3. You will need to change the port and hostname of each product suite for the SM_LICENSE variable as described in "Phase II: Ensuring access to the FLEXnet license server using SM_LICENSE" on page 3-16 to point to the new FLEXnet license server.

Running older versions of license files in parallel

For some upgrades, installations, and migration scenarios, it is necessary that the FLEXnet license server accept a newer permanent license file, as well as license files for older software. In this instance, a single license server manages both the old and the new permanent license files.

This procedure is for running 6.0 and older versions of license files with Smarts version 7.0 software. If you need to run 6.2 or 6.5.1 license files with Smarts version 7.0 software, use the license migration procedure as described in the migration guide that accompanied your product suite. The 6.2 or 6.5.1 license files do not require modifications like the older license files.

Note: If you intend to use a permanent license file for Smarts versions 6.0, 5.0, or 5.0.1 with Smarts version 6.2 (or later) software, those license files must be reformatted because of file format changes and FLEXnet encryption changes. Contact EMC Global Services for assistance.

To run older versions of license files in parallel, perform these steps:

1. Install the newer software on the same system in a different directory than the older software.
2. If you have not already saved the new *<host_ID>.smarts.lic file* in your *BASEDIR* directory on the system where the FLEXnet license server will run, do so. For information, see "Phase I: Set Up the Permanent License File and the FLEXnet License Server" on page 3-14.
3. If you have not already installed the FLEXnet license server using the *install_license* script, do so. For information, see "Phase I: Set Up the Permanent License File and the FLEXnet License Server" on page 3-14.
4. 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 installation directory, *BASEDIR/smarts/local/conf*.

   For versions 5.0, 5.0.1, and 6.0, the original license file name is *license.dat*.
5. Open the old license file, *legacy_license.lic*, using *sm_edit*. In this case, you need to use the sm_edit utility instead of any text editor, because the sm_edit utility will save the file to the correct */local* directory with the proper permissions.

For example on Windows, you issue:

```
C:\InCharge7\SAM\smarts\bin>sm_edit conf\legacy_license.lic
```

6. In the *legacy_license.lic* file, change the following in the first two lines:
   - On the SERVER line, change **hostname** to "**this_host**".

     Note that "this_host" is a literal term that MUST BE in the license file. Do not put quotation marks around the literal, this_host.
   - On the SERVER line, change the port from **744** to **1744** (or the port for the FLEXnet license server if it is different).
   - On the DAEMON line, change the vendor daemon path from *sm_lmgrd* to the full directory path for the old daemon, *BASEDIR/smarts/bin/system/sm_lmgrd*. For Windows, use *sm_lmgrd.exe*.

     In order for the new FLEXnet license server to find the old daemon process, specify the *BASEDIR* for the location of the old installation. In the following examples, host_ID is the system where the newer FLEXnet license server is running. Do not modify the host ID.

     Here is an example of a 5.0.1 license file:

```
SERVER this_host host_ID 1744
DAEMON sm_lmgrd /opt/InCharge5/smarts/bin/system/sm_lmgrd
```

     Here is an example of a 6.0 license file:

```
SERVER this_host host_ID 1744
DAEMON sm_lmgrd /opt/InCharge6/SAM/smarts/bin/system/sm_lmgrd
```

7. Stop the old license server. Use the procedure in "Starting and stopping the license server" on page 3-22 except substitute **stop** for **start**.
8. Uninstall the old license server using the install_license script. Replace the parameter install with uninstall. See "Uninstalling the FLEXnet license server" on page 3-21 for more information.
9. For the older product suite, modify the following to reflect the port and the system where the newer FLEXnet license server is running. The value for *<license_host>* is the system where the newer FLEXnet license server resides.

   For a 5.x Smarts Manager, modify the *SMARTS.licserv* file. The first line must specify port **1744@<license_host>**.

   For a 6.0 Smarts Manager, modify the SM_LICENSE variable in the *runcmd_env.sh* file to specify port **1744@<license_host>**.
10. Start the new FLEXnet license server. See "Starting and stopping the license server" on page 3-22 for more information.
11. To check that the FLEXnet license server loaded the old and new license files, look at the license server's log file, *BASEDIR/smarts/local/logs/flexlm.log*. The log should not contain errors.
12. Restart the older Smarts Managers. See the documentation that accompanied the older Smarts software.

Installing and uninstalling the FLEXnet license server

This section describes the install_license script and explains how the script installs a FLEXnet license server that is provided with Smarts software. The same script is also used to uninstall the FLEXnet license server. If you already use an existing FLEXnet license server for other software applications, refer to your *FLEXnet Licensing End User Guide* for information.

The install_license script, provided by EMC Corporation, performs several functions:

- Installs a FLEXnet license server, as described in "Phase I: Set Up the Permanent License File and the FLEXnet License Server" on page 3-14.
- Installs the permanent license file into the BASEDIR/smarts/local/conf directory. (Initially, the permanent license file is saved to the BASEDIR directory on the system where the license server will run.)

  Also, when the script installs the permanent license file into the BASEDIR/smarts/local/conf directory, the script changes the permanent license file name from <host_id>.smarts.lic to smarts.lic.
- Adds the SM_LICENSE variable in the runcmd_env.sh file so that the variable specifies the <port>@<lic_host> of the FLEXnet license server.
- Configures the system so that the FLEXnet license server starts automatically when the system boots for UNIX. For Windows, the install_license script creates a service for the FLEXnet license server and the service starts automatically when the system boots.

Note: Do not use the install_license script if you are adding more licenses or features. Instead, you should follow the instructions in "Adding more permanent licenses" on page 3-17.

Installing the FLEXnet license server

By default, the install_license script is located in the BASEDIR/smarts/script directory. Since EMC Corporation recommends that the FLEXnet license server reside on the system where the Smarts Broker is running, you must run the install_license script on the same system where the license server runs.

This script requires superuser privileges (user ID 0) for UNIX, and administrator privileges for Windows. You must specify the full path to the permanent license file.

For example, the command on UNIX systems is:

```
▼ # /opt/InCharge7/SAM/smarts/script>install_license.sh install
  <path to license file>/<host_ID>.smarts.lic ▲
```

On Windows systems, for example, the command is:

```
▼ C:\InCharge7\SAM\smarts\script>install_license.cmd install
  <path to license file>\<host_ID>.smarts.lic ▲
```

Note: ▼▲ indicates the command must be typed as one line.

After you run the install_license script, you need to start the FLEXnet license server. See "Starting and stopping the license server" on page 3-22.

In the licensing process, the FLEXnet license server is typically set up along with the permanent license file, as described in "Phase I: Set Up the Permanent License File and the FLEXnet License Server" on page 3-14.

Uninstalling the FLEXnet license server

When you perform uninstallation, the order in which you uninstall Smarts products is significant. See the installation guide that accompanied your product suite for instructions.

You should also keep the following points in mind:

- When uninstalling all of your Smarts products, you must also uninstall the FLEXnet license server.
- If any Smarts product needs to remain in your network and uses the FLEXnet license server, do not uninstall the FLEXnet license server or the product suite where the permanent license file resides.
- If you uninstall Smarts products without uninstalling the FLEXnet license server, a message displays that the FLEXnet (lmgrd) service is still running.
- A Smarts product will not start if it is unable to contact the FLEXnet license server.

To stop and uninstall the FLEXnet license server, perform these steps:

1. Stop the FLEXnet license server. Use the procedure in "Starting and stopping the license server" on page 3-22 and substitute **stop** for **start**.
2. Execute the *install_license* script with the **uninstall** command from the *BASEDIR/smarts/script* directory.

   For example, the command on UNIX systems is:

   ```
   # /opt/InCharge7/SAM/smarts/script>install_license.sh uninstall
   ```

   On Windows systems, for example, the command is:

   ```
   C:\InCharge7\SAM\smarts\script>install_license.cmd uninstall
   ```

Administering license servers

This section provides information about fundamental administration tasks such as determining the version of your FLEXnet license server and how to start and stop it.

License administration tools

FLEXnet provides several utilities for managing licenses. These include utilities for diagnosing licensing problems, forcing the license server to re-read the license files, and monitoring the status of all licensing activities including feature usage. Instructions on using these utilities can be found in the *FLEXnet Licensing End User Guide*.

The utilities are automatically installed on:

- UNIX, in the BASEDIR/smarts/bin directory.
- Windows, in the BASEDIR\smarts\bin\system directory, and they must be run from that directory.

If you are not familiar with FLEXnet, you may want to read the FLEXnet documentation. A PDF version of the *FLEXnet Licensing End User Guide* is automatically installed with the EMC Smarts documentation in the BASEDIR/smarts/doc/pdf directory. The name of the file is LicensingEndUserGuide.pdf.

Verifying the FLEXnet license server version

To verify the version of the FLEXnet license server, specify the `-v` option to the FLEXnet license server command:

```
# /opt/InCharge7/SAM/smarts/bin/lmgrd -v
lmgrd v10.1.3 - Copyright (c) 1988-2004 by Macrovision Corporation.
   All rights reserved.
```

For a list of Smarts software versions and the corresponding FLEXnet versions, see Table 3-1 in "Versions and compatibility" on page 3-10.

Starting and stopping the license server

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

Run the FLEXnet license server at all times to ensure continuous operation of Smarts software. A Smarts Manager (Global Manager or Domain Manager) 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 FLEXnet license server varies according to the operating system on which it runs. When you invoke the install_license script, this command is added to the proper system-specific directory so that the command is invoked when the system starts and, for Windows, the service starts.

The procedure to manually start and stop the FLEXnet license server is described as follows.

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
```

Linux:

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

Windows:

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

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

Running the license server on a different port

The FLEXnet 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, host ID, and port of the license server, as shown in the following example:

```
SERVER this_host 8323fcbf 1744
```

To run the FLEXnet license server on a different port, perform these steps:

1. Change the default port number ("**1744**") on the SERVER line and save the file.
   - If you just received the permanent license file and you saved it to the *BASEDIR* directory but you have not run the *install_license* script yet, you can use any text editor to open the permanent license file.
   - If you ran the *install_license* script and completed the "Setup procedure for a permanent license file" on page 3-13, you need to use the *sm_edit* utility to open the permanent license file, because the *sm_edit* utility will save the file to the correct */local* directory with the proper permissions.
2. Force the license server to restart. Invoke this command from the *BASEDIR/smarts/bin* directory (*BASEDIR\smarts\bin\system* in Windows):

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

3. All Smarts systems must be configured to use the new port. For instructions, see "Phase II: Ensuring access to the FLEXnet license server using SM_LICENSE" on page 3-16.

Running the license server with a firewall

The FLEXnet license server, which starts on port 1744 by default, starts the vendor daemon for EMC Smarts on a random port. You can specify a specific port for the vendor daemon for EMC Smarts, if a firewall is present.

When a FLEXnet license server is deployed on the opposite side of a firewall from other Smarts software installations, the firewall might block communication to the FLEXnet license server and, in turn, the vendor daemon for EMC Smarts.

To enable communication through a firewall, you can either:

- Specify a specific port for the vendor daemon for EMC Smarts in the license file *before* you set up the FLEXnet license server (see "Phase I: Set Up the Permanent License File and the FLEXnet License Server" on page 3-14).
- Or, if the FLEXnet license server is already deployed, use the following procedure.

If the FLEXnet license server is already deployed, perform these steps to enable communication through a firewall:

1. Add the PORT keyword and port number to the second line of the permanent license file, labelled VENDOR, as shown:

   ```
   SERVER this_host 80cdc5dc 1744
   VENDOR sm_lmgrd92 /opt/InCharge7/SAM/smarts/bin/system/sm_lmgrd92
     PORT=<port_number>
   USE_SERVER
   INCREMENT DMC_CORR sm_lmgrd92 7.0 permanent 1 1454E7768847 \
        DUP_GROUP=SITE SN=OR54127 TS_OK
   ```

   Do not remove the 1744 default port value from the SERVER line.

2. Force the license server to restart. Invoke this command from the *BASEDIR/smarts/bin* directory (*BASEDIR\smarts\bin\system* in Windows):

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

Volume licensing for EMC Smarts IP Availability Manager

The Smarts permanent license controls the number of discovered systems. The FLEXnet license server maintains a pool of licenses that get checked out in blocks to IP Availability Manager products on an as-needed basis. A block consists of 50 devices. Products check out additional licenses one block at a time, as described in "Discovery and licenses" on page 3-26.

For more information about volume licensing, see the *EMC Smarts IP Management Suite Discovery Guide*.

Determining the total number of licenses

Each permanent license file contains a line that defines the number of licensed 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 7.0 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 permanent license file; the result is 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 FLEXnet license server.
- The sm_tpmgr utility shows the number of system licenses checked out and the number used for a single IP 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 FLEXnet 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 for UNIX shows the number of device license blocks checked out from the license server running on port 1744 on the host, host2:

```
▼# /opt/InCharge7/IP/smarts/bin/lmutil lmstat -c 1744@host2
   -f AP_SYSTEM_VOLUME ▲
```

This command returns the following information:

```
lmutil - Copyright (c) 1989-2004 by Macrovision Corporation. All
   rights reserved.
Flexible License Manager status on Mon 10/16/2006 16:59

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

"AP_SYSTEM_VOLUME" v7.0, vendor: sm_lmgrd92
floating license

root host1 /dev/tty (v3.00) (host2/1744 2502), start Mon 10/16 14:19
root host1 /dev/tty (v3.00) (host2/1744 2804), start Mon 10/16 16:06,
   2 licenses
root host2 /dev/tty (v3.00) (host2/1744 1301), start Mon 10/16 12:19,
   2 licenses
```

This output shows that the FLEXnet license server is managing a total of 5 license blocks (or 250 devices). Two IP Availability Managers running on host1 have checked out 3 license blocks (1 license block is not shown). One IP Availability Manager running on host2 has checked out 2 license blocks.

Checked-out licenses and license blocks for a single IP Availability Manager

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

Invoke this command from the BASEDIR/smarts/bin directory:

```
# sm_tpmgr -s <availability_manager> --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 50 system licenses (1 block) have been checked out, 38 of the licenses have been used, 450 system licenses remain (9 blocks), and the current limit of discovered systems is 500 (10 blocks):

```
Total System Volume Licenses CheckedOut:                50
Total Systems in Topology:                              38
Remaining Blocks of System Licenses in License Server: 9
Current System Limit:                                   500
```

Discovery and licenses

Volume licensing applies to EMC Smarts IP Availability Managers and permits the product 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 product 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 IP 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 into the FLEXnet license server when discovery ends.

Limiting discovery

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

You can limit the number of systems automatically discovered by an IP Availability Manager by setting autodiscovery filters. See the *EMC Smarts IP Management Suite 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 a product shuts down with the sm_service or dmquit command, the license blocks checked out to the product are returned to the license pool. If the product does not shut down gracefully, the license blocks return to the license pool, but not as quickly as for a gracefully shut down product.

Examples of license files

EMC Corporation generates license files according to the FLEXnet license file format. This section provides an example of a temporary license file and a permanent license file. For more complete descriptions of file format syntax, see the *FLEXnet Licensing End User Guide*.

Temporary license file example

A temporary license file is not user-modifiable, as described in "Rules for using a temporary license file" on page 3-6.

For illustrative purposes, the following example is an excerpt of a smarts.lic file for Smarts software version 7.0.

```
FEATURE DMC_CORR sm_lmgrd92 7.0 04-dec-2006 uncounted 1C8F62B2E672 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2006
FEATURE AD_PEM sm_lmgrd92 7.0 04-dec-2006 uncounted 53BCCF735983 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2006
FEATURE AP_ICSRV sm_lmgrd92 7.0 04-dec-2006 uncounted FFE23AFEFAF8 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2006
FEATURE DMC_DECS sm_lmgrd92 7.0 04-dec-2006 uncounted 5A1809C9A688 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2006
```

The temporary license file consists of a list of applications or features; each line begins with the FEATURE keyword.

Each FEATURE line includes the feature name, the vendor daemon name (sm_lmgrd92), the feature version, the expiration date, the number of concurrent licenses which is "uncounted" in this case, the authentication signature for the feature, the HOSTID=DEMO value that indicates that the feature is not bound to any particular host, the serial number (SN) to identify the FEATURE line, and a start date.

Permanent license file example

A permanent license file is user-modifiable. In most cases, you do not have to modify the permanent license file. The only values that you can safely change without damaging the licensing process are:

- The port number for the FLEXnet license server.
- The directory path to the vendor daemon for EMC Smarts.
- The keyword PORT=<port_value>. Add this keyword and value pair to the VENDOR line for the vendor daemon for EMC Smarts if a firewall is present (see "Running the license server with a firewall" on page 3-24).

For illustrative purposes, the following example is an excerpt of a <host_ID>.smarts.lic file for Smarts software version 7.0.

```
SERVER this_host 80cdc5dc 1744
VENDOR sm_lmgrd92 /opt/InCharge7/SAM/smarts/bin/system/sm_lmgrd92
USE_SERVER
INCREMENT DMC_CORR sm_lmgrd92 7.0 permanent 1 1454E7768847 \
        DUP_GROUP=SITE SN=OR54127 TS_OK
INCREMENT AP_ICSRV sm_lmgrd92 7.0 permanent 1 9639EDFD001C \
        DUP_GROUP=SITE SN=OR54127 TS_OK
```

The permanent license file consists of information for the FLEXnet license server (SERVER line), the vendor daemon (VENDOR line), and a list of applications or features (FEATURE or INCREMENT line). The USE_SERVER line does not affect the server and is used by the application.

- The SERVER line includes the value “this_host” (which is a required literal, that is, not enclosed in quotation marks), the host ID of the system where the FLEXnet license server resides, and the port number of the FLEXnet license server. The default port number is 1744.
- The VENDOR line includes the name of the vendor daemon for EMC Smarts (sm_lmgrd92) and the directory path to the vendor daemon.
- The USE_SERVER line does not specify any arguments or keywords.
- Each INCREMENT line includes the feature name, the vendor daemon name (sm_lmgrd92), the feature version, the expiration date which is “permanent” in this case, the number of concurrent licenses (1 for example), the authentication signature for the feature, the duplicate grouping that indicates site license sharing, the serial number (SN) to identify the INCREMENT line, and access to the application using a Terminal Server window is allowed.

Troubleshooting

This section describes resolutions to common issues experienced by administrators.

If your issue is not listed in this section, before contacting EMC Global Services, make sure you have the following information available:

- The host ID of the system where the FLEXnet license server is installed.
- The exact error message or a copy of log file. The FLEXnet license server provided by EMC Corporation uses the log file BASEDIR/smarts/local/logs/flexlm.log.
- The log file used by the Smarts Manager. The log file name is based upon the Smarts Manager's name (for example, INCHARGE-SA.log) and the log file is located in the BASEDIR/smarts/local/logs directory.
- For an IP Availability Manager, output of the sm_tpmgr command with the `--sizes` option. For example, see "Checked-out licenses and license blocks for a single IP Availability Manager" on page 3-26.
- Output of the lmstat utility with the `-c` and `-a` options. For example, invoke this command from the BASEDIR/smarts/bin directory (BASEDIR\smarts\bin\system in Windows):

```
# lmutil lmstat -c <port>@<lic_host> -a
```

- A copy of your purchase order (PO)
- The permanent license file

Evaluation software no longer runs

Smarts software installed for an evaluation is no longer operational. Check the expiration date listed in the temporary license file. See "Temporary license file example" on page 3-28.

Messages in the flexlm.log file

The FLEXnet license server provided by EMC Corporation uses the log file BASEDIR/smarts/local/logs/flexlm.log.

Messages such as IN and OUT indicate that a product such as a Global Manager or Domain Manager checked licenses in or out.

Another common message begins with UNSUPPORTED FEATURE, which indicates that the product (Smarts Manager) requested to use a feature that the vendor daemon does not support.

For more explanations of messages, see the *FLEXnet Licensing End User Guide* description of the Debug Log File.

Need to determine the host ID for the FLEXnet license server

To verify the host ID or to complete a Permanent License Information Request form for a permanent license file, you need to determine the host ID of the system where the FLEXnet license server is or will be running. The value of host ID is a hexadecimal value. See the procedure in "Obtaining the host ID for a permanent license file" on page 3-4.

Need to determine the FLEXnet license server and all licenses served

To find out information about your existing licensing deployment, use the lmstat utility with the `-a` option. This displays the name of your FLEXnet license server and its port number, the location of your permanent license file, the status of the vendor daemon for EMC Smarts, and feature usage such as the number of licenses in use.

Invoke this command from the BASEDIR/smarts/bin directory for UNIX and the BASEDIR\smarts\bin\system for Windows:

For example, for UNIX, invoke:

```
# /opt/InCharge7/SAM/smarts/bin/lmutil lmstat -a
```

On Windows, for example, invoke:

```
C:\InCharge7\SAM\smarts\bin\system\lmutil lmstat -a
```

The lmstat utility does not provide information about temporary ("uncounted") licenses unless you perform additional configuration. For information and additional lmstat options, see the *FLEXnet Licensing End User Guide*.

Can't find license error

An error message, Can't Find License, occurs in the Smarts Manager's log file (for example, BASEDIR/smarts/local/logs/INCHARGE-SA.log). Check that the values specified for the SM_LICENSE variable in the runcmd_env.sh file of your Smarts product suite are correct. See "Phase II: Ensuring access to the FLEXnet license server using SM_LICENSE" on page 3-16 for information.

Changing an IP address and its effect on licensing

Changing a system's IP address does not affect permanent or trial licensing. Permanent licensing is only affected if the host ID changes; it is not affected by changing the host name.

The procedure is as follows:

1. Stop all of the EMC Smarts services.
2. Change the IP address of the system.
3. Re-boot and then re-start all EMC Smarts services.

Note: The Host name of the system can change but the host ID must remain the same. If the host ID changes, a new permanent license will be required.

Error occurs when starting additional Domain Managers

The Unable to Obtain License error occurs when you attempt to start a Domain Manager. It occurs when you exceed the number of licenses listed in your permanent license file.

For example, the following error appears in the Domain Manager's log file (for example, BASEDIR/smarts/local/logs/INCHARGE-AM.log).

```
RSRV-E-ELICENSE-Unable to obtain license
 LIC-ECHECKOUT-While attempting to check out license for AP_ICSRV
 LIC-EGENERIC-Licensed number of users already reached
  . Feature:       AP_ICSRV
  . License path:
    1744@localhost
  . FLEXlm error:  -4,132
  . For further information, refer to the FLEXlm End User Manual,
  . available at "www.macrovision.com".
```

In the permanent license file, look for the line AP_ICSRV. It should look similar to this:

```
INCREMENT AP_ICSRV sm_lmgrd92 7.0 permanent 3
```

The 3 means you are only allowed to start three IP Domain Managers such as IP Availability Managers and IP Performance Managers. (See "Permanent license file example" on page 3-28 for information about the INCREMENT line.) In order to run more IP Domain Managers, you will need to contact smartslicensing@EMC.com and your account manager to obtain a new license.

License file corrupted when attempting to edit license file

The Unable to Obtain License error occurs as the result of a corrupted license file. For example, the following error appears in the Domain Manager's log file (for example, BASEDIR/smarts/local/logs/INCHARGE-AM.log):

```
Error Message : RSRV-E-Unable to obtain license  LIC-While attempting
   to check out license for AP_ICSRV (version=3.0)
```

The license file was copied and pasted from an email attachment and then saved to a text file. This process wrapped the individual lines in an unexpected manner and added additional line breaks which corrupted the license file.

Use the trial license file or permanent license file from the email attachment sent by EMC Corporation. Never attempt to edit a license file without the assistance of EMC Global Services.

Cannot restart FLEXnet license server after applying a SmartPack

You shut down all running Smarts processes including the FLEXnet License Manager service before installing a SmartPack and then cannot restart the FLEXnet License Manager service after the upgrade completes.

This problem likely occurred because the <host_ID>.smarts.lic file incorrectly exists in BASEDIR/smarts/conf and in BASEDIR/smarts/local/conf. The SmartPack reconciliation process assumes that the license file in BASEDIR/smarts/conf is to be used so it renames the file in BASEDIR/smarts/local/conf. As a result, the FLEXnet License Manager service cannot be restarted.

To resolve this issue, rename the file in BASEDIR/smarts/local/conf back to <host_ID>.smarts.lic and start the FLEXnet License Manager Service.

If you have not yet installed a SmartPack, use the smgetinfo utility as described in the *EMC Smarts SmartPack Read Me First* document to create a backup file and move the backup file to a temporary directory or archive. Or, you can copy the permanent license file to a temporary directory. After you apply the SmartPack, you can retrieve the permanent license file from the backup file and restart the FLEXnet License Manager service.

4

Controlling the Startup of Smarts Software

This chapter includes the following information:

- Overview of services 4-2
- About the sm_service utility 4-3
- sm_serviced and ic-serviced 4-4
- The sm_service command line 4-5
- Installing programs as services with sm_service 4-7
- Displaying installed services and their status 4-9
- Starting and stopping services with sm_service 4-10
- Modifying service parameters with sm_service 4-12
- Removing services with sm_service 4-13
- Default parameters for EMC Smarts services 4-14

Overview of services

Smarts 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. Smarts 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.

EMC Corporation recommends that you install Smarts 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 Smarts programs that should be installed as services:

- Smarts Broker
- Service Assurance Global Manager
- Service Assurance Adapter Platform (including the SNMP Trap Adapter and Syslog Adapter)
- IP Availability Manager and IP Performance Manager

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 4-10.

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 4-7.

In addition, you can modify a service's settings. For more information, refer to "Modifying service parameters with sm_service" on page 4-12

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 *EMC Smarts 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 Smarts applications (for example, the broker, Smarts Managers, and adapters), and that installs those applications as services.

Additionally, the sm_service utility can start non-Smarts applications.

The implementation of the sm_service utility on:

- UNIX 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.
- Windows involves the Windows Service Manager and sm_service, a command-line tool that communicates user requests to Windows Service Manager. The sm_serviced program is not applicable for Windows platforms.

The sm_service utility is installed with the Smarts software (one per system, on UNIX and Windows), 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 Smarts 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; and /sbin/init.d on HP-UX. The control of services can be affected through *sm_service*. For more information, see "The sm_service command line" on page 4-5.

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 EMC Global Services 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 Smarts 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 4-1 lists the standard options that can be used with the sm_service utility.

Table 4-1 Standard options for sm_service

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. The level can be: None, Emergency, Alert, Fatal, Critical, Error, Warning, Notice, Informational, Debug. Note that Fatal is equivalent to Emergency, Alert, or Critical. Default: Error.
`--errlevel=<level>`	Minimum error printing level. The level can be: None, Emergency, Alert, Fatal, Critical, Error, Warning, Notice, Informational, Debug. Note that Fatal is equivalent to Emergency, Alert, or Critical. Default: Warning.
`--tracelevel=<level>`	Minimum stack trace level. The level can be: None, Emergency, Alert, Fatal, Critical, Error, Warning, Notice, Informational, Debug. Note that Fatal is equivalent to Emergency, Alert, or Critical. Default: Fatal.
`--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.

The following example highlights the use of a standard option.

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

sm_service actions

Table 4-2 lists the actions that can be performed with the sm_service utility.

Table 4-2 sm_service actions

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.
start	Starts one or more installed services.

Table 4-2 sm_service actions (continued)

Actions	Description
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.

The following example highlights an action with several install options.

```
▼# BASEDIR/smarts/bin/sm_service install --startmode=runonce
   --name=ic-broker /opt/InCharge7/SAM/smarts/bin/brstart
   --port=1234 ▲
```

Note: ▼▲ indicates 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>...] ▲
```

Note: ▼▲ indicates the command must be typed as one line.

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
   --name=ic-broker /opt/InCharge7/SAM/smarts/bin/brstart
   --port=1234 ▲
```

Install options

Table 4-3 lists the install options that can be used with the sm_service utility.

Table 4-3 Install options for sm_service

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 sm_service show action is invoked. (See "Displaying installed services and their status" on page 4-9 for information on sm_service show.) On Windows platforms, the description will appear in the first column of the Service Control Manager window (the "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) • disable (cannot be started). Default: runonce

Table 4-3 Install options for sm_service (continued)

Options	Description
`--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. <VALUE> may contain one or more values, separated by commas. 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.

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
--name=ic-broker /opt/InCharge7/SAM/smarts/bin/brstart
--port=1234 ▲
```

Note: ▼▲ indicates the command must be typed as one line.

```
▼# BASEDIR/smarts/bin/sm_service install --startmode=runonce
--env=SM_BROKER=localhost:1234 trapadapter
/opt/InCharge7/SAM/smarts/bin/sm_trapd ▲
```

Displaying installed services and their status

On UNIX systems, the sm_service show action displays previously installed services and their status.

On Windows, the sm_service show action displays the parameters of a service. To display the status of a service on Windows systems, you need to access the Control Panel: Settings > Control Panel > Administrative Tools > Services.

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
```

The output is:

```
RUNNING ic-broker
```

For the command line of a service:

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

The output is:

```
sm_service install --force --unmanaged --startmode=runonce\
 '--name=ic-broker' \
 '--description=EMC Smarts Broker' \
 '--env=SM_CLIENTCONNECT=brokerConnect.conf' \
 '/opt/InCharge7/IP/smarts/bin/brstart' \
 '--port=426' \
 '--restore=/opt/InCharge7/IP/smarts/local/repos/broker/broker.rps' \
 '--output'
```

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 4-4 lists the options for the start action.

Table 4-4 Start options for sm_service

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.

Stop options

Table 4-5 lists the options for the stop action.

Table 4-5 Stop options for sm_service

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.

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 4-6 lists the options for the isstopped action.

Table 4-6 Isstopped options for sm_service

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.

Examples of the sm_service isstopped action

The following provide examples of the sm_service isstopped actions on UNIX. These examples also show the use of wildcards. For more information about wildcard syntax, see "Wildcards Used By Smarts Software" on page B-1.

This checks to see whether all of the services started from the /opt/InCharge7 directory are stopped:

```
sm_service isstopped --pattern '/opt/InCharge7/*'
```

This example checks whether all Smarts Managers 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 **Start > Settings > Control Panel > Administrative Tools**.
2. Select **Services.**
3. Select the EMC Smarts 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 4-7 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 EMC Smarts services

During installation, Smarts 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 Smarts products, and briefly describes them. For examples of the sm_service install command, see the installation guide that accompanied your product suite

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 4-12.

Common sm_server options

Several sm_server options are common to all Smarts Managers that are installed as services (the IP Availability Manager, for example, and the Service Assurance Manager). Table 4-7 lists these sm_server options.

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

Table 4-7 Common sm_server options

Options	Description
`--name=<name>`	The name of the Smarts 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.
`--norestore`	Do not attempt to restore any saved objects.
`--subscribe=<sub>`	Start a subscription adapter that automatically subscribes to the specified notifications.
`--broker=<location>`	The alternate broker location as host:port. Also `-b <location>`.
`--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.

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 SNMP Trap Adapter). Table 4-8 lists these sm_adapter parameters or options.

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

Table 4-8 Common sm_adapter options

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 Smarts Manager.
`--server=<name>`	Connect parser to Smarts Manager.
`--broker=<location>`	The alternate broker location as host:port. Also `-b <location>`.
`--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.

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 options are common to Smarts products that use sm_beacon. Table 4-9 lists these sm_beacon options.

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

Table 4-9 Common sm_beacon options

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>`.
`--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.

Table 4-9 Common sm_beacon options (continued)

Options	Description
`--norestore`	Do not attempt to restore any saved objects.
`--noregister`	Do not register the beacon with the Smarts Broker.

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 4-10 lists these sm_sdi options.

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

Table 4-10 Common sm_sdi options

Options	Description
`--broker=<location>`	The alternate broker location as host:port. Also `-b <location>`.
`--name=<name>`	The name of the SQL Data Interface 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.
`--nonpriv`	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.

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

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

5

Securing Access to Smarts Software

This chapter describes the security features of Smarts. It also provides instructions and examples for securing access to Smarts software.

- Smarts security features 5-2
- How Smarts security works 5-3
- Configuring authentication 5-6
- Controlling authentication 5-16
- Encryption 5-17
- Encrypted passwords 5-19
- Encrypted seed files 5-21
- Encrypted connections 5-22
- Configuring a secure broker 5-25
- Additional safeguards 5-26
- Example security configurations 5-27

Note: Configuring security requires time and effort. Before configuring security, carefully consider your requirements along with the trade-offs between system security, time allocated for system administration, and usability.

Smarts security features

Smarts software provides multiple levels of security which Smarts administrators can use to secure their system and control access to it. The various levels of security can be implemented through:

- Authentication and user privileges

  Client-server connections can be 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. See "Configuring authentication" on page 5-6 for detailed information.

- Encrypted passwords

  Passwords can be encrypted in the serverConnect.conf, clientConnect.conf, and brokerConnect.conf files. Encryption is based on a secret phrase, common to all of the Smarts applications that must interact, and is used to encrypt password fields in the authentication records. See "Encrypted passwords" on page 5-19 for detailed information.

  Passwords can also be encrypted in seed files. See "Encrypted seed files" on page 5-21 for detailed information.

- Encrypted connections

  Smarts software can provide three levels of security above cleartext communication: Diffie Helman-Advanced Encryption Standard (DH-AES), encryption based on the site secret, and DH-AES used in conjunction with the site secret. The levels of security are configured by way of values for environment variables. See "Encrypted connections" on page 5-22 for detailed information.

The Smarts security features are enabled with default settings when you install Smarts software. The default security settings are thoroughly documented and permit access to the system. As a result, you should change the default usernames, passwords, and the secret phrase to enforce access control to Smarts software. In addition, you should restrict access to the security configuration files, as described in "Specifying alternate security configuration files" on page 5-14.

Note: A Smarts administrator can also place access restrictions on certain console operations by applying user profiles. This ability to restrict Smarts users to certain operations is described in the *EMC Smarts Service Assurance Manager Configuration Guide*.

How Smarts security works

Smarts security features apply to authentication and communication connections.

Authentication occurs whenever a client program initiates a connection to a server program. The client passes a Smarts 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.

Secure communication connections are implemented through the use of the Diffie Helman-Advanced Encryption Standard (DH-AES), encryption based on a site secret, or DH-AES used in conjunction with the site secret.

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 username and password to pass on to a server?
- How does encryption apply to authentication as well as communication connections?

Note: For the UNIX account used to run Smarts processes, if the system administrator sets explicit permissions for the file mode creation mask (the umask) for a parent process, Smarts software respects those permissions. For processes started by the Smarts sm_serviced program (described in "sm_serviced and ic-serviced" on page 4-4), the umask permission is inherited.

Server and client programs for Smarts software

Knowing which Smarts programs function as servers and which function as clients helps to understand how to configure security. Simply stated, a client is any Smarts program that initiates a connection to another Smarts program. Programs can act both as a client and as a server.

Table 5-1 lists some server and client programs and shows how they can interact.

Table 5-1 Smarts server and client programs

Server programs	Client programs
Broker	Service Assurance Failover System
	Command-line utilities such as dmctl and brcontrol
	Global Console users
	Global Manager
	Notification Adapters
	Service Assurance Adapter Platform
	Tools
	Report Manager
	Smarts Beacon
	Application Connectivity Monitor

Table 5-1 Smarts server and client programs (continued)

Server programs	Client programs
Global Manager	Service Assurance Failover System
	Broker
	Command-line utilities (dmctl)
	Global Console users
	Notification Adapters
	Tools
	Global Manager
	Report Manager
Service Assurance Adapter Platform	Broker
	Command-line utilities (dmctl)
	Global Manager
	sm_ems
	SNMP Trap Adapter
	Syslog Adapter
IP Availability Manager	Global Console
IP Performance Manager	Global Console
IP Discovery Manager	Global Console
Smarts Beacon	Global Console
Application Connectivity Monitor	Global Console

Note: The Smarts 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.

Note that the server and client programs listed in Table 5-1 can be installed on the same host. For example, it is common for the Smarts Broker and a Service Assurance Manager to run on the same host.

Server authentication

When a client program initiates a connection to a server, it must provide a Smarts username and password. A server authenticates a connection request by comparing the Smarts username and password it receives from a client to the authentication records in its configuration file, serverConnect.conf. The server uses the first authentication record that matches the information sent by the client.

The authentication records are re-read 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 Smarts username and password. Other Smarts applications can prompt for a Smarts username and password or be configured to automatically send the username and password.

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

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 username matches the user running the client and whose target matches the name of the server being connected to. Once it finds a match, the client sends the Smarts username and password to the target server as authentication credentials. If the authentication succeeds, the server tells the client its access privilege. If the authentication fails, the server refuses the connection and the failure is recorded in the server's log file.

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

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

For example, when an EMC Smarts IP Availability Manager registers with a Smarts Broker, the IP Availability Manager sends a Smarts username and password from its clientConnect.conf to the broker. The broker checks the username 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 IP Availability Manager is alive (by pinging the manager), the broker must authenticate with that IP Availability Manager. To do this, it finds a Smarts username and password in its brokerConnect.conf file to send to the IP Availability Manager. The IP Availability Manager checks for the username and password in its serverConnect.conf and either grants or denies the right for the broker to ping it.

Encryption

By default, encryption is enabled during the installation process. All Smarts suites and applications that interact with each other must share a common secret phrase. The security 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 imk.dat file. This file is the basis of encryption for Smarts programs. The programs use the site secret to encrypt passwords in the configuration files and keywords in seed files, as well as to encrypt connections between Smarts programs.

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 specify alternate security configuration files.

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. Note that:

- 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.

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 Smarts username. If that username 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 Smarts Software" on page B-1.

Server authentication records

Server authentication involves setting up authentication records in the serverConnect.conf file. Each record contains information to control access to a particular server, a Smarts username and password, and a connection privilege. To modify this file, use the sm_edit utility as described in "Modifying Smarts files" on page 2-5.

Note: Always use sm_edit utility to edit the serverConnect.conf. This is particularly important if the passwords are encrypted. When you use the sm_edit utility to edit the file, it saves changes to the file in BASEDIR/smarts/local/conf and encrypts the passwords; it does not modify the default, non-encrypted version of the file.

Description of serverConnect.conf

The serverConnect.conf file defines who can connect to which server and what privileges they are granted. By default, separate versions of the file reside 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 format of a record in serverConnect.conf is:

```
<target>:<Smarts username>:<password>:<privilege>
```

Table 5-2 describes the four fields of an authentication record in the serverConnect.conf file.

Table 5-2 Field Descriptions for serverConnect.conf

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.
Smarts username	Smarts 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 username. • <AUTO> is provided for site-specific credentials.
password	Password for the Smarts user requesting a connection.	Can be a password or one of the following special values: • <SYS> indicates that the Smarts 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

Remember that during the authentication process the server receives a connection target, Smarts username, and password from a client. 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.

Connection Privileges

Connection privileges are specified in the serverConnect.conf file. Smarts' authentication feature 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 *EMC Smarts 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, the broker must also support ping (<BROKER>:BrokerNonsecure:Nonsecure:ping).
- *None* prevents access to the server. This privilege can be used to explicitly prevent a Smarts 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. A Smarts 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.

Client authentication records

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

Note: 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>:<Smarts username>:<password>
```

Table 5-3 describes the four fields of an authentication record in the clientConnect.conf and brokerConnect.conf files.

Table 5-3 Fields for clientConnect.conf and brokerConnect.conf

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.
Smarts username	Smarts username that is sent to server for authentication.	Can be a username or one of the following special values: • <USERNAME> indicates that the username under which the current process is logged in as is sent as the Smarts username. • <PROMPT> indicates that the client program asks the user to provide a Smarts 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.

It is important to remember that a Smarts 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 Smarts 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 Smarts username in the clientConnect.conf file or a Smarts username and password will not be sent to a server for authentication. In the clientConnect.conf record, a user's login name and Smarts 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 runs. The operators 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 cleartext 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 username and password and do not use clientConnect.conf. In console applications, passwords display as masked characters (*) to avoid displaying the passwords in cleartext. In order for non-console applications to prompt, the value <PROMPT> must be used for the Smarts username and password in clientConnect.conf.

Network account authentication

For UNIX systems, the system authenticates a username 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 Windows Domain Manager, the format of the username must include the correct Windows domain as well as the username:

```
user_name@domain_name
```

Prompting for authentication

In order for non-console applications to prompt for a username 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 username 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 Smarts 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 Smarts 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 use <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 Smarts software, which may not support authentication, Smarts software provides a "default account" mechanism. When an incoming connection does not provide any authentication information, a server substitutes the standard username and password with <DEFAULT>. After that substitution, the <DEFAULT>/<DEFAULT> authentication information is validated in exactly the same way any other username/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 as well as examples of how automatic authentication can be set up in clientConnect.conf and brokerConnect.conf.

Authenticating Legacy Smarts Software

The Smarts 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 5-11.

```
<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 Smarts software prior to version 5.0.

Automating client authentication to the broker

Smarts programs connect to the Smarts 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 Smarts 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 5-25.

Automating broker authentication to servers

The Smarts 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 an automatic authentication record, the only record in the file.

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

For more information about the Ping access permission, see "Connection Privileges" on page 5-8.

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 Smarts 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: EMC Corporation recommends 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 Manager. You should *never* remove or disable an administrative account in the Global 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 the serverConnect.conf file.

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

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

For more information about the All access permission, see "Connection Privileges" on page 5-8.

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 Smarts username "maint" and the password "maint".

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

The second console record authenticates console users who provide the Smarts username "oper" and the password "oper".

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

Consoles do not use clientConnect.conf for authentication (there is no automatic authentication). The console prompts the user for a username 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 the *EMC Smarts Service Assurance Manager Configuration Guide*.

EMC Corporation recommends 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 username and password. For authentication, the system username becomes the Smarts username. This record provides monitoring privileges.

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

Authenticating users interactively

This authentication record in the clientConnect.conf prompts users to type a Smarts 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 5-13:

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

and the record described in "Authenticating users with a valid system account" on page 5-13:

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

Specifying alternate security configuration 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.

Two environment variables allow you to specify distinct configuration files:

- SM_SERVERCONNECT
- SM_CLIENTCONNECT

For each variable, the absolute path to the alternate security file must be specified. Because you specify the file name, you can have multiple files in the same directory.

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.

For example, the following steps set 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/InCharge7/IP/smarts/bin/sm_service show --cmdline ic-am-server
```

The output is:

```
/opt/InCharge7/IP/smarts/bin/sm_service install
--startmode=runonce ic-am-server
 /opt/InCharge7/IP/smarts/bin/sm_server
     --name=INCHARGE-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. The `--env` option specifies the environment variable.

```
▼ # /opt/InCharge7/IP/smarts/bin/sm_service install --force
  --startmode=runonce
  --env=SM_SERVERCONNECT=/opt/InCharge7/IP/smarts/conf/
     serverConnect_IP.conf
  ic-am-server
  /opt/InCharge7/IP/smarts/bin/sm_server
        --name=INCHARGE-AM
        --config=icf
        --bootstrap=bootstrap-am.conf
        --port=0
        --ignore-restore-errors
        --output ▲
```

Note: ▼▲ indicates the command must be typed as one line.

3. Start, or stop and restart, the program.

For client programs, you can specify an alternate clientConnect.conf file by setting the SM_CLIENTCONNECT variable in the runcmd_env.sh file.

Controlling authentication

The environment variable SM_AUTHORITY controls the authentication security features provided by Smarts software. Beginning with version 5.0 of Smarts software, the authentication features are enabled by default. If necessary, you can set the value of this variable to disable them. This environment variable must be set on each system where Smarts software is running.

The environment variable SM_AUTHORITY can have one of two values:

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

When no value is specified, which is the default, Smarts software version 5.0 and later treats this the same as <STD>.

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

When SM_AUTHORITY is set to <NONE>, Smarts software behaves as if no security features are 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.

Setting the SM_AUTHORITY variable

To add and set the SM_AUTHORITY environment variable in the runcmd_env.sh file in the BASEDIR/smarts/local/conf directory, perform the following steps:

1. Use *sm_edit* to open the *runcmd_env.sh* file.

   To open *runcmd_env.sh* on UNIX, invoke *sm_edit* from the *BASEDIR/smarts/bin* directory:

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

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

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

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

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

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

Encryption

By default, encryption is enabled during the installation process. The basis for encryption is a secret phrase that gets transformed into the file imk.dat. The Smarts programs use this site secret to encrypt passwords in the configuration files and passwords to SNMP v3 devices in seed files, as well as to encrypt connections between Smarts programs.

At installation, encryption is enabled with a default secret phrase. This phrase is:

```
Not a secret (literally)
```

and the imk.dat file can be copied. The imk.dat is found in the BASEDIR/local/conf directory. To raise the level of security, you need to change the secret phrase using the sm_rebond utility. Thereafter, the secret phrase should be changed periodically to maintain a secure system.

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: sm_rebond shuts down all of the processes run from the suite that were started using sm_service or sm_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 and restarted once the utility is finished.

To change the secret, perform the following steps:

1. Use the *sm_rebond* utility.

   On UNIX, invoke the utility from the *BASEDIR/smarts/bin* directory.

   ```
   # sm_rebond --basedir=<base directory>
   ```

   On Windows systems, invoke the utility as follows.

   ```
   C:\>sm_rebond --basedir=<base directory>
   ```

   For the <base directory>, enter the path for the directory in which the Smarts software to be rebonded is installed. For example, on UNIX, it could be /opt/InCharge7/SAM/smarts; on Windows, it could be C:\InCharge7\SAM\smarts.

2. When the utility prompts for the old secret phrase, enter it.
3. When the utility prompts for the new secret phase, enter it. 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.
4. Re-enter the new secret to confirm it.
5. Close the utility.

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 EMC Global Services.

Using the secret for multiple suites and sites

The secret phrase can be used across multiple suites and sites. To use the secret phrase across suites and sites, EMC Corporation recommends that you perform the following steps:

1. Install one or more Smarts suites on one or more hosts.
2. If short periods of interrupted communications are not detrimental, stop the Smarts Managers. However, if the suites also use CLEARTEXT connections, you can use the *sm_rebond* utility without stopping the Smarts Managers. Only that suite will lose connections for a short period of time.
3. Use the *sm_rebond* utility to change the secret phrase for each suite on each host.

   Note: The secret phrase must be exactly the same for each suite on each host in order for the applications to make connections.

4. Use the *sm_edit* utility to update the security configuration files and seed files of the suites as needed.
5. Restart the Smarts Managers if applicable.

Using the secret across large numbers of sites

For large numbers of sites, you can perform the following steps:

1. Install a Smarts suite at the primary site.
2. Use the *sm_rebond* utility to change the secret phrase.
3. Use the *sm_edit* utility to update the security configuration files and seed files of the suite as needed.
4. Copy the *imk.dat* file, as well as the configuration files and seed files to the other installations of the Smarts suite.

   Note: You will NOT have to use the *sm_rebond* utility on the other Smarts Managers if the files are moved to the Smarts Managers before the Smarts Managers are started.

Locking the secret

If access to encrypted data is a critical issue for your organization, EMC Corporation provides the option to lock the imk.dat file. This disables copies of the imk.dat file, even by restoration from a backup process. For more information about this level of security, contact EMC Global Services.

Encrypted passwords

The encryption process uses the secret to encrypt passwords contained in the security configuration files. The first line of the serverConnect.conf, clientConnect.conf, and brokerConnect.conf files indicates the appropriate field to which encryption is applied.

The first line in the file must read as follows; it must begin with a pound sign (#).

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

The value n refers to the position of the password field in the configuration 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.

To disable encryption, add a second pound sign (#) at the beginning of the line.

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

Encrypting passwords

Password is the only field in the security configuration files that is encrypted.

To enter or change a password in a configuration file, perform the following steps. The configuration files are located in the BASEDIR/smarts/local/conf directory of a product suite.

1. Use *sm_edit* to open the file.

   For example, to open the *serverConnect.conf file* on UNIX, invoke *sm_edit* from the *BASEDIR/smarts/bin* directory:

   ```
   % sm_edit conf/serverConnect.conf
   ```

   To open the file on Windows systems, invoke *sm_edit* as follows:

   ```
   C:\>sm_edit conf\serverConnect.conf
   ```

2. Review the contents of the file. For example, the *serverConnect.conf* file may include lines similar to the following:

   ```
   #
   <BROKER>    :<DEFAULT>:<DEFAULT>:All
   InCharge    :admin    :<SYS>    :All
   InCharge-WP :user5    :Password :Ping
   InCharge    :operator :Password :Monitor
   InCharge    :*        :<SYS>    :None
   #
   ```

   Encrypted passwords are preceded by <E-1.0>

3. Delete the current password and replace it with the new password in cleartext. If the current value is encrypted, 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 files are not encrypted.

4. Save the file and close it. When you save and close the file, the appropriate values are encrypted as part of the process.

Repeat the steps for all of the configuration files.

If a configuration file already includes the first line for encryption and you wish to encrypt the file without opening it, specify the sm_edit utility with the **--noedit** option as follows:

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

Encrypted seed files

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 seed file must read as follows; it must begin with a pound sign (#).

```
#<encrypted seed>:1.0:AUTHPASS,PRIVPASS
```

The AUTHPASS and PRIVPASS keywords specify the authentication and privacy passwords for an SNMP v3 device.

To disable encryption, add a second pound sign (#) at the beginning of the line.

If a seed file already includes the first line for encryption and you wish to encrypt the file without opening it, specify the sm_edit utility with the **--noedit** option as follows:

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

Encrypted connections

Smarts 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 cleartext, 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 5-17.

Encrypted connections do not work with the following products:

- Pre-6.2 versions of the software
- Perl API
- EMC Smarts Adapter for NetIQ AppManager

Levels of encryption

Smarts software provides three levels of security above cleartext 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 5-17.

Table 5-4 lists the four types of encryption connections.

Table 5-4 Encryption Levels for 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 cleartext
1	DH-AES	No site secret needed, no configuration (default behavior for new installations), protects against eavesdroppers	Slower connection than cleartext 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 cleartext	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 cleartext or level 2 security, must set site secret and keep it common across all communicating entities

Note: The Global Console supports both cleartext (Level 0) and Diffie Helman-Advanced Encryption Standard (Level 1) encrypted connections.

Encrypting connections

Encrypted connections are configured using two environment variables:

- SM_OUTGOING_PROTOCOL— Controls the connections that a Smarts program acting as a client is allowed to initiate.
- SM_INCOMING_PROTOCOL— Controls the connections that a Smarts 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 you do not set the variable, it is the same as specifying “0”, or CLEARTEXT.

When two or more security levels are supported in common, the largest level number is used. For example, if a client has SM_OUTGOING_PROTOCOL set to "0,1" and the server has SM_INCOMING_PROTOCOL set to "0,1,2", the connection uses the highest common level, level 1.

The order in which the different levels are specified for the environment variables has no significance—for example, "0,1" and "1,0" behave exactly the same.

Applications that do not support encryption are treated as only having a CLEARTEXT level of encryption.

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.

To set the environment variables so that they can be used by the programs of a Smarts product suite, edit 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.

   To open *runcmd_env.sh* on UNIX, invoke *sm_edit* from the *BASEDIR/smarts/bin* directory:

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

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

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

2. Review the contents of the file. The default values for the SM_INCOMING_PROTOCOL and SM_OUTGOING PROTOCOL are 1 and 0.

   ```
   SM_INCOMING_PROTOCOL=1,0
   SM_OUTGOING_PROTOCOL=1,0
   ```

3. Update the values for the variables to meet the needs of your system.

   For example, to raise security to the next highest level, change **1** to **2**:

   ```
   SM_INCOMING_PROTOCOL=2,0
   SM_OUTGOING_PROTOCOL=2,0
   ```

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

Suggested encrypted connections

EMC Corporation recommends that you configure your system to use encrypted connections wherever possible.

- Brokers should be able to communicate with cleartext as well as encryption in both SM_INCOMING_PROTOCOL and SM_OUTGOING_PROTOCOL only if they need to monitor the status of pre-version 6.2 software, such as Smarts 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. Brokers that only support version 6.2 software or later do not need to support cleartext if all clients can make encrypted connections.
- If a server must connect to a pre-version 6.2 application (for example, if a version 7.0 Service Assurance Manager must connect to a version 6.0 Domain Manager) then set SM_OUTGOING_PROTOCOL to cleartext 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 6.2 Smarts software, add the cleartext option to the appropriate variable.
- Configure any components (6.2 software or later) 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 6.2 software.

Configuring a secure broker

You can configure the Smarts Broker to run in a secure manner. Using a secure broker results in the following changes to how Smarts 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 Smarts username and password for the secure broker credentials. The new Smarts 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 Smarts username "SecureBroker" and the password "Secure". EMC Corporation recommends that you choose a unique Smarts 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 Smarts 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
     ```

Additional safeguards

In addition to encrypted passwords and encrypted connections, you can secure your system by using the file permissions of the operating system, by limiting access to the security configuration files, and by limiting access to the servers.

Using file permissions

You can limit access to the security configuration files by using the permissions feature of the operating system. Only select users should be allowed access to the files. The users must include those who launch the Smarts applications requiring access to the authentication records.

Limiting access to the configuration files

Prior to version 6.2, the security features used by Smarts software required that Smarts usernames and passwords appear in cleartext in the 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 configuration files installed with Smarts software, which should be edited after installation, are readable by anyone.

Because Smarts Managers and the Smarts Broker typically run as root on UNIX systems and as Local Administrator on Windows systems, the serverConnect.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 Smarts Managers. 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.

Limiting access to servers

The `--accept` option to the sm_server command, described in "SM_MAIN_OPTIONS" on page A-7, 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 Smarts installations to use a variety of levels of security. Higher levels of security require more administrative effort, as well as extra effort on the part of users of the system, but give you better control of the use of Smarts software. The appropriate level of security for your site is determined by the policies and procedures of your organization.

This section describes two levels of security, and describes their advantages and disadvantages. You can choose one of these levels, or combine characteristics of several.

Default security

When initially installed, default security uses the security configuration files with encrypted passwords based on the default secret phrase; the connection protocols are set to use Diffie Helman encryption and cleartext. With the default security, you need to change the passwords for the Smarts usernames in the security configuration files. Default security 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 the usernames before using the system in a production environment.

While the default security provides basic authentication, it does not provide much manageability or security. The oper/oper credentials must be known to all console operators; the admin/changeme credentials must be known 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 Smarts 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 can 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 gets a unique username and password. Accesses to the system can be traced to a particular user, and access can be individually revoked. Specifying <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
```

Customized security

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

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

1. Create a new secret phrase using *sm_rebond*. For more information see "Changing the secret" on page 5-17.
2. Change the username and passwords in *serverConnect.conf*, *clientConnect.conf*, and *brokerConnect.conf*.

   These should provide access control by individual users. Consider using system authentication to authenticate Global Console users as described in "System authentication" on page 5-10. The new usernames and passwords should also address connections between servers and servers or servers and clients.

   If your Smarts 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. 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 Smarts components to communicate with encrypted connections. The types of allowed connections depend on the versions of Smarts software in use.
   - If your site uses only version 6.2 or later software, most connections should be encrypted. The broker does not need to support cleartext if all clients can make encrypted connections.
   - If your site uses version 6.2 with older versions, both SM_OUTGOING_PROTOCOL and SM_INCOMING_PROTOCOL values should include cleartext as well as encryption.

     For more information, see "Encrypted connections" on page 5-22.
4. At the highest security level, you should run with a secure broker as well. Setting up a secure broker is described in "Configuring a secure broker" on page 5-25.

6

Operation of the Smarts Broker

This chapter includes the following information:

- Overview of the Smarts Broker 6-2
- Viewing the registry of the Smarts Broker 6-3
- How the Smarts clients find the broker 6-4
- How to change the broker environment variable 6-5
- Securing the broker 6-7

Overview of the Smarts Broker

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

The broker registry maintains the following information:

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

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

  - *Running* indicates that the broker is able to communicate with the Smarts Manager.
  - *Dead* indicates that the Smarts Manager exited unexpectedly or is unreachable. When a Smarts Manager 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 Smarts Managers to determine their state.
- The process ID of each Smarts Manager. 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 Smarts Manager is set to zero to indicate the broker does not know the process ID of the Smarts Manager.
- The last time the state of the Smarts Manager changed. This value is set when the Smarts Manager registers with the broker and is updated if the broker determines that the Smarts Manager is dead. When the broker restarts, it changes the status of Smarts Managers marked Unknown to Running or Dead.

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

Viewing the registry of the Smarts Broker

The broker is a component of a Smarts 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 [-b host:port]
```

If the broker resides on another system, specify the `-b` option with the host and port number.

How the Smarts clients find the broker

The Smarts Broker facilitates communication between Smarts clients and Smarts Managers. 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 Smarts programs to find the broker. The default host setting for the variable is *localhost*, and the default port setting is 426.

A Smarts 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 Smarts software is installed.

- If during installation, the default broker host and/or port settings were changed, but the entered values are incorrect.
- If the broker is automatically installed when you install a Smarts 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 Smarts application is installed on one host and the Smarts 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.

1. 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 Smarts programs and any subsequent installations for this host.

2. 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
   ```

   The output is:

   ```
   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 ▲
   ```

Note: ▼▲ indicates the command must be typed as one line.

Disabling the broker

In the second scenario, you must change the SM_BROKER_DEFAULT host variable in the runcmd_env.sh file and change the **--startmode** value.

1. 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 Smarts Broker and <value> represents the port of the broker.

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

2. 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
   ```

   The output is:

   ```
   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 ▲
   ```

   Note: ▼▲ indicates the command must be typed as one line.

Securing the broker

Securing access prevents unauthorized usage of the broker, and protects it from being modified (for example, Smarts Managers 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 5-25.

7

Managing Log Files

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

- Overview of message logging 7-2
- Log files 7-5
- Starting a new log file 7-6
- Controlling the number of saved log files 7-7

Overview of message logging

As depicted in Figure 7-1, a Smarts application writes messages to either standard output (stdout) or to standard error (stderr), or sends messages to the Smarts exception logging mechanism for severity level checking and filtering. Standard output is intended for writing conventional output, and standard error is intended for writing diagnostic output.

The Smarts exception logging mechanism prints messages to the following receivers:

- Smarts log file (via redirected stderr)

  A Smarts log file is created in the BASEDIR/smarts/local/logs directory for each Smarts application started with the `--output` option. Any Smarts application installed as a service automatically has `--output` in its invocation.

- System error logger

  This logger is the syslog mechanism on UNIX systems and the Event Logger on Windows systems.

Smarts messages are controlled by the following standard options for the sm_service command and environment variables.

For the sm_service options that use a minimum security level, the level can be: None, Emergency, Alert, Fatal, Critical, Error, Warning, Notice, Informational, and Debug. Note that Fatal is equivalent to Emergency, Alert, or Critical. See the top of Figure 7-1 for the range of security levels.

- Option `--output[=<file>]`

  Used to capture and redirect stdout and stderr output for a Smarts application to a Smarts log file in the BASEDIR/smarts/local/logs directory. The name of the log file is <file>.log, or the `--logname` value if <file> is omitted. If `--output` is not specified in the invocation line, the application's stdout and stderr output prints by default to the console screen.

- Option `--errlevel=<min_severity_level>`

  Used by the Smarts exception logging mechanism to determine the minimum exception level for writing messages to stderr. By default, `--errlevel` is set to Warning. Exceptions below this level are not written to stderr.

- Option `--loglevel=<min_severity_level>`

  Used by the Smarts exception logging mechanism to determine the minimum exception level for sending messages to the system error logger. By default, `--loglevel` is set to Error. Exceptions below this level are not forwarded to the system error logger.

- Option `--logname=<name>`

  Used to specify the name of the Smarts log file and the name of the sending application in messages destined for the system error logger. The default is the name of the application specified by the `--name` option (or just `-n`).

- Option `--tracelevel=<min_severity_level>`

  Used to print a stack trace to the Smarts log file when an exception at this level or above occurs. By default, `--tracelevel` is set to Fatal. Exceptions below this level do not write a stack trace.

- Environment variable SM_MAIN_OPTIONS

  Used to set the defaults for the standard options for all Smarts applications. For example,

  ```
  SM_MAIN_OPTIONS=--loglevel=None
  ```

  sets the default exception level to None for the system error logger, meaning that, by default, the Smarts exception logging mechanism will not send any messages to the system error logger.

  You can specify a space-separated list of standard options for SM_MAIN_OPTIONS, exactly as the options would appear on the invocation line. Of course, when starting a Smarts application, you can override a globally specified default by including the appropriate option and new value in the command line.

- Environment variable SM_LOGFILES

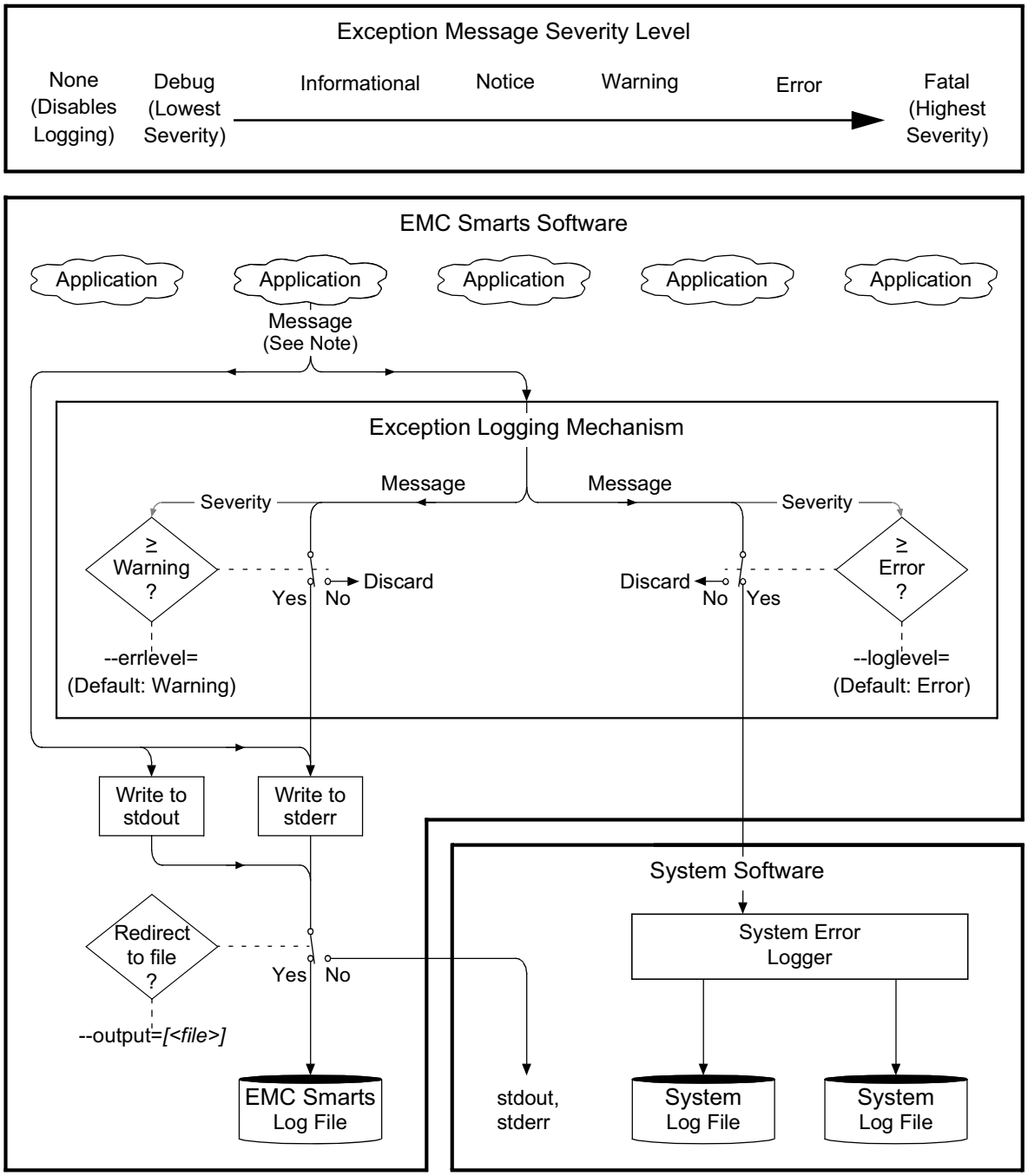
  Used to change the default log location for Smarts log files from BASEDIR/smarts/local/logs to a user-specified location. For example, on a UNIX system,

  ```
  SM_LOGFILES=/u/ps/johndoe/ic/logs
  ```

  results in the writing of the Smarts log files to the /u/ps/johndoe/ic/logs directory.

For information about sm_service command options and environment variables, see "The sm_service command line" on page 4-5 and "Environment Variables Used By Smarts Software" on page A-1, respectively.

Note: In Figure 7-1, the levels Emergency, Alert, and Critical are not shown; Fatal is equivalent to Emergency, Alert, or Critical.



Note: An EMC Smarts application writes a message to stdout *or* to stderr *or* to the exception logging mechanism.

Figure 7-1 Message logging

Log files

When the `--output` option is specified for the sm_service command, each Smarts Manager stores its working history in a log file. Every time a Smarts Manager starts, when errors occur, or when a connection is lost, the Smarts Manager writes information to the log file.

Log files are, by default, written to the BASEDIR/smarts/local/logs directory. The SM_WRITEABLE environment variable specifies the default locations for log files and other output files such as repositories and saved consoles. For more information about SM_WRITEABLE, see "SM_WRITEABLE" on page A-16.

Also, the location where log files are saved can be separately controlled with the SM_LOGFILES environment variable. See "SM_LOGFILES" on page A-16 for information.

All log files have a .log file type. The default name of a log file is based on the Smarts Manager's name. For example, if the name of the Smarts Manager is INCHARGE, the log file is named INCHARGE.log. To change the default name, specify a file name with the `--output` option for the sm_service command. The names of log files written by adapters are described in their respective user's guides.

Smarts 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 Smarts Manager starts up, it renames a file that matches its log file name and adds a .bak suffix to the name (for example, InCharge.log to InCharge.log.bak). If a file with this name already exists, it is renamed InCharge.log.NNN, and a new InCharge.log.bak is created. By default, Smarts programs save two log files, which does not include the active log file.

Starting a new log file

You can request that a Smarts Manager or adapter create a new log file, often referred to as *rolling over* a log file. Typically, this is done when a log file becomes quite large. Log files grow indefinitely, though slowly, under normal conditions.

Most log files are small and rolling over a log file is not necessary. However, if you need to start a new log file, use the following methods:

- For UNIX systems only, use the kill command.
- For UNIX and Windows systems, use the roll_log command.

You can repeat the log file roll-over process as many times as you like. If you need to retain log files beyond the limit specified by the SM_BACKUP_FILE_LIMIT environment variable, you should copy or at least rename the .log.bak files as soon as they are created.

The kill command method for UNIX systems

On UNIX systems only, it is possible to request that a Smarts Manager or an adapter process "roll over" its log files. You do this by sending a SIGUSR1 signal to the process ID writing the log file. Use the kill command of your shell:

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

Upon receipt of the USR1 signal, a Smarts Manager or adapter process first writes an informational message to the end of the current log file, repeats the steps it executed at startup, and creates a new log file and a backup log file. All subsequent logging information is recorded to the new file.

Note: To determine the process number <pid>, you can use the brcontrol utility or for an adapter, use the system's ps command.

The roll_log command method

The roll_log command is invoked through the dmctl utility, which requires that you attach to a Smarts Manager or adapter with administrative privileges. The syntax of the command is:

```
roll_log [<file-name>]
```

The <file-name> option enables you to specify a new name for current and subsequent 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 name 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 of the sm_server command.

For example, on Windows to start a new log file for a Smarts Manager named InCharge, issue:

```
C:\InCharge7\SAM\smarts\bin\dmctl -s InCharge exec roll_log
```

On UNIX systems, the equivalent command is:

```
% /opt/InCharge7/SAM/smarts/bin/dmctl -s InCharge exec roll_log
```

When roll_log is invoked, the Smarts Manager or adapter writes an informational message to the end of the current log file, repeats the steps it executed at startup, and creates a new log file and a backup log file. All subsequent logging information is recorded to the new file.

Controlling the number of saved log files

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

A

Environment Variables Used By Smarts Software

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

- How variable values are interpreted A-2
- Methods for setting environment variables for Smarts software A-4
- Environment variables for users A-6

How variable values are interpreted

This section describes how Smarts 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, Smarts software 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 A-1 provides examples of numeric values and how they are interpreted by Smarts software.

Table A-1 Examples of numeric values

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.

How Boolean variables are interpreted

When an environment variable is expressed as a Boolean, Smarts software 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 Smarts 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 Smarts product suite and the other sets the environment variable for a particular program.

Setting a Smarts environment variable suite-wide

When necessary, you can set an environment variable so that a Smarts 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 Smarts 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:

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

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

Setting a Smarts 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 Smarts Manager 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/InCharge7/IP/smarts/bin/sm_service show --cmdline
   ic-am-server ▲
   ```

The output is:

```
/opt/InCharge7/IP/smarts/bin/sm_service install
--startmode=runonce ic-am-server
 /opt/InCharge7/IP/smarts/bin/sm_server
     --name=INCHARGE-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. The **`--env`** option specifies the environment variable.

```
▼# /opt/InCharge7/IP/smarts/bin/sm_service install --force
--startmode=runonce
--env=SM_SERVERCONNECT=/opt/InCharge7/IP/smarts/conf
/serverConnect_IP.conf
ic-am-server
 /opt/InCharge7/IP/smarts/bin/sm_server
     --name=INCHARGE-AM
     --config=icf
     --bootstrap=bootstrap-am.conf
     --port=0
     --ignore-restore-errors
     --output ▲
```

Note: ▼▲ indicates the command must be typed as one line.

3. Start, or stop and restart, the program.

Environment variables for users

The following list of environment variables can be applied to Smarts 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_DISABLE_KEEPALIVES
- 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_COUNT_REPOS_LOCKS
- SM_REPOS_LOCKS_LIMIT

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) 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 Window System 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 EMC Smarts 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 Smarts software. You can override the values specified by SM_MAIN_OPTIONS by providing alternative values on the command line.

Table A-2 Standard options supported by SM_MAIN_OPTIONS

Standard option	Definition
`--accept=<hosts>`	Specify a list of hosts, by name or IP address, from which Smarts programs accept connections. Multiple entries should be separated by commas. By default, Smarts programs accept connections from any host. Smarts 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 Smarts software is installed, specify the name or IP address of the host instead of localhost.
`--daemon`	Run the Smarts 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, Fatal, Critical, Error, Warning, Notice, Informational, Debug. The default is Warning. (Note that Fatal is equivalent to Emergency, Alert, or Critical.)
`--loglevel=<level>`	Specify the minimum level at which event messages are written to the system logging facility. Possible values include: None, Emergency, Alert, Fatal, Critical, Error, Warning, Notice, Informational, Debug. The default is Error. (Note that Fatal is equivalent to Emergency, Alert, or Critical.)

Table A-2 Standard options supported by SM_MAIN_OPTIONS (continued)

Standard option	Definition
`--logname=<name>`	Specify the name used to identify the Smarts 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, Fatal, Critical, Error, Warning, Notice, Informational, Debug. The default is Fatal. (Note that Fatal is equivalent to Emergency, Alert, or Critical.)
`--useif=<IP address>`	Specifies the IP address that the Smarts 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 Smarts program.

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

SM_SNMP_BUG_COMPATIBLE

Smarts implementation of the SNMP protocol is strict in that Smarts 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 Smarts programs may receive non-conforming SNMP messages. In some cases, Smarts 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, Smarts software only accepts conforming SNMP messages. You can, however, set the SM_SNMP_BUG_COMPATIBLE environment variable, so that Smarts software accepts the non-conforming SNMP messages described above.

When set, any invalid SNMP message that is accepted by Smarts 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 set to 1, it enables the use of the SM_SNMP_Actions::Set( ) method. When it is disabled (set to 0), the method cannot be used to set the value of an OID.

SM_OUTGOING_PROTOCOL

This environment variable controls the kinds of connections that a Smarts 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". Smarts software releases prior to Smarts 6.2 cannot use encrypted connections.

SM_INCOMING_PROTOCOL

This environment variable controls the kinds of connections that a Smarts 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". Smarts software releases prior to Smarts 6.2 cannot use encrypted connections.

SM_DISABLE_KEEPALIVES

This environment variable controls KeepAlive messages for both Smarts Managers and clients. To disable KeepAlive messages, specify a non-zero number. To enable the messages, specify a zero (0).

Smarts software uses KeepAlive messages to test TCP connections between Smarts components. If a KeepAlive message is not received, a message is written to the Smarts Manager's log file. These logged messages may indicate that failed KeepAlive messages may be the result of other software applications and hardware not supporting TCP urgent data, and not the result of the connection being down.

To control KeepAlive messages for only the Global Console, specify the Java property com.smarts.disable_keepalives as described in the *EMC Smarts Service Assurance Manager Operator's Guide*.

Variables that control the Smarts Broker

SM_BROKER_DEFAULT

This environment variable specifies the default location of the broker. When you install Smarts 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 Smarts software is installed. In addition, you may have to edit the broker startup script.

Smarts software follows these steps 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 FLEXnet licensing

LM_LICENSE_FILE

The LM_LICENSE_FILE environment variable is used by the FLEXnet 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 FLEXnet 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 FLEXnet 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 the LM_LICENSE_FILE and SM_LMGRD_LICENSE_FILE environment variables, see the *FLEXnet Licensing End User Guide*, which is automatically installed with the EMC Smarts documentation into the BASEDIR/smarts/doc/pdf directory. The name of the file is LicensingEndUserGuide.pdf.

Variables that control Smarts security

SM_AUTHORITY

This environment variable determines the selection of the authority programs used by the Smarts software for different parts of the authentication process. 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=someAuthority
  ```

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

Smarts Managers 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 Smarts 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.

TZ

This system environment variable determines what time zone is used when formatting a time. As such, it controls 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, Smarts Managers 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 Smarts 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 A-3 lists common date conversion specifications.

Table A-3 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.

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 A-4 lists common time conversion specifications.

Table A-4 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

Smarts software also supports the special conversion specification “%.”, which is described in the section “SM_FORMAT_HPTIME” on page A-14.

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 Smarts Managers 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, Smarts software provides a mechanism for displaying times with a greater precision to the nearest nanosecond, with the actual resolution being about a millisecond.

Smarts Managers 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 Smarts Manager 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, three digits will appear.

Table A-5 lists the high precision time conversion specifications.

Table A-5 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".

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, three 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 A-6 lists the interval conversion specifications.

Table A-6 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.

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 Smarts reads and writes files

SM_BACKUP_FILE_LIMIT

Smarts 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 Smarts programs. For more information, including an example, see "Managing Log Files" on page 7-1.

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 Smarts 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 A-7 lists the directories whose location is controlled by SM_WRITEABLE:

Table A-7 Directories controlled by SM_WRITEABLE

Subdirectory	Contains
/consoles	Saved consoles
/logs	Log files
/repos	Repository files

The location where log files are saved can be separately controlled with the SM_LOGFILES environment variable.

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, Smarts 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 Smarts programs can invoke ASL scripts.

When set, the directories specified by SM_RULESET_PATH are searched before the system defaults.

Variables that control thread locks

SM_COUNT_REPOS_LOCKS

A Domain Manager crashes whenever the stack used by a thread exceeds the space allocated. Threads acquire locks to access instances in the Domain Manager's repository. If a thread exceeds a certain number of locks, then the lock fails with an error code of ELOOP and a message is written to the log file.

This environment variable is used to control lock counting. SM_COUNT_REPOS_LOCKS must be set before starting the Domain Manager. By default, the lock counting is turned off. To enable lock counting, set SM_COUNT_REPOS_LOCKS to TRUE, yes, or 1.

SM_REPOS_LOCKS_LIMIT

This environment variable determines the number of locks which, if exceeded, causes the ELOOP error. SM_REPOS_LOCKS_LIMIT must be an integer number. The default value is 50.

EMC Corporation recommends a value that is less than 100. Setting low values might cause legitimate requests to fail and affect analysis. Setting high values might allow the thread to run out of stack space before reaching the lock count limit.

B

Wildcards Used By Smarts Software

This appendix describes the wildcards used by Smarts software. These wildcards can be used for pattern matching for the security configuration files and also utilities such as sm_service.

EMC Smarts programs, such as the Global Console, enable users to specify wildcard patterns in text fields. You use a wildcard pattern to match a number of elements instead of specifying each element by name.

Types of wildcard patterns

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 B-1. 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 B-2.

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 "&".

Table B-1 Basic wildcard patterns

Character	Description
?	Matches any single character. For example, server?.example.com matches server3.example.com and serverB.example.com, but not server10.example.com.
*	Matches an arbitrary string of characters. The string can be empty. For example, server*.example.com matches server-ny.example.com and server.example.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. For example, server[789-].example.com matches server7.example.com through server9.example.com, but not server6.example.com. It also matches server-.example.com. For example, server[^12].example.com does not match server1.example.com or server2.example.com, but will match server8.example.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. For 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.
\\	Removes the special status of the characters (,), [,], <, and >. For example, the pattern \\(A\\) matches the string (A).

Special characters for compound wildcard patterns are summarized in Table B-2.

Table B-2 Compound wildcard patterns

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.

Index

Symbols

<host_ID>.smarts.lic 3-11, 3-14, 3-28
~BROKER token 5-7, 5-9

A

Access privileges
- All 5-8
- Monitor 5-8
- None 5-8
- Ping 5-8

Additional safeguards 5-26
- File permissions 5-26
- Limiting access to configuration files 5-26
- Limiting access to servers 5-26

Alternate security configuration files
- Specifying 5-14

ASL script
- SM_RULESET_PATH A-16

Audit file 2-9
Authentication
- record format 5-7, 5-8

Availability Manager
- see IP Availability Manager

B

BASEDIR vii, 2-2
brcontrol utility 6-3
Broker
- And licensing 3-8
- brcontrol 6-3
- Clients locating A-9
- Manager
  - Process ID 6-2
  - State last changed 6-2
  - Status 6-2
- Ping access privilege 5-8
- Registry 6-2
- Secure 5-25, 5-29
- Security 5-4
- TCP port 6-4

BROKER token 5-7, 5-9, 5-11
brokerConnect.conf 5-5
- Description 5-8
- Encrypted passwords 5-19
- Syntax 5-6

C

clientConnect.conf
- ~BROKER token 5-9
- BROKER token 5-9
- DEFAULT token 5-9
- Description 5-8
- Encrypted passwords 5-19
- login user field 5-9
- password field 5-9
- PROMPT token 5-9
- record format 5-8
- Syntax 5-6
- target field 5-9
- username field 5-9
- USERNAME token 5-9

Common subdirectories 2-3
Configuration files
- runcmd_env.sh 2-5

Customer support x

D

Default parameters for services 4-14
DEFAULT token 5-7, 5-9
Diffie Helman-Advanced Encryption Standard 5-2
Disabling security 5-16
Displaying installed services 4-9
dmctl utility 2-9
- roll_log 7-6

E

EDITOR 2-6
ELOOP error A-17
EMC customer service x
EMC product information x
EMC technical support x
Enabling security 5-16
Encrypted connections 5-22
- Configuring 5-24

Encrypted passwords 5-19
Encryption 5-17
Environment variables
Boolean values A-3
Date and time formatting A-11
EDITOR 2-6
Hexadecimal values A-2
How values are interpreted A-2
Integer values A-2
LC_TIME A-12
LM_LICENSE_FILE A-10
Octal values A-2
runcmd_env.sh 2-5, 6-5
Setting A-4
SM_AUTHORITY 5-16, A-10
SM_BACKUP_FILE_LIMIT 7-5, 7-7, A-16
SM_BROKER A-10
SM_BROKER_DEFAULT 6-4, 6-5, A-9
SM_CLIENTCONNECT 5-14, A-11
SM_COUNT_REPOS_LOCKS A-17
SM_DISABLE_KEEPALIVES A-9
SM_DISPLAY A-7
SM_EDITOR 2-6, A-7
SM_ENABLE_SNMP_SET A-8
SM_FORMAT_DATE A-12
SM_FORMAT_DATETIME A-13
SM_FORMAT_HPINTERVAL A-15
SM_FORMAT_HPTIME A-14
SM_FORMAT_INTERVAL A-14
SM_FORMAT_TIME A-13
SM_INCOMING_PROTOCOL 5-23, A-9
SM_JAVA A-7
SM_JAVAHOME A-7
SM_KEYFILE A-11
SM_LICENSE 3-9, 3-11, A-10
SM_LMGRD_LICENSE_FILE A-10
SM_LOGFILES 2-10, 7-3, A-16
SM_MAIN_OPTIONS 7-3, A-7
SM_OKLOGIN_LOGLEVEL A-11
SM_OUTGOING_PROTOCOL 5-23, A-9
SM_REPOS_LOCKS_LIMIT A-17
SM_RULESET_PATH 2-8, A-16
SM_SERVERCONNECT 5-14, A-11
SM_SNMP_BUG_COMPATIBLE A-8
SM_WRITEABLE 2-9, 7-5, A-16
TZ A-11
VISUAL 2-6
Evaluation license file 3-2
Event Logger 7-2

F

Firewall
Access through 3-13, 3-14, 3-24
And licensing 3-12
One-way connection 3-12
flexlm.log 3-9, 3-30
FLEXnet 3-2
Document (PDF) 3-1, 3-22
FLEXnet Manager daemon (lmgrd) 3-10
License server 3-2, 3-8, 3-11, 3-12
LM_LICENSE_FILE A-10
lmutil 3-4
log file 3-9, 3-30
Port for license server 3-14, 3-23
SM_LMGRD_LICENSE_FILE A-10
Tools 3-22

G

Global Console
Client 5-3
Security prompting 5-5

H

Help x
Host ID
And IP address 3-31
Licensing 3-4

I

ic-serviced 4-4
install_license script 3-14, 3-20
Installing programs as services 4-7
Integer values A-2
IP Availability Manager
Discovery and volume licensing 3-26

K

KeepAlive messages A-9
kill command 7-6

L

LC_TIME A-12
Licensing
<host_ID>.smarts.lic 3-11, 3-14, 3-28
Adding features or licenses 3-5, 3-17
Block of licenses 3-25
Components 3-8
Considerations 3-11
Determining number of licenses in use 3-31
Discovery and volume licensing 3-26
Examples of license files 3-28
Firewall 3-12
Firewall access 3-13, 3-14, 3-24
flexlm.log file 3-9, 3-30
FLEXnet license server 3-2, 3-8, 3-11, 3-12
FLEXnet Manager daemon (lmgrd) 3-10
FLEXnet software 3-2
Host ID 3-4
Host name 3-31
How products are licensed 3-10
How to obtain
A license file 3-4
Additional licenses 3-5
Information request form 3-4
install_license script 3-14, 3-20
Installation upgrade 3-8, 3-18
Installing license server 3-14, 3-20
lmutil 3-4

Multiple sites 3-13
New installation 3-8
Number of licenses 3-29
Permanent license file 3-2
Port for FLEXnet license server 3-14, 3-23
Port for vendor daemon 3-13, 3-14, 3-24
Process flow 3-9
Proper location for
Permanent 3-14
Temporary 3-6
Renew a temporary license file 3-4
Requirements 3-2
For permanent license file 3-4
For temporary license file 3-4
Host ID 3-4
Rules for
Permanent license file 3-11, 3-28
Temporary license file 3-6
Running old license files 3-18
Set up
Permanent 3-8, 3-13
Temporary 3-6
SM_LICENSE 3-9, 3-11
smarts.lic 3-6, 3-28
Starting license server 3-15, 3-22
Stopping license server 3-22
System IP address 3-31
Temporary license file 3-2
Tools 3-22
Travelling license file 3-2
Troubleshooting 3-30
Types of daemons 3-8
Types of license files 3-2
Uninstalling license server 3-21
Upgrade 3-17
Utilities for volume licensing 3-25
Vendor daemon (sm_lmgrd92) 3-10, 3-13, 3-14, 3-24
Verifying version of license server 3-22
Versions and compatibility 3-10
Volume licensing 3-25
LM_LICENSE_FILE A-10
lmutil 3-4
Locating the broker A-9
Lock counting A-17
Log files 7-1, 7-2
Adapter 7-5
Audit file 2-9
Controlling the number 7-7
FLEXnet log file 3-9
Manager log file 7-5
Naming convention 7-5
Rolling over 7-6
Starting a new one 7-6
Statistics log file 2-9
Logging mechanism 7-2
Logging messages
Overview 7-2
login user 5-9

M

Manager
Status
Dead 6-2
Running 6-2
Unknown 6-2
Matching
Pattern B-1
Modifying service parameters 4-12

O

Operator
Wildcard B-2

P

Passwords
Encrypting 5-19
Pattern B-1
Pattern matching B-1
Permanent license file 3-2
Permanent license information request form 3-4
Port
Common port value defaults 3-13
process ID 7-6
Product suites and directories 2-2
PROMPT token 5-9

R

Removing services 4-13
roll_log command 7-6
runcmd_env.sh 2-5, 6-5, A-4
And licensing 3-16

S

Script
install_license 3-20
Secret phrase 5-17
Changing 5-17
Locking 5-18
Using for multiple suites and sites 5-18
Security
Access privileges
All 5-8
Monitor 5-8
None 5-8
Ping 5-8
Automatic authentication 5-5
Broker 5-4
brokerConnect.conf
Description 5-8
Syntax 5-6
clientConnect.conf
Description 5-8
Syntax 5-6
Diffie Helman-Advanced Encryption Standard 5-2
Enabling or disabling 5-16
Encryption 5-5, 5-17
Global Console 5-5

Network account authentication 5-10
serverConnect.conf
Description 5-7
Syntax 5-6
Security configuration
Examples 5-27
Security feature 5-2
Authentication 5-2
Encrypted connections 5-2
Encrypted passwords 5-2
Seed file
Encrypted 5-21
serverConnect.conf
~BROKER token 5-7
BROKER token 5-7, 5-11
DEFAULT token 5-7
Description 5-7
Encrypted passwords 5-19
password field 5-7
privilege field 5-7
record format 5-7
Syntax 5-6
SYS token 5-7
target field 5-7
username field 5-7
Service x
Services
Definition 4-2
sm_service utility 4-2
Setting environment variables A-4
sm_adapter 4-15
common options 4-15
SM_AUTHORITY 5-16, A-10
Setting the variable 5-16
sm_authority program 2-10
SM_BACKUP_FILE_LIMIT 7-5, 7-7
sm_beacon 4-15
common options 4-15
SM_BROKER A-10
SM_BROKER_DEFAULT 6-4, 6-5, A-9
SM_CLIENTCONNECT 5-14, A-11
SM_COUNT_REPOS_LOCKS A-17
SM_DISABLE_KEEPALIVES A-9
SM_DISPLAY A-7
sm_edit utility 2-4, 2-5, 2-6, A-7
Encrypting passwords 5-19, 5-20, 5-21
SM_EDITOR 2-6, A-7
SM_ENABLE_SNMP_SET A-8
SM_FILE_BACKUP_LIMIT A-16
SM_FORMAT_DATE A-12
SM_FORMAT_DATETIME A-13
SM_FORMAT_HPINTERVAL A-15
SM_FORMAT_HPTIME A-14
SM_FORMAT_INTERVAL A-14
SM_FORMAT_TIME A-13
SM_INCOMING_PROTOCOL 5-23, A-9
SM_JAVA A-7
SM_JAVAHOME A-7
SM_KEYFILE A-11
SM_LICENSE 3-9, 3-11, A-10
SM_LMGRD_LICENSE_FILE A-10
sm_logerror 2-10
SM_LOGFILES 2-10, 7-3, A-16
SM_MAIN_OPTIONS 7-3, A-7
SM_OKLOGIN_LOGLEVEL A-11
SM_OUTGOING_PROTOCOL 5-23, A-9
SM_REPOS_LOCKS_LIMIT A-17
SM_RULESET_PATH 2-8, A-16
sm_server
common options 4-14, 4-16
SM_SERVERCONNECT 5-14, A-11
sm_service 4-3
actions 4-5
install action 4-7
install options 4-7
remove action 4-13
show action 4-9
standard options 4-5
standard options for logging 7-2
start action 4-10
start options 4-10
stop action 4-10
stop options 4-10, 4-11
sm_serviced 4-4
SM_SNMP_BUG_COMPATIBLE A-8
sm_tpmgr command 3-26
SM_WRITEABLE 2-9, 7-5, A-16
smarts.lic 3-2, 3-6, 3-28
smgetinfo utility 2-11
SNMP A-8
Starting and stopping services 4-10
Starting license server 3-15, 3-22
statistics log file 2-9
Stopping license server 3-22
Subdirectories
common 2-3
SYS token 5-7
syslog mechanism 7-2

T

TCP
Broker port 6-4
Technical support x
Time zone A-11
Trial license file 3-2
TZ A-11

U

umask
UNIX file mode creation mask 5-3
UNIX syslog mechanism 7-2
USERNAME token 5-9
Utilities
brcontrol 6-3
dmctl
roll_log 7-6
sm_edit 2-4, 2-5, A-7
smgetinfo 2-11

V

VISUAL 2-6
Volume licensing 3-25

W

Wildcard B-1
    Chart of operators B-2
Windows Domain Manager 5-10
Windows Event Logger 7-2