EMC® Smarts®

Foundation 9.4

System Administration Guide

P/N 302-002-291

REV 01

EMC²

Copyright © 2007 - 2015 EMC Corporation. All rights reserved. Published in the USA.

Published October, 2015

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.

EMC², EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (https://support.emc.com).

EMC Corporation
Hopkinton, Massachusetts 01748-9103
1-508-435-1000 In North America 1-866-464-7381
www.EMC.com

CONTENTS

Chapter 1 Introduction

Overview 10
Setting the locale 10
Locating and modifying files 10
Licensing the software 11
Controlling the startup of EMC Smarts programs 11
Securing access to software 11
Operation of the Broker 11
Managing log files 12

Chapter 2 Locale

What is a locale 14
  Client and Domain Manager use different locales 14
Locale types 15
  Process locale 15
  Client locale 15
  Domain Manager process locale 16
  Support locale 16
Standards 16
  Unicode 16
  ISO 639 language codes 16
  ISO 3166 country codes 16

Chapter 3 Locating and Modifying Files

Directories 18
User-modifiable files 19
  Modifying EMC Smarts files 20
  Associating files with sm_edit on Windows 22
Locating user-modifiable files 23
  Controlling the location for rule set files 23
  Controlling location of the user modified files 23
Writing output files 25
  Controlling where the software writes output files 25
  Controlling where the software writes log files 26
  sm_logerror 26
Using the smgetinfo utility to create a support package 27
  sm_getinfo files 27
  sm_getinfo command-line syntax 28
  sm_getinfo invocation examples 30
  sm_getinfo data collection 30

Chapter 4 License Administration

Overview of the steps to set up license management 34
License components and process flow 35
Examples of license files 37
  Temporary license file example 37
  Permanent license file example 37

Obtain and set up a temporary license file 39
How to renew a temporary license file 40
Obtain and set up a permanent license file 40
Rehost license 42
Regenerate license 42
Install and configure the FlexNet Publisher license server 43
Use an existing FlexNet Publisher license server 44
Add another permanent license to a running license server 45
Change the license server port in a running license server 46
Enable communication through a firewall in a running license server 47
Use of multiple FlexNet Publisher license servers 47
Upgrade licensing during EMC Smarts software upgrades 48
Run older versions of license files in parallel 48
Overview of the install_license script 50
Uninstall the FlexNet Publisher license server 51
License server administration tools 52
Verify the FlexNet Publisher license server version 52
Starting and stopping the license server 52
License checking by an EMC Smarts application 53
Volume licensing for EMC Smarts IP Availability Manager 54
Determining the total number of licenses 54
Determining the number of checked-out licenses 54
Discovery and licenses 56
Troubleshooting 57
Evaluation software no longer runs 57
Messages in the flexlm.log file 57
Need to determine the FlexNet Publisher license server and all licenses served 58
Cannot find license error 58
Changing an IP address and its effect on licensing 58
Error occurs when starting additional Domain Managers 59
License file corrupted when attempting to edit license file 59
Cannot restart FlexNet Publisher license server after applying a Service Pack 60
Introduction to Service Assurance Suite (SA Suite) and its licensing 61
Switch/router classification and conversion examples 62
SA Suite license count report 63

Chapter 5 Controlling Startup

Root access requirement on UNIX and Linux 66
Overview of services 66
About the sm_service utility 67
sm_serviced and ic-serviced 67
The sm_service command line 69
Standard options 69
sm_service actions 70
Installing programs as services with sm_service 71
Install options 71
Examples of the sm_service install action 72
Displaying installed services and their status 73
Examples of the sm_service show action 73
Starting and stopping services with sm_service 74
Start options 74
Stop options 74

Examples of the sm_service start and stop actions 74
Checking whether services are stopped 75
Examples of the sm_service isstopped action 75
Starting and stopping services from the Windows desktop 75
Modifying service parameters with sm_service 76
Removing services with sm_service 76
Example of the sm_service remove action 76
Default parameters for services 77
Common sm_server options 77
Common sm_adapter options 79
Common sm_sdi options 80
Running as non root on UNIX and Linux systems 81

Chapter 6 Securing Access to the Software

Security features 84
How security works 85
Server and client programs for the software 85
Server authentication 87
Client authentication 87
Encryption 88
Configuring authentication 88
Syntax of the security files 88
Server authentication records 89
Client authentication records 91
Special authentication values 93
Default authentication records 95
Specifying alternate security configuration files 98
Controlling authentication 100
Setting the SM_AUTHORITY variable 100
Changing Cryptographic Settings 101
Changing Pseudo-random Number Generators (PRNG) algorithm 101
Changing the use of the default PRNG algorithm: 101
Encryption 101
Changing the secret 102
Using the secret for multiple products and sites 103
Locking the secret 103
Encrypted passwords 104
Encrypting passwords 104
Encrypted seed files 106
Encrypted connections 106
Levels of encryption 106
Encrypting connections in non-FIPS 140 mode 108
Encrypting connections in FIPS 140 mode 109
Suggested encrypted connections in non-FIPS 140 mode 109
Replacing or importing TLS certificates 110
Configuring a secure Broker 110
Additional safeguards 112
Using file permissions 112
Limiting access to the configuration files 112
Limiting access to servers 112
Example security configurations 113
Default security 113
Customized security 114

Chapter 7 Operation of the Broker

Overview of the Broker 118
Viewing the registry of the Broker 119
How the clients find the Broker 119
How to change the Broker environment variable 120
Correcting the Broker settings 120
Securing the Broker 121

Chapter 8 Continuous Audit of Client Connections

Overview of continuous auditing of client connections 124
System Element objects 125
SM_SystemElement 125
SM_ElementManager 126
Client data persists after connection terminates 128
File descriptor usage warning event 128
Acquiring client information 129
Viewing SM_ElementManager attribute values 130
Changing SM_ElementManager attribute values 131

Chapter 9 Managing Log Files

Overview of message logging 134
Command-line options that control messages 137
Retrieving and setting log, error and trace levels at runtime 138
Log files 139
Diagnostic header 140
Date and timestamp, and thread id in each log message 142
Message at the end of a rolled log 142
Starting a new log file 142
Controlling the number of saved log files 144

Appendix A Environment Variables

How variable values are interpreted 146
How integer variables are interpreted 146
How Boolean variables are interpreted 147
Methods for setting environment variables 147
Setting an environment variable product-wide 147
Setting an environment variable for a program 148
Environment variables for users 149
Variables that control the locale and encoding 161
Variables that control Java settings 163
Variables that control the Broker 164
Variables that control FlexNet Publisher licensing 165
Variables that control security 165
Variables that control date and time formatting 167
Variables that control reading and writing files 169
Variables that control thread locks 171
Variables that control log subscription changes and Dynamic Modeling errors 171

Appendix B Wildcards

Types of wildcard patterns 174

FIGURES

	Title	Page
1	sm_getinfo file tree	27
2	Process flow	36
3	SM_SystemElement modeling	125
4	Message logging	134

TABLES

	Title	Page
1	Directories	18
2	Subdirectories for user-modifiable files	19
3	Default subdirectories for writable files	25
4	sm_getinfo-created files	27
5	Command-line options for the sm_getinfo utility	29
6	Products - Standard / Advanced	61
7	Pricing model based on Managed Devices and Managed Objects	62
8	Standard options for sm_service	69
9	sm_service actions	70
10	Install options for sm_service	71
11	Start options for sm_service	74
12	Stop options for sm_service	74
13	isstopped options for sm_service	75
14	Common sm_server options	77
15	Common sm_adapter options	79
16	Common sm_sdi options	80
17	Additional options for running as non root on UNIX and Linux systems	81
18	EMC Smarts server and client programs	86
19	Field descriptions for serverConnect.conf	90
20	Fields for clientConnect.conf and brokerConnect.conf	92
21	Encryption levels for connections for non-FIPS 140 mode	107
22	Encryption for connections for FIPS 140 mode	107
23	SM_SystemElement subclasses	125
24	Key attributes for SM_ElementManager	126
25	Log filenames, locales and encodings	136
26	SM_LOCALE and SM_OUTPUT_ENCODING examples	136
27	Computed attributes to retrieve and set log, error, and trace levels at runtime	138
28	Examples of numeric values	146
29	Standard options supported by SM_MAIN_OPTIONS	153
30	Acceptable values for the SM_IP_VERSIONS environment variable	156
31	Recommended values for SM_MAX_MONITORING_THREADS	157
32	Valid levels for SM_MAX_SESSION_VERSION	158
33	Directories controlled by SM_WRITEABLE	170
34	Basic wildcard patterns	174
35	Compound wildcard patterns	175

CHAPTER 1
Introduction

This chapter consists of the following sections:

- Overview 10
- Setting the locale 10
- Locating and modifying files 10
- Licensing the software 11
- Controlling the startup of EMC Smarts programs 11
- Securing access to software 11
- Operation of the Broker 11
- Managing log files 12

Overview

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

EMC Smarts installation directory

In this document, the term BASEDIR represents the location where EMC Smarts software is installed.

- For Windows, this location is C:\InCharge\‹*product*›.
- For UNIX, this location is /opt/InCharge/‹*product*›.

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

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

However, you cannot change the ‹*product*› location under the root directory.

This guide provides more information about the directory structure.

Where to get help

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

Product information — For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

https://support.emc.com

Technical support — Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

Setting the locale

A locale is a combination of language and country identifiers, and the specification of the locale is important for the purpose of externalized textual data. Clients and Domain Managers can be running in different locales but it is important that textual data transferred to a client as part of error handling by the Domain Manager appear in the language and country of the client. Chapter 2, "Locale," describes how to specify the locales.

Locating and modifying files

Some files can be modified to meet the needs of your network environment. Chapter 3, "Locating and Modifying Files," describes the directory structure of the EMC Smarts software. The chapter also discusses the files that can be modified and how to locate the files and provides procedures for modifying those files. Additionally, the chapter describes the files created by EMC Smarts Foundation software (for example, logs, repositories, and saved consoles), and details how to control their location.

Licensing the software

EMC 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. Chapter 4, "License Administration," provides procedures for obtaining both evaluation and permanent license files for your software. Topics include:

- Setting up the license file
- Starting the FlexNet Publisher license server
- Pointing the software to the license server
- Licensing additional programs and adding licenses

Controlling the startup of EMC Smarts programs

EMC Smarts programs can either be installed as services or as manual processes. Chapter 5, "Controlling Startup," provides procedures that use the sm_service utility to:

- Install programs as services
- Start and stop services
- Start and stop processes
- Modify the runtime parameters of services

Securing access to software

Security is a critical concern in the world of large-scale distributed networks. Therefore, EMC Smarts security provides a means by which network administrators can set up security to control access to the software. Chapter 6, "Securing Access to the Software," describes how security works, as well as the authentication processes used by the software. It then provides procedures for setting up security and controlling access to EMC Smarts programs. The chapter also discusses how to set up encrypted connections between EMC Smarts clients and servers.

Operation of the Broker

Client programs, such as a console or an adapter, use the Broker to determine where Managers are running. When a Manager starts, it registers the hostname of the machine it is running on, as well as the TCP port it is listening to, with the Broker. Thereafter, clients retrieve this information from the Broker so that they can communicate with the Manager. Chapter 7, "Operation of the Broker," describes the functions of the 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, EMC Smarts software can maintain up to *999* backup log files of each log. For example, if you are generating two log files, MYDM_en_US_UTF-8.log and MYDM_fr_UTF-8.log file, you can maintain up to *999* backup log files for the MYDM_en_US_UTF-8.log and *999* backup log files for the MYDM_fr_UTF-8.log.

You can configure the number of log files retained by the software. Chapter 9, “Managing Log Files,” discusses the management of log files, and describes how to “roll over” the files.

CHAPTER 2
Locale

This chapter consists of the following sections:

- What is a locale 14
- Locale types 15
- Standards 16

What is a locale

A *locale* is a set of parameters that defines:

1. Language
2. Country (or region)
3. Other specific cultural elements that vary

For the purpose of externalized textual data, the language and country data of the locale specification are important.

Languages are specified in two lowercase ASCII letters. Some examples are en for English and fr for French.

Country identifiers are specified as two or three uppercase ASCII letters. Some examples are: US, CA, FR, and JP, for United States, Canada, France, and Japan, respectively.

"Standards" on page 16 provides more information on language codes and country identifiers.

A locale is a combination of the language and country identifiers into one string, connected by the underscore (_) character. For example, the locale for English (en) as spoken in the United States (US) is defined as `en_US`.

Client and Domain Manager use different locales

Client programs may be running in a locale that is not the same as the locale of the Domain Manager. For example, a Domain Manager may be running in a process where the locale is specified as "French as used in France." Different end-users of an application may be concurrently connected, and use different locales, for example "German as spoken in Germany," and "Spanish as spoken in Spain," or "German as spoken in Austria."

It is important that textual data transferred to the client as part of error handling by the Domain Manager appear in the language and country of the client. A German client will see German exception messages, even if the client application is connected to a Domain Manager whose preferred language (for logging) is French.

The server will respect the locale of the client when rendering several types of textual output. Applications may also use classes and methods in the clsapi subsystem to enable locale-aware operations, where appropriate.

Locale types

Each process' locale is determined by the SM_LOCALE environment variable, as seen by that process. When a client process attaches to a server process, they negotiate on the locale for that session. The SM_LOCALE environment variable is discussed in "SM_LOCALE" on page 161.

Process locale

You can specify the locale of any program, such as a client, Broker, server, utility by using the SM_LOCALE environment variable. For all such programs, the initialization logic validates the specified value, and ensures that all internationalization features can use it predictably.

The locale can also be specified for Java applications. Use the Java property `com.emc.cmp.dmt.locale` on the startup of the Java Virtual Machine or on the command line of any program that supports the `-java` command operand.

Client locale

The locale of a client program may be different from the locale of the Domain Manager. There are two different types of client locales:

- Client process locale
- Client session locale

The client process locale and the client session locale do not need to be the same.

Client process locale

The client process locale is the locale that is set by the SM_LOCALE variable in the environment of the client program, such as **dmctl**, **brcontrol**, and others. The client process locale is used by default when rendering messages generated in the client process.

Client session locale

The client session locale is the locale that is sent to a Broker or Domain Manager in two possibly ways:

- The client process locale is by default sent to the Domain Manager, and identifies the initial session locale.
- The client session locale can be changed by calling the `setLocale()` or `setRemoteLocale()` method. Calling one of these methods causes a protocol verb to be executed at the server, which causes the server to remember the client session locale.

The client program may use the `setLocale()` client API calls to indicate to the Domain Manager a select locale. Setting the locale that the Domain Manager remembers for the client does not change the configured setting of the client process locale.

Domain Manager process locale

The Domain Manager process locale is the one set by the SM_LOCALE variable in the environment of the Domain Manager. The Domain Manager locale primarily influences logging output, and does not impact the processing of locale for client processes or the client sessions. Conversely, the client locale, configured or set by API call, does not influence the locale handling for the Domain Manager output.

Support locale

The process locale specified by SM_LOCALE and the Java property specifies the language for output that the user sees. However, EMC engineering and support teams must be able to interpret log data and other output in English. Programs, primarily server processes, will always generate diagnostic information in the support locale, which is `en_US`. The main impact of the difference between the process locale and the support locale applies to the software logging system. See "Overview of message logging" on page 134 for more details.

Standards

Several internationalization standards apply directly to the software.

Unicode

Unicode is the universal standard of encoding for written characters and text, covering all known world languages. UTF-8 (8-bit form) is used by the software applications.

ISO 639 language codes

Languages are specified in two lowercase ASCII letters. Some examples of standard codes for language are: en, fr, and ja for English, French, and Japanese respectively.

ISO 3166 country codes

This standard defines the universally recognized codes for county (or region). There are two alphabetic standards for country codes:

- ISO-3166-1 Alpha-2 describes the two-letter codes.
- ISO-3166-1 Alpha-3 describes the three-letter codes.

Country identifiers are specified as two or three uppercase ASCII letters. For example, US for United States and JP for Japan.

Language and country identifiers are combined into one string, connected by the underscore character. For example, en_US for English as used in the US, and ja_JP for Japanese as used in Japan.

CHAPTER 3
Locating and Modifying Files

This chapter consists of the following sections:

- Directories 18
- User-modifiable files 19
- Locating user-modifiable files 23
- Writing output files 25
- Using the smgetinfo utility to create a support package 27

Directories

Table 1 on page 18 lists software directories and a brief description of their contents.

Table 1 Directories

Directory	Contents
/bin	Executables for starting and interacting with Domain Managers as well as utilities.
/classes	Software jar files and third-party jar files upon which EMC Smarts programs depend.
/conf	Domain Manager configuration files.
/doc	User and reference guides.
/examples	Sample files for C, C++, and Java APIs as well as examples of MODEL applications and JIIM.
/include	The .h and .mdl files for components of the software programs.
/integration	Sample probe for the integration accessor.
/l10n	The l10n directory name is a lowercase L, followed by the number 10 and then a lowercase n. Contains Resource Packages (which hold externalized and localized data, such as error strings).
/lib	Shared libraries.
/local	Top-level directory where local copies of configuration, rules, and script files are created and stored. The default location where files such as repository and log files are written. The /local/l10n directory contains new Resource Packages introduced by update kits, patches, language packs, and so on.
/perl	Files used for the Perl API.
/rules	ASL rule sets.
/script	Utility scripts such as the license server installation scripts.
/symbols	Symbols for debugging and gathering extended information about stack traces (Windows only).

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

User-modifiable files

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

Never alter system files, such as executables, shared libraries, MODEL files, and setup files. Do not 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 discovery.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. The 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, the software then searches appropriate non-local directories.

To edit a file, invoke the sm_edit utility from the 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. "Modifying EMC Smarts files" on page 20 provides further information about how to use the sm_edit utility.

You should:

- *Only* modify local copies of user-modifiable files.
- *Always* retain the original, unedited, version of these files.

Table 2 on page 19 identifies the default subdirectories that include user-modifiable files found under BASEDIR/smarts/local. The subdirectories vary by product.

Table 2 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

NOTICE

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 on page 19. During installation, the software may create empty subdirectories under the BASEDIR/smarts/local directory. These directories are reserved for use for future service packs and Rolling Patches. The exception is the BASEDIR/smarts/local/bin directory that contains files necessary for the successful installation of the software on Windows 2003 (Enterprise Edition) operating systems.

In addition, the BASEDIR/smarts/local/conf directory contains the runcmd_env.sh file, which is created during installation. The runcmd_env.sh is a file for setting environment variables, including the default location for the Broker. You can edit this file to change the default location of the Broker or to define any EMC Smarts Foundation-wide environment variable. "Methods for setting environment variables" on page 147 provides information about setting environment variables.

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 product or module within a product that uses the files.

You can alter the location for rules files, as described in "Controlling the location for rule set files" on page 23.

NOTICE

If EMC installs a site-specific patch at your location, then additional directories may be included under BASEDIR/smarts/local. Never alter files in any of these additional subdirectories.

Modifying EMC Smarts files

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

To modify files, use sm_edit utility that is installed with the software.

NOTICE

You must use the sm_edit utility that is included with the installation of a product to modify files applicable to that product. Do not, for example, edit IP Manager files with the sm_edit that is installed with the Service Assurance Manager.

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.

NOTICE

You can configure Windows to automatically invoke the sm_edit utility when you open a file through Windows Explorer, as described in "Associating files with sm_edit on Windows" on page 22.

To use sm_edit from the command line, specify the filename and include the subdirectory under /local where the file resides. For example, to edit the discovery.conf file, enter the following command by using administrative privileges:

```
BASEDIR/smarts/bin> sm_edit conf/discovery/discovery.conf
```

In this example, sm_edit searches in the BASEDIR/smarts/local/conf/discovery directory for the discovery.conf file. If it finds discovery.conf, it opens the file in a text editor. If sm_edit does not find discovery.conf in the BASEDIR/smarts/local/conf/discovery directory, it creates a local copy of discovery.conf and writes it to the BASEDIR/smarts/local/conf/discovery 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 that is specified by one of the following means:

- SM_EDITOR environment variable, as described in "Environment variables for users" on page 149.
- VISUAL environment variable
- EDITOR environment variable
- If the SM_EDITOR, VISUAL or EDITOR environment variables are not defined, sm_edit uses the following:
  - Edit program on UNIX
  - WordPad program on Windows

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

Associating files with sm_edit on Windows

On Windows systems, you can configure the system so that sm_edit is automatically invoked when an EMC 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 installed on the same host, this association works correctly only with one product (the product from which the associated sm_edit is run). For the other products, you must run the sm_edit by using a command prompt in the product that contains the file to modify.You can associate the following file types with sm_edit:

- For configuration files, use .conf.
- For ASL rulesets, use .asl.
- For data import files, use .template.

NOTICE

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 product. For example, the discovery.conf file located in the BASEDIR/smarts/conf/discovery directory of a given product.
2. Right-click the file and select **Open With** › **Choose Program**.
3. From the **Open With** dialog box, select **Other**.
4. Select the **sm_edit** utility in the BASEDIR/smarts/bin directory to add **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**.
8. Repeat this procedure, except for steps 3 and 4, for each file type that 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."**

Locating user-modifiable files

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

NOTICE

BASEDIR/smarts can refer to different product directories. Therefore, the configuration file in this example could be a configuration file for any product such as the IP Manager or Service Assurance Manager.

Controlling the location for rule set files

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

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

NOTICE

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

Appendix A, “Environment Variables,” provides further information about SM_RULESET_PATH.

Controlling location of the user modified files

In case you would like to run more than one domain manager out of a single installation direcotry you can use the SM_SITEMOD variable to specify a separate local directory structure for each domain.

This allows you to specify configuration files unique to each domain, yet still use the same core files that are not different between sites.

The SM_SITEMOD variable provides a search list that InCharge uses to locate files. This list is used to find files users can, and in some cases must, modify to use InCharge at their site. Such files include configuration files, ASL rulesets, and scripts. The components of the list are separated by colons (:) on UNIX and semicolons (;) on Microsoft Windows. The default value of SM_SITEMOD is: BASEDIR/smarts/local:BASEDIR/smarts. You will need to change this value for each domain to first search BASEDIR/smarts/‹domain_local ›then BASEDIR/smarts/local and the BASEDIR/smarts. Follow these steps:

1. Create a diectory for each domain.

   ```
   $mkdir /opt/InCharge/smarts/local_d1.
   ```

2. Place a file that is unique for this domain under /opt/InCharge/smarts/local_d1.

```
$mkdir /opt/InCharge/smarts/local_d1/conf/trapd
$cp   /opt/InCharge/smarts/conf/trapd
/opt/InCharge/smarts/local_d1/conf/trapd
```

3. Define SM_SITEMOD for the current shell or Unix of the Command prompt on Windows.

   For Unix,

```
export
SM_SITEMOD=/opt/InCharge/IP/smarts/local_d1:/opt/InCharge/IP/smarts
/local:/opt/Incharge/IP/smarts
```

   For Windows,

```
C:\set
SMS_ITEMOD=C:\InCharge\IP\smarts\local_d1;C:\InCharge\IP\smarts\loc
al;C:\InCharge6\IP\smarts
```

4. Use sm_edit to modify the rest of the files that are specific to this domain.

5. Configure sm_service command such that every time the domain starts up, it loads files from the proper directory.

   For Unix,

```
sm_service install --force --name=ic-am-pm-server-2
--description="INCHARGE-AM-PM for Site 2" --startmode=runonce
--env=SM_SITEMOD=/opt/InCharge/IP/smarts/local_d1:/opt/InCharge/IP/
smarts/local:/opt/Incharge/IP/smarts
/opt/InCharge/IP/smarts/bin/sm_server --name=INCHARGE-AM-PM-2
--config=icf --bootstrap=bootstrap-am-pm.conf --port=0
--subscribe=default --ignore-restore-errors --output
```

Writing output files

The BASEDIR/smarts/local directory is also the default location for files written by the EMC Smarts software. Writable files include logs, saved consoles, and repository files. If desired, you can alter the location for log files, as described in "Controlling where the software writes log files" on page 26. Table 3 on page 25 defines the default subdirectories found under BASEDIR/smarts/local that contain writable files.

Table 3 Default subdirectories for writable files

Installation path	Directory	Description
BASEDIR/smarts/local	/consoles	EMC Smarts consoles that have been saved.
	/logs	Manager and adapter log files that the 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 the software has written.

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

```
dmctl -s <manager_name> invoke ICS_ActionManager::ICS-ActionManager
   dumpStatistics <filename>

dmctl -s <manager_name> invoke
   ICS_AutoActionManager::ICS-AutoActionManager dumpStatistics
   <filename>
```

The *‹filename›* is optional; otherwise, it is *‹manager_name›*-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 the software writes output files

The software defines the location of writable files by using the SM_WRITEABLE environment variable. By default, SM_WRITEABLE is defined as BASEDIR/smarts/local.

This means the following:

- Saved consoles are written to BASEDIR/smarts/local/consoles.
- Log files are written to BASEDIR/smarts/local/logs.
- Repository files are written to BASEDIR/smarts/local/repos.

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

To set SM_WRITEABLE, add the variable to the runcmd_env.sh file.

Controlling where the software writes log files

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

To set SM_LOGFILES, add the variable to the runcmd_env.sh.

SM_LOGFILES takes precedence over SM_WRITEABLE, as described in "SM_LOGFILES" on page 170.

sm_logerror

On Solaris and Linux systems, every ITOps 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 ITOps 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 Manager, or checks credentials received by a Manager. Since the authenticator itself is a an ITOps program, it starts its own sm_logerror. Thus, if you follow the tree of processes under an sm_server, you might find:

```
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 Manager starts as a service:

- The Domain Manager 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 create a support package

The sm_getinfo utility is used by customers to collect data for troubleshooting EMC Smarts Manager (server) problems. The utility backs up the current configuration for a server by creating a tar archive of all files and user customizations that are essential to troubleshooting the server. Customers then email the tar archive to EMC Customer Support for problem resolution.

sm_getinfo files

The sm_getinfo utility, which is supported on Solaris, Linux, and Windows, creates three types of files in the installation directory area from which it is invoked. The files are shown in Figure 1 on page 27 and described in Table 4 on page 27.

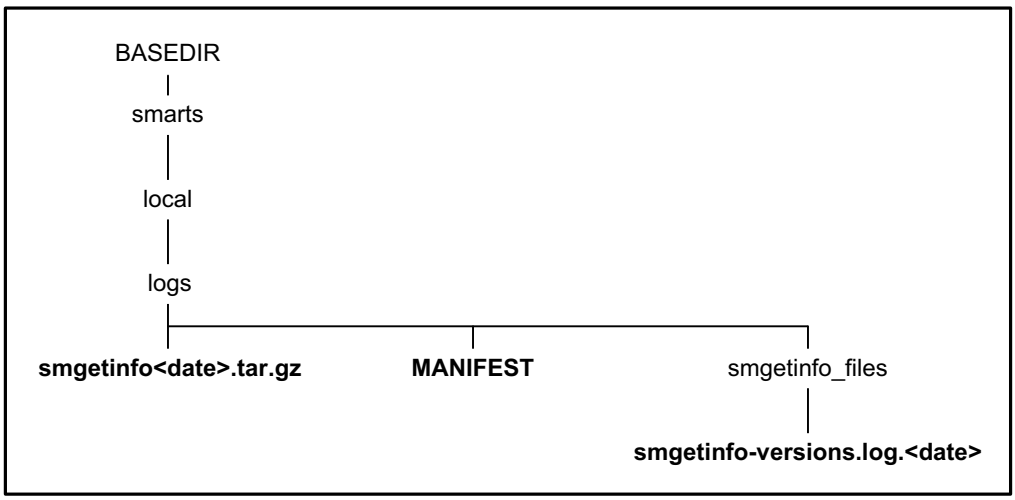

Figure 1 sm_getinfo file tree

Table 4 sm_getinfo-created files (page 1 of 2)

Filename	Description
Files in logs directory	
sm_getinfo‹date›.tar.gz Example: sm_getinfo26Mar2012-015952.tar.gz	A compressed tar archive in which the sm_getinfo utility stores a server 's log files, repository files, core files (Solaris, Linux) or dump files (Windows), user-modified files (using sm_edit), user-introduced files, and system environment information. The actual content of the tar archive depends on the user-specified options on the sm_getinfo invocation command line. The name of the tar archive includes the date when the tar archive was created.
MANIFEST	A text file that lists all of the files that the sm_getinfo utility includes in the tar archive.

Table 4 sm_getinfo-created files (page 2 of 2)

Filename	Description
Files in logs/smgetinfo_files directory	
smgetinfo-versions.log.‹date› Example: smgetinfo-versions.log.26Mar2012-020004 Example of other files in smgetinfo_files: TTP-Installed-versions.log.26Mar2012-020005	A log file in which the sm_getinfo utility writes information about a server's log file or repository file that is larger than 700 megabytes (MB). The sm_getinfo utility does not include any log or repository file in a server's tar archive that exceeds 700 MB. In addition, sm_getinfo writes system information to the log file. The name of the log file includes the date when the log file was created.

sm_getinfo command-line syntax

You run the sm_getinfo utility from the BASEDIR/smarts/bin directory. The options that you specify on the invocation command line determine which files are included in the sm_getinfo-created tar archive.

Here is the command line syntax for sm_getinfo:

```
sm_perl sm_getinfo.pl
| --server <server name> --pid <server process ID>
| --broker <location>
| --log [<number of latest logs>]
| --nolog
| --repos [--latest]
| --norps
| --core
| --all
| --version
| --help
| --smmonitor "<sm_monitor options>"
| --flush
```

where:

- ‹› Angle brackets are user-supplied parameter values (variables).
- [ ] Square brackets are optional entries.
- | Vertical bar symbols are alternate selections.

The command-line options are described in Table 5 on page 29.

Table 5 Command-line options for the sm_getinfo utility

Option	Description
--server |‹server name› --pid ‹server process ID› | -s ‹server name› -p ‹server process ID›	Specifies the EMC Smarts server name and PID against which the sm_getinfo utility will collect information. To dump a core file for a running server on Solaris or Linux, the pid option must be provided. Also, the server name is used to invoke the sm_monitor tool.
--broker ‹location› | -b ‹location›	Specifies an alternate Broker location as host:port.
--log [‹number of latest logs›] | -l [‹number of latest logs›]	For each EMC Smarts server, collects and stores a user-specified number of latest logs or all generated log files that are in the BASEDIR/smarts/local/logs directory. Note that whenever collecting a log, the related .audit and .archive files should be collected as well. If a server name is specified, only the files that correspond to the server will be collected. To avoid generating a too-large tar archive (too large to email), any log file that is larger than 700 MB will be excluded and its information will be logged in smgetinfo-versions.log‹date›.
--nolog | -g	Excludes log files when collecting local files. This option and --log are mutually exclusive.
--repos [--latest] | -r [-t]	For each EMC Smarts server, collects and stores the latest repository file or all repository files. If a server name is specified, only the file corresponding to the server will be collected. Any repository file that is larger than 700 MB will be excluded and its information will be logged in smgetinfo-versions.log‹date›.
--norps | -n	Excludes repository files when collecting local files. This option and --repos are mutually exclusive.
--core | -c	Collects and stores the core files (Solaris, Linux) or dump files (Windows) that are generated by the EMC Smarts software, and the corresponding logs.
--all | -a	All data and files in BASEDIR/smarts/local and BASEDIR/smarts/setup directories will be collected and stored. If --all option is specified, the other options will be ignored except --server and --pid.
--version | -v	Print version information and exit.
--help | -h	Print usage information and exit.
--smmonitor “‹sm_monitor options›” | -m “‹sm_monitor options›”	Specifies the options for running sm_monitor, which will override the default options “-m run-all -z.” Approximately two cycles are run to collect the required information. The collected information is output to the BASEDIR/smarts/local/logs/SM-Monitor-‹server name› directory.
--flush | -f	Force a flush. Needed when the sm_getinfo utility is invoked from a remote host.

sm_getinfo invocation examples

To gather the five latest logs, enter:

```
sm_perl sm_getinfo.pl  --logs 5
```

To gather the latest repository data and core files, enter:

```
sm_perl sm_getinfo.pl  --repos --latest --core
```

To gather the entire BASEDIR/smarts/local directory, enter:

```
sm_perl sm_getinfo.pl  --all
```

To invoke sm_monitor, enter:

On UNIX/Linux:

```
sm_perl sm_getinfo.pl -s <server name> -m "-m correlation -z"
```

On Windows:

```
sm_perl sm_getinfo.pl -s <server name>  -m "-m mem"
```

sm_getinfo data collection

If no command-line option is specified, the sm_getinfo utility will store the following information in the tar archive:

- For each server, the latest server log file in BASEDIR/smarts/local/logs and the related .audit and .archive files in BASEDIR/smarts/local/logs. If a server name is specified, only the files that correspond to the server will be collected. To avoid generating a too-large tar archive (too large to email), any log file that is larger than 700 MB will be excluded and its information will be logged in smgetinfo-versions.log‹date›.
- For each server, the latest repository file in BASEDIR/smarts/local/repos. If a server name is specified, only the repository file for the server will be archived. Any repository file that is larger than 700 MB will be excluded and its information will be logged in smgetinfo-versions.log‹date›.
- The local files that are not in the BASEDIR/smarts/local/logs and repos directories and changed since last temporary test patch (TTP) and patch.
- The new local files that are not in the BASEDIR/smarts/local/logs and repos directories and were added since the last TTP and patch.
- All local files except the files in logs and repos directories if no TTP or patch is installed.
- All files in the BASEDIR/smarts/setup/info and BASEDIR/smarts/setup/logs directories.
- Core files (Solaris, Linux) or dump files (Windows) that are generated by the EMC Smarts software, and the corresponding server log files. On UNIX or Linux, some library (lib) files that are related to the cores are also collected.

- EMC Smarts TTP or patch version information. Additionally, it verifies MD5 checksum for the installed TTP files and the files that are listed in manifest.md5 in the BASEDIR/smarts/setup/info directory.
- The data collected by sm_monitor. If a server name is specified, only the files that correspond to the server will be collected.
- The data generated by the EMC Ionix ITOps Health Monitor.
- System environment information.

CHAPTER 4
License Administration

This chapter consists of the following sections:

- Overview of the steps to set up license management 34
- License components and process flow 35
- Examples of license files 37
- Obtain and set up a temporary license file 39
- Obtain and set up a permanent license file 40
- Install and configure the FlexNet Publisher license server 43
- Use an existing FlexNet Publisher license server 44
- Add another permanent license to a running license server 45
- Change the license server port in a running license server 46
- Enable communication through a firewall in a running license server 47
- Use of multiple FlexNet Publisher license servers 47
- Upgrade licensing during EMC Smarts software upgrades 48
- Run older versions of license files in parallel 48
- Overview of the install_license script 50
- Uninstall the FlexNet Publisher license server 51
- License server administration tools 52
- License checking by an EMC Smarts application 53
- Volume licensing for EMC Smarts IP Availability Manager 54
- Troubleshooting 57
- Introduction to Service Assurance Suite (SA Suite) and its licensing 61

Overview of the steps to set up license management

You should use the license administration procedures in this chapter in conjunction with the procedures in each EMC Smarts product's installation guide and in the *FlexNet Publisher License Administration Guide*. To complete the procedures, you will need administrative privileges for both the EMC Smarts software and the host systems on which the software is running.

For licensing and license management, EMC Smarts Foundation products rely on a third-party licensing software product, FlexNet Publisher®, from Flexera Software™ Corporation. The FlexNet Publisher licensing software is installed automatically during the software installation process.

You will need to perform the following to complete the licensing process for your EMC Smarts products:

1. Obtain and setup one of the following license files from EMC:
   - Temporary — time-limited.

     "Obtain and set up a temporary license file" on page 39 provides further information.

     Setting up a temporary license file is a postinstallation task.
   - Permanent — not time-limited.

     "Obtain and set up a permanent license file" on page 40 provides further information.

     For installation upgrades (such as upgrading entire products or adding products to an existing directory), obtaining and setting up a permanent license file 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 postinstallation task.

     The installation guide that accompanied your product describes the order of installation tasks. The migration guide that accompanied your product describes the license migration procedure.
2. For permanent license files only, do one of the following:
   - Use an existing FlexNet Publisher license server that you already installed for other software applications. "Use an existing FlexNet Publisher license server" on page 44 provides additional information.
   - Install and configure the FlexNet Publisher license server provided with the EMC Smarts software. "Install and configure the FlexNet Publisher license server" on page 43 provides additional details.

License components and process flow

Licensing involves the following components:

- FlexNet Publisher license server—The FlexNet Publisher license server authenticates EMC Smarts products. The FlexNet Publisher 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 Publisher-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 EMC Smarts software.
- Permanent license file—The permanent license file is generated by EMC. It controls access to applications and features as well as limits the number of systems discovered by EMC Smarts products.
- EMC Smarts products—These products communicate with the FlexNet Publisher license server to request a license.

  EMC recommends that the FlexNet Publisher license server run on the same host as the Broker.

Process flow

Figure 2 on page 36 shows a FlexNet Publisher license server, a vendor daemon for EMC Smarts, and a permanent license that resides on the same host as the license server.

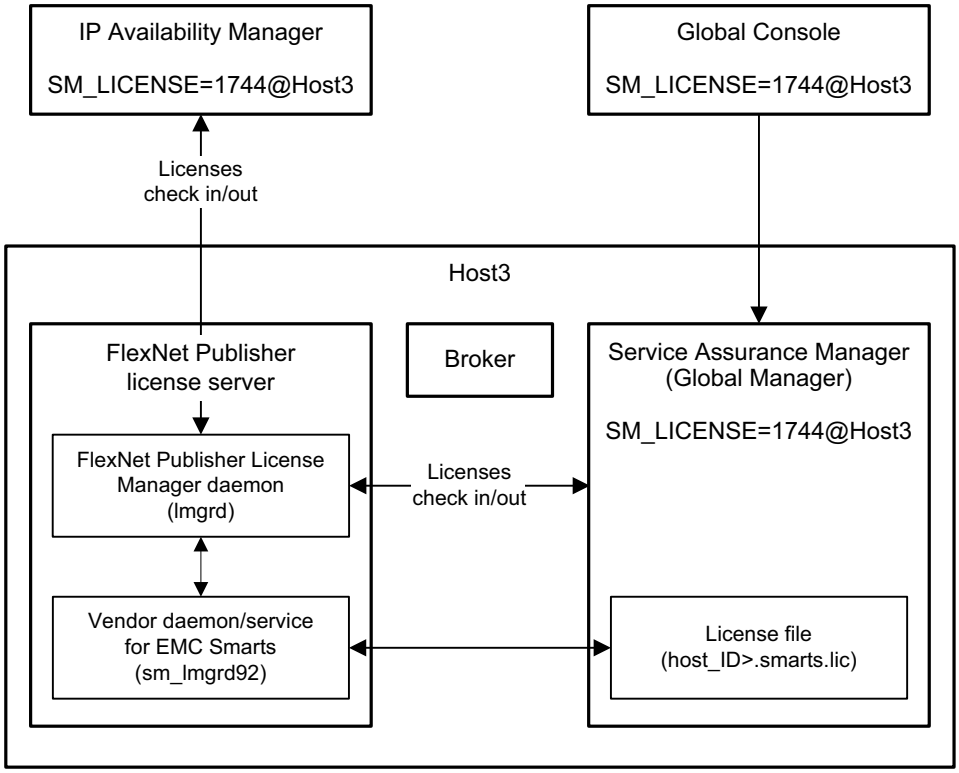

Figure 2 Process flow

For all EMC Smarts products, the SM_LICENSE variable specifies the port and host where the FlexNet Publisher license server resides.

The process flow is as follows:

1. An EMC Smarts product sends a request to the FlexNet Publisher license server for a license to access an application or a feature. In the case of a Global Console, a console relies on its Manager to send a request for licenses to access console features.
2. The FlexNet Publisher 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 EMC 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 Manager that requested the license (for example, BASEDIR/smarts/local/logs/MYDM-SA_en_US_UTF-8.log).

Examples of license files

EMC generates license files according to the FlexNet Publisher license file format. This section provides an example of a temporary license file and a permanent license file. The *FlexNet Publisher License Administration Guide* provides a more complete descriptions of file format syntax.

Temporary license file example

A temporary license file is not user-modifiable.

For illustrative purposes, the following example is an excerpt of a smarts.lic file for EMC Smarts Foundation software version 9.0.

```
FEATURE DMC_CORR sm_lmgrd92 9.0 04-dec-2008 uncounted 1C8F62B2E672 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2008
FEATURE AD_PEM sm_lmgrd92 9.0 04-dec-2008 uncounted 53BCCF735983 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2008
FEATURE AP_ICSRV sm_lmgrd92 9.0 04-dec-2008 uncounted FFE23AFEFAF8 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2008
FEATURE DMC_DECS sm_lmgrd92 9.0 04-dec-2008 uncounted 5A1809C9A688 \
   HOSTID=DEMO SN=DO315530889 START=4-nov-2008
```

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

Each FEATURE line includes the following:

- Feature name
- Vendor daemon name (sm_lmgrd92)
- Feature version
- Expiration date
- Number of concurrent licenses, which is “uncounted” in this case
- Authentication signature for the feature
- HOSTID=DEMO value that indicates that the feature is not bound to any particular host
- Serial number (SN) to identify the FEATURE line
- 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 as follows:

- Port number for the FlexNet Publisher license server.
- Directory path to the vendor daemon for EMC Smarts.
- 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.

For illustrative purposes, the following example is an excerpt of a ‹host_ID›.smarts.lic file:

```
SERVER this_host 80cdc5dc 1744
VENDOR sm_lmgrd92 BASEDIR/smarts/bin/sm_lmgrd92
USE_SERVER
INCREMENT DMC_CORR sm_lmgrd92 9.0 permanent 1 1454E7768847 \
        DUP_GROUP=SITE SN=OR54127 TS_OK
INCREMENT AP_ICSRV sm_lmgrd92 9.0 permanent 1 9639EDFD001C \
        DUP_GROUP=SITE SN=OR54127 TS_OK
```

The permanent license file consists of information for the FlexNet Publisher 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 following:
  - Value “this_host” (which is a required literal, that is, not enclosed in quotation marks).
  - Host ID of the system where the FlexNet Publisher license server resides.
  - Port number of the FlexNet Publisher 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 following:
  - Feature name
  - Vendor daemon name (sm_lmgrd92)
  - Feature version
  - Expiration date, which is “permanent” in this case
  - Number of concurrent licenses (1 for example)
  - Authentication signature for the feature
  - Duplicate grouping that indicates site license sharing
  - Serial number (SN) to identify the INCREMENT line
  - Indication of whether access to the application by using a Terminal Server window is allowed

Obtain and set up a temporary license file

"Temporary license file example" on page 37 presents an excerpt of a temporary license file. Note that an evaluation copy of the Global Console does *not* require a temporary license file.

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.

Because a temporary license file contains only *uncounted* licensed features, it does not require that a FlexNet Publisher license server be running in order to use it. An uncounted licensed feature has an associated license count of 0 (unlimited).

To obtain and set up a temporary license file:

1. Complete an evaluation request and an evaluation agreement.
2. Contact your sales representative or go to Service Center on EMC Online Support (https://support.emc.com) and create a service request indicating that you need a temporary (evaluation) license file.
3. A License will be sent to the email address included in the order.

   The name of the license file has a .lic file extension:

   a. Do not modify the license file contents.

   b. Do not copy and paste its contents to another file.
4. If required, you can rename the name of the license file:

   a. The file must have the .lic file extension

   b. Do not include a space before the .lic.
5. In the installation directory of each product that you are evaluating, do the following:

   a. Place a copy of the temporary license file (‹license_name›.lic) in the BASEDIR/smarts/local/conf directory. For example, for an IP Manager installation on a Windows host:

   ```
   C:\InCharge\IP\smarts\local\conf\<license_name>.lic
   ```

   b. Go to the BASEDIR/smarts/bin directory and enter the following command to open the runcmd_env.sh file:

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

   c. Find the **SM_LICENSE** variable and edit it to point to the license file. For example, on a Windows host:

   ```
   SM_LICENSE=C:\InCharge\IP\smarts\local\conf\<license_name>.lic
   ```

   NOTICE

   Ensure that the name and extension of the license file in the SM_LICENSE variable definition match the name and extension of the license file.

   d. Save and close the file.

The modified version of the runcmd_env.sh file is saved to the BASEDIR/smarts/local/conf directory.

e. Any Manager that was launched from this installation directory before you edited the runcmd_env.sh file needs to be restarted. Chapter 5, "Controlling Startup," describes how to start services.

For a host on which multiple EMC Smarts products are installed, you can install the temporary license in just one installation directory and then set the SM_LICENSE variable for each product to point to that one location.

How to renew a temporary license file

To renew a temporary license file, you must request another temporary license file. Follow the instructions in"Obtain and set up a temporary license file" on page 39.

Obtain and set up a permanent license file

"Permanent license file example" on page 37 presents an excerpt of a permanent license file. A permanent license file defines the number of available licenses for each application or feature, and the number of devices that an EMC Smarts manager such as EMC Smarts IP Availability Manager can discover. A permanent license file does not expire.

1. Contact your sales representative or go to Service Center on EMC Online Support (https://support.emc.com) and create a service request indicating that you need a permanent license file.

   You will receive a License Authorization Code (LAC) letter.

2. Delete any previous temporary license file in the BASEDIR/smarts/local/conf directory.

3. Determine the host id of the computer where the FlexNet Publisher license server is running:

   - UNIX: BASEDIR/smarts/bin/lmutil lmhostid
   - Windows: BASEDIR\smarts\bin\lmutil lmhostid

   The output should resemble the following:

   ```
   BASEDIR/smarts/bin/lmutil lmhostid

   lmutil - Copyright (c) 1989-2012 Flexera Software, Inc. All Rights
   Reserved.
   The FLEXnet host ID of this machine is "000c293226ec"
   ```

4. Following the instructions in the LAC letter, go to EMC Online Support (https://support.emc.com) and access the License Management home page to retrieve your license file. Provide the host id of the computer where the FlexNet Publisher license server is running.

   You will be emailed the license file.

   Use the license *as is*. Do not rename the file. Do not modify the file contents. Do not copy and paste its contents to another file.

5. Save the permanent license file, smarts.lic, in the BASEDIR/smarts directory on the host where the license server is or will be running. EMC recommends that this location be the same host and BASEDIR/smarts as for the Broker.

   NOTICE

   Do not save the permanent license file (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.

6. By default, the FlexNet Publisher license server uses port 1744 to listen for requests. If port 1744 is used by another required process on the server:
   a. Use any text editor to open the BASEDIR/smarts/smarts.lic file.
   b. Locate the line that starts with **SERVER**. This line contains the hostname, host id, and port of the license server, similar to:

      ```
      SERVER this_host 8323fcbf 1744
      ```

   c. Change the SERVER port number from the default (1744) to the required port number.
   d. Save and close the permanent license file.
7. If the FlexNet Publisher license server must communicate through a firewall to the EMC Smarts Foundation products:
   a. Configure security policies (rules) to enable a one-way connection from the FlexNet Publisher license server and the Broker to the various products.
   b. Determine the appropriate port to be used by the FlexNet Publisher license server to respond to requests. This port is used by the vendor daemon for EMC Smarts which grants and records the licenses for the software. The port number for the vendor daemon must be different from the SERVER port number.
   c. Use any text editor to open the BASEDIR/smarts/smarts.lic file.
   d. Locate the line that starts with **VENDOR**, similar to:

      ```
      VENDOR sm_lmgrd92 BASEDIR/smarts/bin/sm_lmgrd92
      ```

   e. Append the **PORT** keyword and assigned port number to the end of the **VENDOR** line in the permanent license file:

      ```
      VENDOR sm_lmgrd92 BASEDIR/bin/sm_lmgrd92 PORT=<port_number>
      ```

   f. Save and close the permanent license file.
   g. If you request licenses for additional products or additional device licenses follow instructions "Add another permanent license to a running license server" on page 45.

Rehost license

Rehost is the process of moving activated entitlements from one device or machine to a different device or machine. An entitlement is a record of the software you are entitled to use. For example, Global Console 25 Pack is represented as SMA-GC-25P.

If you are a customer who has already interacted with the EMC licensing team and obtained your license by following instructions in the LAC letter, setup for the first time, you can rehost a license without opening an Service Request with EMC.

Go to EMC Online Support and access the License Management home page. You need one of four options to search for existing installed license keys and follow the prompts to move the key to a new host. For example, choose

- **A rehost may be partial** (moving a portion of the entitlements) or
- **A rehost may be full** (moving everything on the device or machine).

For example, a full rehost would be completed when a new server is installed and replaces an existing server. The entitlements would be rehosted and locked to the new server in the licensing system by using the "Locking ID" of the new server. If you do no have LAC letter contact licensing team by opening a Service Request on https://support.emc.com.

Regenerate license

Regeneration is a process of recreating activated entitlements. If you have already interacted with the EMC licensing team and obtained your license by following instructions in the LAC letter, you can regenerate a license without emailing EMC.

Go to EMC Online Support and access the License Management home page. Click the **Activation** tab and choose **Regenerate License**. Use the online help to obtain additional information.

Install and configure the FlexNet Publisher license server

The FlexNet Publisher license server is installed using the install_license script:

- By default, the install_license script is located in the BASEDIR/smarts/script directory.
- The install_license script must be run on the same system where the license server runs.
- Administrative privileges are required.

1. If operating in a firewall environment, make sure that the ports specified in the **SERVER** and **VENDOR** lines of the permanent license file are open.
2. Change to the BASEDIR/smarts/script directory.
3. Install the FlexNet Publisher license server by using the install_license script. Specify the full path to the permanent license file.

   On UNIX systems, for example, the command is:

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

   On Windows systems, for example, the command is:

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

4. Start the FlexNet Publisher license server. You must have administrative privileges.

   UNIX systems:

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

   Windows:

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

   a. Select **Start › Settings › Control Panel › Administrative Tools**.
   b. Select **Services**.
   c. Select the **FLEXlm License Manager** service.
   d. Right-click and choose **Start**.

   When the FlexNet Publisher license server is started, it runs in the background and automatically starts the vendor daemon for EMC Smarts programs, running as:

   - UNIX: daemon
   - Windows: service

5. Verify that the FlexNet Publisher license server started by checking your Manager log file.
   a. Start your Manager, as described in Chapter 5, "Controlling Startup."
   b. Use any editor to open the Domain Manager log file. The log filename is based upon the Manager's name (for example, MYDM-SA_en_US_UTF-8.log) and the log file is located in the BASEDIR/smarts/local/logs directory.
   c. In the Manager log file, locate two messages that indicate success:
      – ICF-N-LICENSEDFEATURE-Enabling
      – ICF-N-RESTOREFINISH-PersistenceManager: restore finished

      The log filename is based upon the Manager's name (for example, MYDM-SA_en_US_UTF-8.log) and the log file is located in the BASEDIR/smarts/local/logs directory.

Use an existing FlexNet Publisher license server

You can use an existing FlexNet Publisher license server that is being used for other software applications. The FlexNet Publisher version of the license server must be version 11.9 in order to manage the vendor daemon for EMC Smarts. Consult your *FlexNet Publisher License Administration Guide* for more complete information.

1. Copy the permanent license file, smarts.lic, to the host where the FlexNet Publisher license server resides.
2. On other hosts wherever EMC Smarts software is installed, modify the SM_LICENSE variable in the runcmd_env.sh file to specify the port and host of the FlexNet Publisher license server.
   a. Go to the BASEDIR/smarts/bin directory and enter the following command to open the runcmd_env.sh file:

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

   b. Locate the line starting with **SM_LICENSE:**

      ```
      SM_LICENSE=<port>@<hostname>
      ```

   c. Specify the correct port number and hostname of the FlexNet Publisher license server, as in the following example:

      ```
      SM_LICENSE=1744@System1
      ```

   d. Save and close the file.

      The modified version of the runcmd_env.sh file is saved to the BASEDIR/smarts/local/conf directory.

Add another permanent license to a running license server

If you need to add new features or increase the number of licenses:

1. Contact your sales representative or go to Service Center on EMC Online Support (https://support.emc.com) and create a service request and provide the completed Permanent License Information Request form.

   You will receive a License Authorization Code (LAC) letter.

2. Following the instructions in the LAC letter, access EMC Online Support, Licensing option, to retrieve your license file, smarts.lic. Provide the host id of the computer where the FlexNet Publisher license server is running.

   The .lic file will be sent to the customer.

3. Use any text editor to modify the new file so that the port number and path match those in the original license file.

4. Copy the new file to BASEDIR/smarts/local/conf.

5. Force the license server to reread the license files. In the BASEDIR/smarts/bin directory, with administrative privileges, enter:

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

   - UNIX example

     ```
     BASEDIR/smarts/bin/lmutil lmreread -c 1744@localhost
     ```

   - Windows example

     ```
     BASEDIR\smarts\bin\lmutil lmreread -c 1744@localhost
     ```

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

The *FlexNet Publisher License Administration Guide* provides additional information about the lmreread utility.

Change the license server port in a running license server

Once the permanent license has been installed, you can change the port on which the FlexNet Publisher license server runs.

1. Use any text editor to open the BASEDIR/smarts/smarts.lic file.
2. Locate the line that starts with **SERVER**. This line contains the hostname, host id, and port of the license server. In this example, the port number is 1744.

   ```
   SERVER this_host 8323fcbf 1744
   ```

3. Change the SERVER port number to the required port number.
4. Save the permanent license file.
5. Enter the following command from the BASEDIR/smarts/bin directory to force the license server to restart:

   lmutil lmreread -c *<port>@<lic_host>*

6. Enter the following command from the BASEDIR/smarts/bin directory to open the runcmd_env.sh file:

   sm_edit conf/runcmd_env.sh

   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
   ```

7. Set the **SM_LICENSE** variable to the ‹port›@‹lic_host› of the FlexNet Publisher license server.
8. Save and close the file.

   The modified version of the runcmd_env.sh file is saved to the BASEDIR/smarts/local/conf directory.

   At this point, any EMC Smarts program that is started from this installation directory will use the applicable environment variables that are specified in the runcmd_env.sh file.

9. EMC Smarts programs that were previously launched from this installation directory need to be restarted for any new environment variable to take effect. Chapter 5, “Controlling Startup,” describes how to start services.

Enable communication through a firewall in a running license server

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

   ```
   SERVER this_host 80cdc5dc 1744
   VENDOR sm_lmgrd92 BASEDIR/smarts/bin/sm_lmgrd92 PORT=<port_number>
   USE_SERVE
   INCREMENT DMC_CORR sm_lmgrd92 9.0 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 with administrative privileges:

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

Use of multiple FlexNet Publisher license servers

More than one FlexNet Publisher license server may be required depending on your deployment. If there is more than one Broker for the installation, there may be good reason to use more than one license server. In general, you should run a FlexNet Publisher license server on each host that runs a Broker.

Each FlexNet Publisher 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. For each license server, perform the steps described in "Obtain and set up a permanent license file" on page 40.

In certain cases, you may use multiple FlexNet Publisher license servers in your network for the same reasons that you split a topology or choose different locations for 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 Publisher 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 is aware of the configuration details.

Upgrade licensing during EMC Smarts software upgrades

1. Before you upgrade the software, check the release notes if a new license file is required. If not, then you can continue using your existing license. If a new license is required then request one by opening an Service Request on EMC Online Support https://support.emc.com.

   You will receive a License Authorization Code (LAC) letter.

2. Upgrade the installation to the new version.

   THe FlexNet server will be upgraded with the rest of the software.

3. Modify the new license file to match the port number in the existing license file.
4. Replace the existing ‹BASEDIR›/smarts/local/conf/smarts.lic with the new license file.
5. Start the license server.
6. Start your application.

   Note: No action is required on hosts that were pointing to this license server before the upgrade.

Run older versions of license files in parallel

For some upgrades, installations, and migration scenarios, it is necessary that the FlexNet Publisher 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.

1. Install the newer software on the same system in a different directory than the older software.
2. If you have not already obtained and setup the new smarts.lic file in your BASEDIR/smarts directory on the system where the FlexNet Publisher license server will run, do so. "Obtain and set up a permanent license file" on page 40 provides additional information.
3. If you have not already installed the FlexNet Publisher license server by using the install_license script, do so. "Install and configure the FlexNet Publisher license server" on page 43 provides additional information.
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.
5. Use the sm_edit utility to open the old license file, legacy_license.lic. In this case, use sm_edit instead of any text editor because sm_edit will save the file to the correct *local* directory with the proper permissions.

   For example on Windows, issue:

   ```
   BASEDIR\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**.

     NOTICE

     The "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 Publisher 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; for example, on UNIX, **/opt/InCharge/**‹*product*›**/bin/sm_lmgrd**. For Windows, use **sm_lmgrd.exe**.

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

     Here is an example of a license file:

     ```
     SERVER this_host host_ID 1744
     DAEMON sm_lmgrd BASEDIR/smarts/bin/sm_lmgrd
     ```

7. Stop the old license server. Use the procedure in "Starting and stopping the license server" on page 52 except substitute **stop** for **start**.
8. Uninstall the old license server by using the install_license script. Replace the parameter install with uninstall, as described in "Uninstall the FlexNet Publisher license server" on page 51.
9. For the older product , modify the following to reflect the port and the system where the newer FlexNet Publisher license server is running. The value for ‹*license_host*› is the system where the newer FlexNet Publisher license server resides.

   Modify the SM_LICENSE variable in the runcmd_env.sh file to specify port **1744@**‹*license_host*›.
10. Start the new FlexNet Publisher license server, as described in "Starting and stopping the license server" on page 52.
11. To check that the FlexNet Publisher 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 Managers, by using the documentation that accompanied the older EMC Smarts software.

Overview of the install_license script

This section describes the install_license script and explains how the script installs a FlexNet Publisher license server that is provided with EMC Smarts software. The same script is also used to uninstall the FlexNet Publisher license server. The *FlexNet Publisher License Administration Guide* provides further information if you already use an existing FlexNet Publisher license server for other software applications.

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

- Installs a FlexNet Publisher license server, as described in "Install and configure the FlexNet Publisher license server" on page 43.
- Installs the permanent license file into the BASEDIR/smarts/local/conf directory. (Initially, the permanent license file is saved to the BASEDIR/smarts 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 filename 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 Publisher license server.
- For UNIX, it configures the system so that the FlexNet Publisher license server starts automatically when the system boots.
- For Windows, the install_license script creates a service for the FlexNet Publisher license server and the service starts automatically when the system boots.

NOTICE

Do not use the install_license script if you are adding more licenses or features. Instead, you should follow the instructions in "Upgrade licensing during EMC Smarts software upgrades" on page 48.

Uninstall the FlexNet Publisher license server

When you uninstall the license server, the order in which you uninstall EMC Smarts products is significant. The installation guide that accompanied your product provides instructions.

Also keep the following points in mind:

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

To stop and uninstall the FlexNet Publisher license server:

1. Stop the FlexNet Publisher license server. Use the procedure in "Starting and stopping the license server" on page 52 and substitute **stop** for **start**.
2. Execute the install_license script with the **uninstall** command from the BASEDIR/smarts/script directory. You must do so with administrative privileges.

   On UNIX systems, for example, the command is:

   ```
   BASEDIR/smarts/script> install_license.sh uninstall
   ```

   On Windows systems, for example, the command is:

   ```
   BASEDIR\smarts\script> install_license.cmd uninstall
   ```

License server administration tools

FlexNet Publisher provides several utilities for managing licenses. These include utilities for the following:

- Diagnosing licensing problems
- Forcing the license server to reread the license files
- Monitoring the status of all licensing activities including feature usage

Instructions on using these utilities can be found in the *FlexNet Publisher License Administration Guide*.

The utilities are automatically installed on:

- UNIX in the BASEDIR/smarts/bin directory.
- Windows in the BASEDIR\smarts\bin directory.

If you are not familiar with FlexNet Publisher, you may want to read the FlexNet Publisher documentation. A PDF version of the *FlexNet Publisher License Administration Guide* is automatically installed with the EMC Smarts documentation in the BASEDIR/smarts/doc/pdf directory. The name of the file is fnp_licAdmin.pdf.

Verify the FlexNet Publisher license server version

To verify the version of the FlexNet Publisher license server, use the `-v` option for the FlexNet Publisher license server command. Enter this command with administrative privileges:

```
BASEDIR/smarts/bin/lmgrd -v
```

The output will be similar to the following:

```
lmgrd v11.9.0.0 build 87342 x64_lsb - Copyright (c) 1988-2012 Flexera
   Software, Inc. All Rights Reserved.
```

Starting and stopping the license server

The FlexNet Publisher license server (lmgrd) is automatically installed in the BASEDIR/smarts/bin directory. The FlexNet Publisher license server runs in the background and automatically starts the vendor daemon for EMC. It runs as:

- Daemon on UNIX
- Service on Windows

Run the FlexNet Publisher license server at all times to ensure continuous operation of EMC Smarts software. A Manager (Global Manager or 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 Publisher 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 Publisher license server is described as follows.

One of two possible options must be supplied with the command. You must have administrative privileges to run these commands.

The **start** option starts the process; the **stop** option ends the process.

Solaris:

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

Linux:

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

Windows:

To start the license server on Windows manually, 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**).

License checking by an EMC Smarts application

When an EMC Smarts application starts up, it prints a list of features that are licensed, followed by a list of the features that are not licensed. The second list starts with *The following feature(s) are not licensed*, followed by a list of features, one per line. That list is intended to help in the resolution of licensing problems.

Volume licensing for EMC Smarts IP Availability Manager

The EMC Smarts permanent license controls the number of discovered systems. The FlexNet Publisher 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 56.

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 that are 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 that are checked out from the FlexNet Publisher license server.
- The sm_tpmgr utility shows the number of system licenses that are checked out and the number of system licenses that are used for a single 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 Publisher 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 with administrative privileges:

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

The following example shows the number of device license blocks checked out from the license server running on port 1744 on the host, host2:

```
BASEDIR/smarts/bin/lmutil lmstat -c 1744@host2 -f AP_SYSTEM_VOLUME
```

This command returns the following information:

```
lmutil - Copyright (c) 1989-2010 Flexera Software, Inc. All Rights
   Reserved.
Flexible License Manager status on Fri 11/12/2010 18:06

[Detecting lmgrd processes...]
License server status: 1744@host2
    License file(s) on host2: smarts.lic-1744:

host2: license server UP (MASTER) v10.1

Vendor daemon status (on fawn.smarts.com):

sm_lmgrd92: UP v10.1
Feature usage info:

Users of AP_SYSTEM_VOLUME:  (Uncounted, node-locked)

  "AP_SYSTEM_VOLUME" v6.2, vendor: sm_lmgrd92
  uncounted nodelocked license locked to NOTHING (hostid=ANY)

   root smarts1 /dev/tty (v3.00) (host2/1744 607031), start Tue 10/19
      18:12
   root smarts1 /dev/tty (v3.00) (host2/1744 454338), start Tue 10/19
      18:12
```

This output shows that the FlexNet Publisher license server is managing a total of five license blocks (or 250 devices). Two IP Availability Managers running on host1 have checked out three license blocks (One license block is not shown). One IP Availability Manager running on host2 has checked out two 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 with administrative privileges:

```
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 (One 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 Publisher 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. The *EMC Smarts IP Manager Reference Guide* provides additional 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.

Troubleshooting

This section describes resolutions to common issues experienced by administrators.

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

- The host ID of the system where the FlexNet Publisher license server is installed.
- The exact error message or a copy of log file. The FlexNet Publisher license server provided by EMC uses the log file BASEDIR/smarts/local/logs/flexlm.log.
- The log file used by the Manager. The log filename is based upon the Manager's name (for example, MYDM.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. An example is described in "Checked-out licenses and license blocks for a single IP Availability Manager" on page 55.
- Output of the lmstat utility with the -c and -a options. For example, invoke this command from the BASEDIR/smarts/bin directory with administrative privileges:

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

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

Evaluation software no longer runs

EMC Smarts software installed for an evaluation is no longer operational. Check the expiration date listed in the temporary license file. An example is described in "Temporary license file example" on page 37.

Messages in the flexlm.log file

The FlexNet Publisher license server provided by EMC 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 Manager checked licenses in or out.

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

The *FlexNet Publisher License Administration Guide* description of the Debug Log File provides an additional explanation of messages.

Need to determine the FlexNet Publisher 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 Publisher 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 the following command from the BASEDIR/smarts/bin directory with administrative privileges:

For example, for UNIX, invoke:

```
BASEDIR/smarts/bin/lmutil lmstat -a
```

For example, on Windows, invoke:

```
BASEDIR\smarts\bin\lmutil lmstat -a
```

The lmstat utility does not provide information about temporary ("uncounted") licenses unless you perform additional configuration. The *FlexNet Publisher License Administration Guide* provides further information and additional lmstat options.

Cannot find license error

An error message, Can't Find License, occurs in the Manager's log file (for example, BASEDIR/smarts/local/logs/MYDM.log). Check that the values specified for the SM_LICENSE variable in the runcmd_env.sh file of your EMC Smarts product are correct.

On other systems wherever EMC Smarts software is installed, verify that the SM_LICENSE variable in the runcmd_env.sh file specifies the port and host of the FlexNet Publisher license server.

1. Invoke the following command from the BASEDIR/smarts/bin directory to open the runcmd_env.sh file:

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

2. Locate the line starting with **SM_LICENSE**:

   ```
   SM_LICENSE=<port>@<hostname>
   ```

3. Verify that the line specifies the correct port number and hostname of the FlexNet Publisher license server, as in the following example. Modify the port and hostname if required:

   ```
   SM_LICENSE=1744@System1
   ```

4. Save and close the file.

   The modified version of the runcmd_env.sh file is saved to the BASEDIR/smarts/local/conf directory.

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 hostname.

To change the IP address:

1. Stop all of the EMC Smarts services.
2. Change the IP address of the system.
3. Reboot and then restart all EMC Smarts services.

NOTICE

The hostname 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 Manager. It occurs when you exceed the number of licenses listed in your permanent license file.

For example, the following error appears in the Manager's log file (for example, BASEDIR/smarts/local/logs/MYDM.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 FlexNet Publisher License
   Administration Guide,
  . available at "www.flexerasoftware.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 Managers such as IP Availability Managers and IP Performance Managers. "Permanent license file example" on page 37 provides additional information about the INCREMENT line. In order to run more IP Managers, contact your account manager or go to Service Center on EMC Online Support (https://support.emc.com) and create a service request 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 Manager's log file (for example, BASEDIR/smarts/local/logs/MYDM.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. Never attempt to edit a license file without the assistance of EMC Global Services.

Cannot restart FlexNet Publisher license server after applying a Service Pack

You shut down all running EMC Smarts processes including the FlexNet Publisher License Manager service before installing a Service Pack and then cannot restart the FlexNet Publisher License Manager service after the upgrade completes.

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

To resolve this issue, rename the file in BASEDIR/local/conf back to ‹host_ID›.smarts.lic and start the FlexNet Publisher License Manager Service.

If you have not yet installed a Service Pack, use the sm_getinfo utility as described in “Using the smgetinfo utility to save modifications” on page 30 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 Service Pack, you can retrieve the permanent license file from the backup file and restart the FlexNet Publisher License Manager service.

-

Introduction to Service Assurance Suite (SA Suite) and its licensing

This section on licensing applies only if you have upgraded from a standalone (non-SA Suite deployment) to the SA Suite. The new SA Suite licensing does not apply to a standalone install of any of the Smarts products - the existing product-centric licensing still applies. The SA Suite is a bundle of products / functionality (Smarts products, EMC M&R, and NCM) orderable as a single part number / SKU and licensing is consistent across all products in the SA Suite. Table 6 on page 61 provides information on Standard and Advanced editions of the SA Suite.

Table 6 Products - Standard / Advanced

Products	Standard/Advanced
Smarts Service Assurance Manager (SAM)	Standard / Advanced
Smarts IP Manager (IP)	Standard / Advanced
Smarts VoIP Availability Manager (VoIP-AM)	Standard / Advanced
Smarts Server Manager (ESM)	Standard / Advanced
Smarts Network Protocol Manager (NPM)	Standard / Advanced
Smarts Network Configuration Manager (NCM)	Standard / Advanced
EMC M&R	Standard / Advanced
Smarts Application Connectivity Monitor (ACM)	Standard / Advanced
Smarts SAM Adapters	Standard / Advanced
Alcatel 5620 SAM Adapters (ASAM)	Advanced Only
Smarts Multicast (MCAST)	Advanced Only
Smarts MPLS Manager (MPLS)	Advanced Only

NOTICE

Contact your account representative for more information on the SA suite. There are no differences in the licensing methodology between Standard and Advanced editions of the SA Suite. Advanced uses the same device method as Standard. All examples or references apply to both Standard and Advanced.

In the SA Suite, licensing has moved to the device scalability (number of network adapters a device has) and the number of network adapters is translated into a Managed Object (MO) count. A device can consume one (1) or more MO licenses based on the device scalability. This can be compared to a point-based licensing system.

The SA Suite looks at the number of network adapters that are on a device and the SA Suite defines a device as having 24 or less network adapters - this equates to one (1) MO. The state of the network adapters (managed versus unmanaged) is not a consideration in the SA Suite licensing because in the SA Suite, all functionality is applied to the suite-level and not at the product level. For example, with NCM as part of the suite, managing the

configuration of a device is at the device level and not at the network adapters level, hence the state of the network adapter(s) is not a governing factor in the MO license determination of a device.

SA Suite defines a device as having 24-ports/interfaces. If a device has more than twenty four (24) ports/interfaces, it counts each group of 24-ports/interfaces as a device (or MO). Hence for devices that are greater than 24-ports/interfaces, there is a conversion factor of the number of ports/interfaces on the device to a MO license count. For example, if a device has 528-ports/interfaces, the MO count for the device would be 528 / 24 = 22 MO.

For some devices such as Access Points (AP) and VoIP Phones, the calculation of the MO license count for these devices is a 10:1 conversion, meaning ten (10) APs / VoIP phones = one (1) MO license count.

All license count conversions (Device:MO) are rounded up in all calculations.

Table 7 on page 62 provides the Device:MO license count conversions used within the SA Suite:

Table 7 **Pricing model based on Managed Devices and Managed Objects**

Category	License consumption	MD:MO ratio	Calculation
Network Adapters			
• Less than or equal to 24-ports/interfaces	Each device consumes one (1) license	1:1	
• Greater than 24-ports/interfaces	Each group of 24 ports/interfaces consume one (1) license	1:x	Round-up
Servers			
Physical or Virtual	Each server consumes one (1) license	1:1	
IP Phones	Ten (10) IP phones consume one (1) license	10:1	Round-up
Access Points	Ten (10) access points consume one (1) license	10:1	Round-up

NOTICE

There is no change to the actual license files and the existing licensing technology is used. This new methodology is used only as a price calculation and auditing tool. It is not enforced by Smarts and the existing enforcement based on AP-SystemVolume licenses is still applicable. Further, the new licensing does not apply to Smarts IP. It applies to the SA Suite.

Switch/router classification and conversion examples

528-port switch / router

- If only physical ports are counted

- All Virtual Switches as part of virtual compute infrastructure are counted and charged by hypervisors
- The cost to manage the device is based on the Managed Ports and Interfaces on the devices in increments of 24.

◆ If the switch is managed:

- Each 24-ports/interfaces equate or convert to one (1) MO
- 1:X conversion in multiple of 24 physical ports/interfaces
- For a 528 port switch, calculation for MO would be:
  - Number of Managed Objects = Switch physical ports / 24 (rounded up to the next whole number)
  - Number of Managed Objects = 528/24 = 22 MO
  - Managed Devices:Managed Objects = 1:22

SA Suite license count report

This section of licensing applies only if you have upgraded from a standalone (non-SA Suite deployment) to the SA Suite. The new SA Suite licensing does not apply to a standalone install of any of the Smarts products - the existing product-centric licensing still applies.

The SA Suite has been enhanced to include the ability to report the managed devices and ports, and the corresponding managed objects (MO) count based on the point-based device counting scheme as discussed above, in a CSV format. This report utility is available as a script, so that it can be scheduled in your native operating system.

To generate the report:

1. Open the command-line interface (CLI) on the host your Smarts IP domain is installed and go to the /opt/InCharge/IP/smarts/bin folder.
2. Type the following command:

   ```
   sm_perl sm_licenseUtilization.pl -b <broker>
   ```

   For example:

   ```
   sm_perl sm_licenseUtilization.pl -b localhost:2233
   ```

This will create a report in CSV format under the ‹BASEDIR›/smarts/local/logs/ folder.

CHAPTER 5
Controlling Startup

This chapter consists of the following sections:

- Root access requirement on UNIX and Linux 66
- Overview of services 66
- About the sm_service utility 67
- sm_serviced and ic-serviced 67
- The sm_service command line 69
- Installing programs as services with sm_service 71
- Displaying installed services and their status 73
- Starting and stopping services with sm_service 74
- Modifying service parameters with sm_service 76
- Removing services with sm_service 76
- Default parameters for services 77
- Running as non root on UNIX and Linux systems 81

Root access requirement on UNIX and Linux

Super-user access is required on UNIX and Linux systems for running the following:

- sm_server - Domain Manager, handling ICMP packets, SNMP trap receiving on default port 162
- brstart - Broker on default port 426
- sm_trapd - SNMP trap receiving on default port 162

A Domain Manager must bind to multiple low numbered (less than 1024) ports for ICMP traffic. In addition, the default port for starting a Broker (426) and SNMP trap processing port (162) are also low-numbered ports. Access to low-numbered ports requires root access on UNIX and Linux systems.

You can also start a Domain Manager with root access and then open additional ports as a non-root user, if this is required. "Running as non root on UNIX and Linux systems" on page 81 provides additional information.

Overview of services

EMC Smarts Foundation programs can be installed as services. A service is a program that, once started, is intended to run continuously. A service is administered by the sm_service utility. 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 recommends that you install EMC Smarts Foundation 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 programs that should be installed as services:

- 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. "Starting and stopping services with sm_service" on page 74 describes starting and stopping services.

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, described in "Installing programs as services with sm_service" on page 71.

In addition, you can modify a service's settings, as described in "Modifying service parameters with sm_service" on page 76.

NOTICE

The sm_service utility cannot be used with the Failover System. The *EMC Smarts Failover System User Guide* provides specific information about starting, stopping, and configuring the Failover System scripts.

About the sm_service utility

The sm_service is a cross-platform utility that sets up the environment for standard EMC Smarts Foundation applications (for example, the Broker, Domain Managers, and adapters), and that installs those applications as services.

Additionally, the sm_service utility can start non-EMC 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 software (one per system), and is used from the BASEDIR/smarts/bin directory. Select the BASEDIR/smarts with the latest version of software. The utility can only be used to control services on the local machine.

sm_serviced and ic-serviced

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.

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

Thereafter, a script, ic-serviced, can be used to start and stop sm_serviced. The script is stored in a system-specific location: /etc/init.d on Solaris and Linux. The control of services can be affected through sm_service. "The sm_service command line" on page 69 provides additional information.

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.

NOTICE

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 with administrative privileges to start it:

```
/etc/init.d/ic-serviced start
```

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

```
/etc/init.d/ic-serviced stop
```

To check that sm_serviced is running and responding, issue the following command with administrative privileges:

```
/etc/init.d/ic-serviced status
```

NOTICE

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

The sm_service command line

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

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

Standard options

Table 8 on page 69 lists the standard options that can be used with the sm_service utility.

Table 8 Standard options for sm_service (page 1 of 2)

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 filename is *‹file›*, or the --logname value if *‹file›* is omitted. Log files are always placed in $SM_LOGFILES or $SM_WRITEABLE/logs.

Table 8 Standard options for sm_service (page 2 of 2)

Options	Description
--facility=*‹facility›*	Instructs the program to log any syslog messages with the specified syslog facility (UNIX/Linux only). The default is Daemon. The *‹facility›* setting can be one of the following: • Kern • User • Mail • Daemon • Auth • Lpr • News • Uucp • Cron • Local0 • Local1 • Local2 • Local3 • Local4 • Local5 • Local6 • Local7
--	Stop scanning for options.

The following example highlights the use of a standard option:

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

sm_service actions

Table 9 on page 70 lists the actions that can be performed with the sm_service utility.

Table 9 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.
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 --name=ic-broker
   --startmode=runonce BASEDIR/smarts/bin/brstart --port=1234
```

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). The sm_service then uses the information to start and run the program as a service. You must have administrative privileges to enter this command.

The syntax for the sm_service install action is:

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

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

For example:

UNIX

```
BASEDIR/smarts/bin/sm_service install --name=ic-broker
   --startmode=runonce BASEDIR/smarts/bin/brstart --port=1234
```

Windows

```
BASEDIR\smarts\bin\sm_service install --name=ic-broker
   --startmode=runonce BASEDIR\smarts\bin\brstart --port=1234
```

Install options

Table 10 on page 71 lists the install options that can be used with the sm_service utility.

Table 10 Install options for sm_service (page 1 of 2)

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 Windows platforms, the description will appear in the first column of the Service Control Manager window (the "Descriptive Name").

Table 10 Install options for sm_service (page 2 of 2)

Options	Description
--startmode=*‹mode›*	The service start policy. Where *‹mode›* is one of the following: • 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
--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. The --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.

UNIX

```
BASEDIR/smarts/bin/sm_service install --name=ic-broker
   --startmode=runonce BASEDIR/smarts/bin/brstart --port=1234

BASEDIR/smarts/bin/sm_service install --name=trapadapter
   --startmode=runonce --env=SM_BROKER=localhost:1234
       BASEDIR/smarts/bin/sm_trapd
```

Windows

```
BASEDIR\smarts\bin\sm_service install --name=ic-broker
   --startmode=runonce BASEDIR\smarts\bin\brstart --port=1234

BASEDIR\smarts\bin\sm_service install --name=trapadapter
   --startmode=runonce --env=SM_BROKER=localhost:1234
       BASEDIR\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. You must enter this command with administrative privileges:

```
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, enter the following with administrative privileges:

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

Output:

```
RUNNING ic-broker
```

For the command line of a service, enter the following with administrative privileges:

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

Output:

```
BASEDIR/smarts/bin/sm_service install
   --force
   --name=ic-broker
   --description="EMC Broker"
   --env=SM_CLIENTCONNECT=brokerConnect.conf
   --startmode=runonce
   BASEDIR/smarts/bin/brstart
   --port=426
   --restore=BASEDIR/smarts/local/repos/broker/broker.rps
   --output
```

Starting and stopping services with sm_service

The sm_service utility can be used to start and 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.

To start a program, enter the following command with administrative privileges from the ‹BASEDIR›/smarts/bin directory:

```
sm_service start [options]
```

To stop a program, enter the following command with administrative privileges from the ‹BASEDIR›/smarts/bin directory:

```
sm_service stop [options]
```

Start options

Table 11 on page 74 lists the options for the start action.

Table 11 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 12 on page 74 lists the options for the stop action.

Table 12 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 provides examples of the sm_service start and stop actions.

For the start action, enter the following command with administrative privileges:

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

For the stop action, enter the following command with administrative privileges:

```
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 13 on page 75 lists the options for the isstopped action.

Table 13 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 provides examples of the sm_service isstopped actions on UNIX. These examples also show the use of wildcards.

The isstopped action verifies whether all of the services started from the BASEDIR/smarts directory are stopped:

```
sm_service isstopped --pattern 'BASEDIR/smarts/*'
```

This example checks if all 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:

1. Select **Start › Settings › Control Panel › Administrative Tools**.
2. Select **Services.**
3. Select the EMC Smarts service.
4. Select **Start**.

NOTICE

To stop a service, select **Stop** instead of **Start**.

Modifying service parameters with sm_service

The parameters for a service can be modified with the sm_service install action. "Installing programs as services with sm_service" on page 71 provides 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
```

You must enter this command with administrative privileges.

To modify service parameters:

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>...]
```

You must have administrative privileges to enter this command.

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
```

You must have administrative privileges to enter this command.

Default parameters for services

During installation, EMC Smarts Foundation 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 EMC Smarts Foundation products, and briefly describes them. The installation guide that accompanied your product provides examples of using the sm_service install command.

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 76.

Common sm_server options

Several sm_server options are common to all Managers that are installed as services (the IP Availability Manager, for example, and the Service Assurance Manager). Table 14 on page 77 lists these sm_server options.

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

Table 14 Common sm_server options (page 1 of 2)

Options	Description
--name=*‹name›*	Specifies the name of the Manager (Manager or Global Manager). The name can contain alphanumeric characters (a-z, A-Z, 0-9) and the underscore (_) character. Do not include spaces.
--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›*	Specifies the alternate Manager port.
--bootstrap=*‹file›*	Specifies the alternate bootstrap configuration file. The file name is relative to the configuration directory (as set by the --config option). The default is bootstrap.conf.
--ignore-restore-errors	Indicates ignore errors encountered while restoring the saved topology.
--norestore	Indicates do not attempt to restore any saved objects.
--subscribe=*‹sub›*	Start a subscription adapter that automatically subscribes to the specified notifications.
--broker=*‹location›*	Specifies the alternate Broker location as host:port. Also -b *‹location›*.
--output[=*‹file›*]	Redirects 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.
--daemon	Run the program as a daemon (UNIX only). Messages generated from internal logging calls are directed to the log file. Log files are written to the BASEDIR/smarts/local/logs directory.

Table 14 Common sm_server options (page 2 of 2)

Options	Description
--facility=*‹facility›*	Instructs the program to log any syslog messages with the specified syslog facility (UNIX/Linux only). The default is Daemon. The *‹facility›* setting can be one of the following: • Kern • User • Mail • Daemon • Auth • Lpr • News • Uucp • Cron • Local0 • Local1 • Local2 • Local3 • Local4 • Local5 • Local6 • Local7
--clean-topology	Migrates the customized Polling and Threshold settings from a previous version of an EMC Smarts product to a new version of the product. Here is an example command: sm_server -n ‹name of old domain› -c icf --clean-topology --ignore-restore-errors --output --daemon After the old domain's repository file is "cleaned," that is, after all but the customized Polling and Thresholds settings have been removed from the old domain's repository file, a user places the repository file in the new domain's BASEDIR/smarts/local/repos/‹*config*› directory, to become the new domain's initial repository file. ‹*config*› refers to the configuration directory that is specified on the command line when the old domain is started.

To display a complete list of the sm_server options on your system, enter the following command with administrative privileges:

```
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 15 on page 79 lists these sm_adapter parameters or options.

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

Table 15 Common sm_adapter options

Options	Description
--name=*‹name›*	Specifies the name of the adapter.
--model=*‹name›*	Specifies the name of model library to load.
--tail=*‹path›*	Specifies read input by tailing a file.
--rserver=*‹name›*	Specifies auto-reconnect parser to Manager.
--server=*‹name›*	Specifies connect parser to Manager.
--broker=*‹location›*	Specifies the alternate Broker location as host:port. Also -b *‹location›*.
--output=*‹file›*	Directs 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.
--daemon	Run the program as a daemon (UNIX only). Messages generated from internal logging calls are directed to the log file. Log files are written to the BASEDIR/smarts/local/logs directory.
--facility=*‹facility›*	Instructs the program to log any syslog messages with the specified syslog facility (UNIX/Linux only). The default is Daemon. The *‹facility›* setting can be one of the following: • Kern • User • Mail • Daemon • Auth • Lpr • News • Uucp • Cron • Local0 • Local1 • Local2 • Local3 • Local4 • Local5 • Local6 • Local7

To display a complete list of the sm_adapter options on your system, enter the following command with administrative privileges:

```
BASEDIR/smarts/bin/sm_adapter --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 16 on page 80 lists these sm_sdi options.

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

Table 16 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 filename is relative to the configuration directory (as set by the --config option). The default is bootstrap.conf.
--nonpriv	Allow sm_sdi to run under a nonprivileged user ID. This option must be specified in order for a nonprivileged user to start sm_sdi.

To display a complete list of the sm_sdi options on your system, enter the following command with administrative privileges:

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

Running as non root on UNIX and Linux systems

Table 17 on page 81 describes two options that are used in conjunction with each other to run as non root. These two options are only available on UNIX and Linux systems. These options are available for the sm_server, brstart, and sm_adapter commands.

Table 17 Additional options for running as non root on UNIX and Linux systems

Options	Description

--privopen=*‹arg›*[,*‹arg›*]	**Note:** This option is only used in conjunction with the --run-as-user option to run as non root.
	
	Opens privileged sockets. The *‹arg›* parameter has the following syntax:
	‹type›[:*‹family›*]:*‹port›*[,#*‹count›*]
	where:
	• *‹type›* is one of the following:
	  • TCP (for a TCP connection)
	  • UDP (for a datagram)
	  • IP (for a raw socket)
	• *‹family›* is one of the following:
	  • :v4 (IPv4 address family)
	  • :v6 (IPv6 address family)
	  If *‹family›* is not specified, the address family defaults to IPv4.
	• *‹port›* is one of the following:
	  • The required privileged port for a TCP socket
	  • The required privileged port for a UDP socket
	  • The protocol for IP
	• *‹count›* is the number of sockets of the type, family and port. The default count is 1.

--run-as-user=*‹username›*	**Note:** This option is only used in conjunction with the --privopen option to run as non root. Specifies a valid user name.

The --privopen and --run-as-user options are used in conjunction to run a Domain Manager with non root privileges. The Domain Manager can startup as root, open privileged ports, and change to a non root user.

You can open multiple ports by repeating the *‹arg›* parameter. Each instance is separated by a comma (,). For example:

```
--privopen=UDP:v4:161,#2
--privopen=IP:1,#6
--privopen=IP:v6:58,#6
--run-as-user=testuser1
```

In this example:

- The first privopen line opens two UDP IPv4 sockets on port 161.
- The second privopen line opens six raw IPv4 sockets for ICMP.
- The third privopen line opens six raw IPv6 sockets for ICMPv6.

After the sockets are opened, the process will change to run as user, “testuser1.”

CHAPTER 6
Securing Access to the Software

This chapter consists of the following sections:

- Security features 84
- How security works 85
- Configuring authentication 88
- Controlling authentication 100
- Changing Cryptographic Settings 101
- Encryption 101
- Encrypted passwords 104
- Encrypted seed files 106
- Encrypted connections 106
- Replacing or importing TLS certificates 110
- Configuring a secure Broker 110
- Additional safeguards 112
- Example security configurations 113

- Changing Cryptographic Settings 101
- Encryption 101
- Encrypted passwords 104
- Encrypted seed files 106
- Encrypted connections 106
- Replacing or importing TLS certificates 110
- Configuring a secure Broker 110
- Additional safeguards 112
- Example security configurations 113

NOTICE

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.

Security features

EMC Smarts software provides multiple levels of security which 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 by 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. "Configuring authentication" on page 88 provides detailed information.

- Encrypted passwords

  Passwords can be encrypted in the serverConnect.conf, clientConnect.conf, and brokerConnect.conf files, which are located in the BASEDIR/smarts/conf directory of each EMC Smarts software installation. Encryption is based on a secret phrase, common to all of the EMC Smarts applications that must interact, and is used to encrypt password fields in the authentication records. "Encrypted passwords" on page 104 provides detailed information.

  Passwords can also be encrypted in seed files. "Encrypted seed files" on page 106 provides detailed information.

- Encrypted connections

  The software provides several levels of security above cleartext communication:

  - Encryption based on the site secret
  - Diffie Helman-Advanced Encryption Standard (DH-AES)
  - DH-AES used in conjunction with the site secret
  - Transport Layer Security (TLS) v1.2 with or without Federal Information Processing Standard (FIPS) Publication 140-2 validated cryptography.

  The levels of security are configured by way of values for environment variables. "Encrypted connections" on page 106 provides detailed information.

The security features are enabled with default settings when you install the 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 the software. In addition, you should restrict access to the security configuration files, as described in "Specifying alternate security configuration files" on page 98.

NOTICE

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

How security works

EMC 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 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:

- Encryption based on the site secret
- Diffie Helman-Advanced Encryption Standard (DH-AES)
- DH-AES used in conjunction with the site secret
- Transport Layer Security (TLS) v1.2 with or without Federal Information Processing Standard (FIPS) Publication 140-2 validated cryptography.

In Foundation 9.2, a FIPS 140 enabled Broker is able to communicate with FIPS 140 enabled Domain Managers as well as non FIPS 140 enabled Domain Managers. In addition, a FIPS 140 enabled Domain Manager is able to communicate with FIPS 140 enabled Domain Managers as well as non FIPS 140 enabled Domain Managers.

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?

NOTICE

For the UNIX account that is used to run EMC Smarts processes, if the system administrator sets explicit permissions for the file mode creation mask (the umask) for a parent process, EMC Smarts software respects those permissions. For processes started by the sm_serviced program that is described in "sm_serviced and ic-serviced" on page 67, the umask permission is inherited.

Server and client programs for the software

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

Table 18 on page 86 lists some server and client programs and shows how they can interact.

Table 18 EMC 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
	Beacon
	Application Connectivity Monitor
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
Beacon	Global Console
Application Connectivity Monitor	Global Console

NOTICE

The 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.

NOTICE

The server and client programs that are listed in Table 18 on page 86 can be installed on the same host. For example, it is common for the 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 an EMC Smarts username and password. A server authenticates a connection request by comparing the EMC 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 reread for each attempted connection so any changes to the file will take effect immediately.

The Global Console always displays a log-on dialog box where a user must enter a username and password. Other 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 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 that are 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 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 that runs the client and whose target matches the name of the server that is being connected to. Once it finds a match, the client sends the 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 reread for each attempted connection. As such, you can edit the configuration files at any time and the changes take effect immediately.

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

For example, when a Domain Manager (such as an EMC Smarts IP Availability Manager) registers with a Broker, the Domain Manager sends a 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 Domain Manager is alive (by pinging the manager), the Broker must authenticate with that Domain Manager. To do this, it finds a username and password in its brokerConnect.conf file to send to the Domain Manager. The Domain 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, non-FIPS 140 mode encryption is enabled during the installation process. All EMC Smarts products and applications that interact with each other must share a common secret phrase, and the same FIPS 140 mode. The security configuration files and files with passwords to SNMPv3 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 EMC 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 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 the following:

- 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.

NOTICE

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 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. Appendix B, "Wildcards," provides additional information about wildcards.

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 username and password, and a connection privilege. To modify this file, use the sm_edit utility as described in "Modifying EMC Smarts files" on page 20.

NOTICE

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>:<ITOps username>:<password>:<privilege>
```

Table 19 on page 90 describes the four fields of an authentication record in the serverConnect.conf file.

Table 19 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 applies only to the Broker. • ~‹BROKER› indicates this record applies to all servers except the Broker.
‹EMC Smarts username›	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 user requesting a connection.	Can be a password or one of the following special values: • ‹SYS› indicates that the username must be a valid login name on the local system. The server passes the credentials to the host operating system for validation. • ‹DEFAULT› is provided for legacy clients that cannot send a password. • ‹AUTO› is provided for site-specific credentials.
‹privilege›	Access privileges of the client.	Valid values include: • All • Monitor • None • Ping

Remember that during the authentication process the server receives a connection target, EMC 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, and authentication features provide 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. The *EMC Smarts Service Assurance Manager Configuration Guide* provides additional information about using user profiles to restrict access to console operations.
- *Monitor* access supports a monitoring console. A monitoring console cannot change server database or configuration parameters except in special circumstances such as acknowledgement of 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 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 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 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.

NOTICE

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>:<ITOps username>:<password>
```

Table 20 on page 92 describes the four fields of an authentication record in the clientConnect.conf and brokerConnect.conf files.

Table 20 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 applies only to the Broker. • ~‹BROKER› indicates this record applies to all servers except the Broker.
‹EMC Smarts *username›*	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 EMC Smarts username. • ‹PROMPT› indicates that the client program asks the user to provide an EMC 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 an EMC 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 Manager may need to run with root privileges because it sends ICMP pings or receives SNMP traps.
- A Domain Manager is as a Broker, and the Broker is listening on port 426. Port 426 is a privileged port, meaning the process must be running as root to open the port. Ports numbered below 1024 require root privileges.

  NOTICE

  A process, without being root, can connect to a process listening on a port below 1024.

- A user's system login name must correspond to an EMC Smarts username in the clientConnect.conf file or an EMC Smarts username and password will not be sent to a server for authentication. In the clientConnect.conf record, a user's login name and EMC Smarts username do not have to be identical.

Running without root privileges

All Domain Managers do not need to send ICMP pings or receive SNMP traps and, therefore, do not need to run with root privileges.

- Start a Broker on a port that is not privileged. The software registers the following non privileged ports:
  - Broker is port 426
  - Domain Manager is port 4426

  The Broker could be started on port 4426 if a well-known port is required.
- Domain Managers can startup as root, open privileged ports, and then change to a non root user, as described in "Running as non root on UNIX and Linux systems" on page 81.

Special authentication values

Special authentication describes the configuration of the serverConnect.conf, clientConnect.conf and brokerConnect.conf by 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. By using this method, you give every console operator an account on the host on which the server runs. The operators log in by using the username and password defined for that account. (The account can be disabled to prevent actual interactive access to the system.) By using this method:

- Each console operator has a unique username and password.
- Accesses to the system can be traced to a particular user.
- 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 example would grant the users "fred" and "george" All access, once they provide the passwords for their accounts on the host. 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 nonconsole applications to prompt, the value ‹PROMPT› must be used for the EMC Smarts username and password in clientConnect.conf.

Network account authentication

In UNIX systems, the system authenticates a username based on its password. In 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 nonconsole 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 EMC Smarts username and password, the client generates a password by using its secret. The server validates the password to accept or reject the connection.

NOTICE

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

So for connections to a particular Domain Manager (such as 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 Domain Manager.

NOTICE

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 software, which may not support authentication, EMC 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 software

The 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. “Legacy system authentication” on page 95 provides additional information on the ‹DEFAULT› value.

```
<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 nonsecure Broker, equivalent in security to EMC Smarts software prior to version 5.0.

Automating client authentication to the Broker

EMC Smarts programs connect to the 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 EMC 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 nonsecure configuration for the Broker. In this configuration, consoles do not prompt for a username and password when connecting to the Broker. "Configuring a secure Broker" on page 110 provides additional information regarding a secure Broker configuration.

Automating Broker authentication to servers

The 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
```

"Connection privileges" on page 91 provides additional information about the Ping access permission.

Authenticating administrative users

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

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

This record authenticates clients that provide the EMC 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.

NOTICE

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. Replace the admin user only *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 by 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").

"Connection privileges" on page 91 provides additional information about the All access permission.

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 EMC Smarts username

"maint" and the password "maint".

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

The second console record authenticates console users who provide the EMC 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.

NOTICE

User profile restrictions can be used to further limit certain Global Console operations, as described in the *EMC Smarts Service Assurance Manager Configuration Guide*.

EMC recommends that you create separate accounts for each operator. 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 enter a valid system username and password. For authentication, the system username becomes the EMC Smarts username. This record provides monitoring privileges.

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

Authenticating users interactively

This authentication record in the clientConnect.conf prompts users to enter an EMC 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 by 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. Noninteractive programs skip this record because they cannot prompt for information.

This record could match the following serverConnect.conf authentication records, as described in the following:

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

In "Authenticating Global Console users" on page 97:

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

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

In "Authenticating users with a valid system account" on page 97

```
* : * : <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/smarts, 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 filename, 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. You must have administrative privileges when typing the following command.

   ```
   BASEDIR/smarts/bin/sm_service show --cmdline ic-am-server
   ```

   Output:

   ```
   BASEDIR/smarts/bin/sm_service install
      --name=ic-am-server
      --description="EMC Smarts IP Availability Manager"
      --startmode=runonce
      BASEDIR/smarts/bin/sm_server
      --name=INCHARGE-AM
      --config=icf
      --bootstrap=bootstrap-am.conf
      --port=0
      --subscribe=default
      --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.

   ```
   BASEDIR/smarts/bin/sm_service install
      --force
      --name=ic-am-server
      --description="EMC Smarts IP Availability Manager"
      --startmode=runonce
      --env=SM_SERVERCONNECT=BASEDIR/smarts/conf/serverConnect_IP.conf
      BASEDIR/smarts/bin/sm_server
      --name=INCHARGE-AM
      --config=icf
      --bootstrap=bootstrap-am.conf
      --port=0
      --subscribe=default
      --ignore-restore-errors
      --output
   ```

3. Do one of the following:

   - Start the program.
   - 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 the 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 EMC 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, the software treats this the same as ‹STD›.

NOTICE

The angle brackets (‹ ›) are required.

When SM_AUTHORITY is set to ‹NONE›, the 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:

1. Go to the BASEDIR/smarts/bin directory and enter the following command to open the runcmd_env.sh file:

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

2. Use the following syntax to add the **SM_AUTHORITY** variable and its value:

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

3. Save and close the file.

   The modified version of the runcmd_env.sh file is saved to the BASEDIR/smarts/local/conf directory.

   At this point, any EMC Smarts program that is started from this installation directory will use the applicable environment variables that are specified in the runcmd_env.sh file.

4. EMC Smarts programs that were previously launched from this installation directory need to be restarted for any new environment variable to take effect. Chapter 5, "Controlling Startup," describes how to start services.

Changing Cryptographic Settings

Changing Pseudo-random Number Generators (PRNG) algorithm

The Smarts Global Console SSL implementation uses ECDRBG128 (Dual EC DRBG) as the default PRNG algorithm. Following NIST's decision to recommend against the use of Dual_EC_DRBG, it is recommended that you discontinue the use of Dual EC DRBG, and move to a different PRNG (HMACDRBG128) at this time.

Changing the use of the default PRNG algorithm:

Add the following line in the java.security file located at BASEDIR/_jvm/lib/security/ directory.

- com.rsa.crypto.default.random = HMACDRBG128

This change needs to be applied to the following components:

- "Smarts Global Console
- "Any other custom programs written by you using DMT Java remote API

Encryption

By default, encryption (non-FIPS 140 mode) is enabled during the installation process. The basis for encryption is a secret phrase that gets transformed into the file imk.dat.

The transformation of the site secret into the imk.dat files differs, depending on whether encryption is set to FIPS 140 mode. EMC Smarts programs using FIPS 140 mode encryption cannot use the same imk.dat file as EMC Smarts programs using non-FIPS 140 mode. Therefore, all clients and servers using an imk.dat file must all be set to the same FIPS 140 mode.

The programs use the 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 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/smarts/local/conf directory.

NOTICE

The imk.dat file can only be copied to other systems with the same operating system (OS), OS version, and FIPS 140 mode setting.

To raise the level of security, you need to change the secret phrase by using the sm_rebond utility. Thereafter, the secret phrase should be changed periodically to maintain a secure system.

NOTICE

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 that run on the same host and use the same imk.dat file.

NOTICE

The sm_rebond utility shuts down all of the processes run from the product that were started by using sm_service or sm_serviced, and that use the same imk.dat file. The sm_rebond utility 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. You must enter this command with administrative privileges.

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

   For the *‹base directory›*, enter the path for the directory in which the software that is to be rebonded is installed.

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.

NOTICE

The sm_rebond utility encrypts only files that reside in BASEDIR/smarts/local/conf and three levels of subdirectories below that. To encrypt files outside of that directory area, contact EMC Global Services.

NOTICE

When using the sm_rebond utility to change the default password for SNMPvV3 password encryption, ensure that all the passwords for the Domain Managers in your deployment are the same.

Using the secret for multiple products and sites

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

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

   NOTICE

   The secret phrase must be exactly the same for each product 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 products as needed.
5. Restart the Managers if applicable.

Using the secret across large numbers of sites

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

1. Install an EMC Smarts product 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 product as needed.
4. Copy the imk.dat file, as well as the configuration files and seed files to the other installations of the software.

NOTICE

The imk.dat file can only be copied to other systems with the same operating system (OS), OS version, and FIPS 140 mode setting.

You will not have to use the sm_rebond utility on the other Managers if the files are moved to the Managers before the Managers are started.

Locking the secret

If access to encrypted data is a critical issue for your organization, EMC 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.

NOTICE

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

Encrypting passwords

Password is the only field in the security configuration files that is encrypted. The configuration files are located in the BASEDIR/smarts/local/conf directory of a product.

To type or change a password in a security configuration file:

1. Go to the BASEDIR/smarts/bin directory and enter the following command to open the security configuration file:

   ```
   sm_edit conf/<security configuration file>
   ```

   For example, to open the serverConnect.conf file:

   ```
   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
   Smarts     :admin     :<SYS>     :All
   Smarts-WP  :user5     :Password  :Ping
   Smarts     :operator  :Password  :Monitor
   Smarts     :*         :<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 and close the file.

The modified version of the security configuration file is saved to the BASEDIR/smarts/local/conf directory.

The appropriate values are encrypted as part of the process.

5. Repeat the steps for all of the security configuration files.
6. Restart the Managers if applicable.

If a security configuration file already includes the first line for encryption, specify the sm_edit utility with the --noedit option to encrypt the file without opening it. You must have administrative privileges when entering this command.

```
BASEDIR/smarts/bin/sm_edit --noedit conf/<security configuration file>
```

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, specify the sm_edit utility with the --noedit option to encrypt the file without opening it. You must have administrative privileges when entering this command.

```
BASEDIR/smarts/bin/sm_edit --noedit conf/<security configuration file>
```

Encrypted connections

Software components communicate over TCP connections by using the EMC Smarts Remote API. Clients which are 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 by using man-in-the-middle configurations.

You can encrypt certain connections by using different keys for the Advanced Encryption Standard based on a combination of the Diffie Helman standard and the site secret associated with the installation. "Encryption" on page 101 provides additional information about the site secret.

Encrypted connections do not work with EMC Smarts Adapter for NetIQ AppManager.

Levels of encryption

The software provides several levels of security above cleartext communication as shown in Table 21 on page 107 and Table 22 on page 107.

Table 21 Encryption levels for connections for non-FIPS 140 mode

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
Not applicable	TLSv1.2	• Standards-based	• Incompatible with previous releases of the software • Slower than cleartext

Table 22 Encryption for connections for FIPS 140 mode

Description	Advantages	Disadvantages
TLSv1.2 with FIPS 140 cryptography	• Standards-based • Compatible with FIPS 140-2 compliant cryptography	• Incompatible with previous releases of the software • Slower than cleartext

Any encryption based on the site secret should only be used once the secret phrase has been changed by using sm_rebond described in "Changing the secret" on page 102. The Global Console supports cleartext (Level 0), Diffie Helman-Advanced Encryption Standard (Level 1), and TLSv1.2 encrypted connections.

Encrypting connections in non-FIPS 140 mode

In Foundation 9.2 onwards, Domain Managers and Brokers running in non-FIPS 140 mode can interact with Domain Managers and Brokers running in non-FIPS 140 mode or FIPS 140 mode. Although a Domain Manager running in non-FIPS 140 mode can interact with a Domain Manager running in FIPS 140 mode, the two applications cannot interoperate with one another because they cannot share encrypted data such as SNMPv3 credentials.

EMC Smarts applications that are based on 9.1 foundation or later software, and running in non-FIPS 140 mode use TLSv1.2 encrypted connections by default. This behavior is different than the behavior for EMC Smarts applications that are based on pre-9.1 foundation software, where the encryption for encrypted connections was configured by using two environment variables, SM_INCOMING_PROTOCOL and SM_OUTGOING_PROTOCOL. These environment variables are ignored if the communicating applications are TLS-capable, but are honored when a TLS-capable application is interacting with a non-TLS-capable application.

For pre-9.1 applications that are operating in non-FIPS 140 mode, encrypted connections are configured by using two environment variables:

- SM_OUTGOING_PROTOCOL— Controls the connections that an EMC Smarts application that is acting as a client is allowed to initiate.
- SM_INCOMING_PROTOCOL— Controls the connections that an EMC Smarts application that is acting as a server is allowed to accept.

Each of these environment variables can contain a list of security levels, except for TLS security, which cannot be controlled by using these variables. 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, such as when 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 an EMC Smarts product, edit the runcmd_env.sh file, which is located in the BASEDIR/smarts/local/conf directory of that product.

1. Go to the BASEDIR/smarts/bin directory and enter the following command to open the runcmd_env.sh file:

   ```
   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 and close the file.

   The modified version of the runcmd_env.sh file is saved to the BASEDIR/smarts/local/conf directory.

5. Restart the Managers if applicable.

Encrypting connections in FIPS 140 mode

In FIPS 140 mode, the software uses Transport Layer Security v1.2 and ignores the SM_INCOMING_PROTOCOL and SM_OUTGOING_PROTOCOL settings. Because of this, in Foundation pre-9.1 (but not in 9.1 or later versions), a FIPS 140-enabled Domain Manager cannot interact with a non FIPS 140-enabled Domain Manager.

Setting the environmental variable SM_FIPS140 to true enables FIPS 140 mode. "SM_FIPS140" on page 154 provides additional information.

You must enable FIPS 140 mode before the Broker and Domain Managers are started.

Suggested encrypted connections in non-FIPS 140 mode

EMC 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, or in TLSv1.2, if a client only supports cleartext. 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 do not need to support cleartext if all clients can make encrypted connections.
- If a Domain Manager must connect to a client that only supports cleartext, set SM_OUTGOING_PROTOCOL to cleartext as well as encryption.

- Adapters based on Foundation 9.2 onwards use TLSv1.2 to communicate by default. Only adapters that register with the Broker (--name option) can accept incoming connections. If you have adapters that accept incoming connections from clients that are not TLS-capable, setting SM_INCOMING_PROTOCOL to require encryption is appropriate.

  Also, if the adapter must connect, or be connected, to clients that support only cleartext, then add the cleartext option to the appropriate variable.

- Configure any components that must run on networks outside the management domain 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 by using a console.

Replacing or importing TLS certificates

By default, a Smarts Domain Manager generates a self-signed TLS certificate that it will present to the incoming TLS connections. However, you can configure a certificate to use by performing the following step:

- "Place the certificate and key files in $SM_SITEMOD/conf/tls/$DOMAIN.crt and $SM_SITEMOD/conf/tls/$DOMAIN.key

Where $DOMAIN is the name of the domain used to register with the broker. The files must be in PKCS#8, PEM encoded form with no password.

Self-signed certificates generated by Smarts are stored in the process memory. To regenerate default self-signed TLS certificates, stop and restart Smart processes.

Configuring a secure Broker

You can configure the Broker to run in a secure manner. Use of a secure Broker results in the following changes to how the software runs:

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

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

1. Choose a unique EMC Smarts username and password for the secure Broker credentials. The new username and password will be used by both servers and clients:
   - Servers will use these credentials to register with the Broker.
   - Clients will use these credentials to connect to the Broker and determine the location of a server.

For example, you could use the username “SecureBroker” and the password “Secure”. Choose a unique EMC 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 EMC Smarts programs send the SecureBroker/Secure credentials when connecting to the Broker.

   a. Comment out the following line:

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

   b. Type a new line configuring a secure Broker. This new line is added below the BrokerNonsecure line that you commented out. For example:

   ```
   #*:<BROKER>:BrokerNonsecure:Nonsecure
   *: <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. This new record must be below the BrokerNonsecure/Nonsecure record. For example:

   ```
   <BROKER>:BrokerNonsecure:Nonsecure:Ping
   <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 EMC Smarts applications requiring access to the authentication records.

Limiting access to the configuration files

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 EMC Smarts software, which should be edited after installation, are readable by anyone.

Because Managers and the Broker typically run with administrative privileges, the serverConnect.conf and brokerConnect.conf files should only be readable by users with administrative privileges 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 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_SNMP_BUG_COMPATIBLE" on page 151, 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 EMC 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 EMC 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 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 Transport Layer Security v1.2, Diffie Helman encryption, and cleartext. With the default security, you need to change the passwords for the usernames in the security configuration files. Default security prevents casual inspection, but does not pose much of a deterrence from more strenuous efforts.

NOTICE

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 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. These operators log in by 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 EMC 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.

To configure customized security:

1. Create a new secret phrase by using sm_rebond, as described in “Changing the secret” on page 102.
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 93. The new usernames and passwords should also address connections between servers and servers or servers and clients.

   If your software resides on a single host, a single username and password for automatic client authentication is usually sufficient. However, if you have clients, such as adapters, on multiple hosts, or generally in different security domains, you may 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 to grant 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.

NOTICE

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 components to communicate with encrypted connections. The types of allowed connections depend on the versions of EMC Smarts software in use.
   - Most connections should be encrypted. The Broker does not need to support cleartext if all clients can make encrypted connections. "Encrypted connections" on page 106 provides additional information.
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 110.

CHAPTER 7
Operation of the Broker

This chapter consists of the following sections:

- Overview of the Broker 118
- Viewing the registry of the Broker 119
- How the clients find the Broker 119
- How to change the Broker environment variable 120
- Securing the Broker 121

Overview of the Broker

A client application, such as a console or an adapter, utilizes the Broker to determine where Managers are running. When a Manager starts, it registers with the broker:

- Hostname of the machine that it is running on.
- TCP port that it is listening on.

Clients retrieve this information from the Broker so that they can communicate with the Manager.

The Broker registry maintains the following information:

- Name of the Manager, including the host and TCP port it is running on.
- Status of each Manager.

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

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

The Broker changes the status of the Manager to Dead when it cannot connect to the Manager. However, the Broker continues to try to connect to the Manager every five minutes. If the Broker succeeds in connecting to the Manager, it changes its status back to Running.

Viewing the registry of the Broker

The Broker is a component of an EMC 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.

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 clients find the Broker

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

A client performs the following steps to determine the Broker's location.

1. Verifies whether 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.

NOTICE

The hostname 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 the software is installed.

- If during installation, the default Broker host and/or port settings were changed, but the typed values are incorrect.
- If the Broker is automatically installed when you install an EMC Smarts Foundation 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 an EMC Smarts application is installed on one host and the 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>
   ```

   where:

   - *‹host›* is the location of the Broker.
   - *‹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 EMC Smarts Foundation 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
   ```

   Output:

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

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
   --name=ic-broker
   --description="EMC Broker"
   --env=SM_CLIENTCONNECT=brokerConnect.conf
   --startmode=runonce
   BASEDIR/smarts/bin/brstart
   --port=<NEW VALUE>
   --restore=BASEDIR/smarts/local/repos/broker/broker.rps
   --output
```

Securing the Broker

Securing access prevents unauthorized usage of the Broker, and protects it from being modified (for example, Managers being deleted from the Broker). Authenticating users and determining their privileges are accomplished by using 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. "Configuring a secure Broker" on page 110 provides additional information.

CHAPTER 8
Continuous Audit of Client Connections

This chapter describes a framework for exposing information about client connections, threads, and other aspects of EMC Smarts applications. It consists of the following sections:

- Overview of continuous auditing of client connections 124
- System Element objects 125
- Client data persists after connection terminates 128
- File descriptor usage warning event 128
- Acquiring client information 129
- Viewing SM_ElementManager attribute values 130
- Changing SM_ElementManager attribute values 131

Overview of continuous auditing of client connections

The foundation software on which EMC Smarts applications are built is essentially a sophisticated and specialized object database. It consists of MODEL class libraries, tools, commands, and utilities.

By exposing information about client connections, threads, and other aspects of an EMC Smarts application as MODEL objects in the repository, the foundation's tools, such as multi-language client libraries, subscription facilities, and analysis can be used to query and report on the application itself. A client connection is any connection that is initiated by a remote client, such as a Global Console, remote repository accessor, data exchange adapter, dmctl, Perl client, Java client, or C++ client.

The following enhancements are intended to improve the serviceability of EMC Smarts applications by making more information about client connections easily accessible through normal foundation mechanisms.

To disable logging, do the following:

1. Open the `runcmd_env.sh` file where the broker was started using the command line.
2. Run `sm_edit:<Basedir>/smarts/bin/sudo ./sm_edit <Basedir>/local/conf/runcmd_env.sh.`
3. Add or edit the following line in the `runcmd_env.sh` file:

   ```
   SM_NO_LOG_CLIENTS=TRUE
   ```

4. Save and close the `runcmd_env.sh` file.
5. Restart the Smarts broker service.

System Element objects

Two MODEL classes, SM_SystemElement and SM_ElementManager, are used to expose information about an application's client connections, threads, and queues. SM_ElementManager is responsible for managing the subclasses of SM_SystemElement.

SM_SystemElement

The SM_SystemElement class is an abstract base class for the SM_Client, SM_CThread, and SM_Queue classes, which are defined in Table 23 on page 125 and shown in Figure 3 on page 125.

Table 23 SM_SystemElement subclasses

Subclass name	Description
SM_Client	An external representation of an incoming client connection. It has attributes that represent information about the connection. An SM_Client is served by SM_CThreads and SM_Queues.
SM_CThread	An external representation of the internal thread that is associated with a client connection. It has attributes that represent information about the internal thread.
SM_Queue	An external representation of the internal queue that is associated with a client connection. It has attributes that represent information about the internal queue.

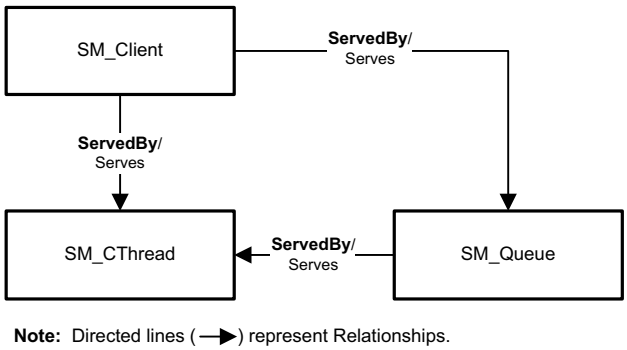

Figure 3 SM_SystemElement modeling

An SM_Client object and an SM_CThread object are created whenever a client successfully establishes an API connection to the application. A client's SM_Queue object and a second SM_CThread object are created when and if a subscription request is made by the client.

SM_ElementManager

The SM_ElementManager class manages the life cycle of SM_SystemElement objects. It is a singleton class (the instance is SM_ElementManager::SM-ElementManager) and has attributes for configuring various aspects of the life cycle management. The key attributes for SM_ElementManager are listed in Table 24 on page 126.

Table 24 Key attributes for SM_ElementManager (page 1 of 2)

Attribute	Value	Description
Enabled	Boolean Default: TRUE	Enables or disables the tracking of client connections to this application: • A value of TRUE enables the tracking of client connections. • A value of FALSE disables the tracking of client connections.
RecentHighestFD	Unsigned long integer This attribute is read-only.	Highest file descriptor (highest socket number) in recent use, for a client connection to this application.
FDLimit	Unsigned long integer This attribute is read-only.	Maximum number of file descriptors (socket numbers) that this application is allowed to use for client connections at any one time. For an application on a Solaris- or Linux-based host, the file-descriptor limit is imposed on the application by the OS *as long as that limit is 1024 or less*. If that limit is more than 1024, the foundation startup code sets the limit to 1024. FDLimit is not supported for Windows.
FDWarningHeadroom	Unsigned integer Default: 100	An integer in the following equation that determines when the number of file descriptors for this application increases to an unacceptable level: Number of file descriptors › FDLimit – FDWarningHeadroom For example, if FDLimit = 1024 and FDWarningHeadroom = 100, the number of file descriptors increases to an unacceptable level when that number is greater than 924. FDWarningHeadroom is not supported for Windows.

Table 24 Key attributes for SM_ElementManager (page 2 of 2)

Attribute	Value	Description
ReapInterval	Unsigned long integer Default: 60 seconds	Interval, in seconds, between successive expiry scans of terminated client connections for this application. This attribute works in conjunction with two other attributes, ClientRetainTime and AbnormalClientRetainTime, to determine when an expiry scan deletes (reaps) the client objects and data for a terminated connection: • For any normally terminated connection, the expiry scan deletes the associated client objects and data when the time between the connection's termination timestamp and the current time exceeds the ClientRetainTime. • For any abnormally terminated connection, the expiry scan deletes the associated client objects and data when the time between the connection's termination timestamp and the current time exceeds the AbnormalClientRetainTime. The effective retain time for a normally terminated connection is 1/2 the ReapInterval plus the ClientRetainTime. The effective retain time for an abnormally terminated connection is 1/2 the ReapInterval plus the AbnormalClientRetainTime.
ClientRetainTime	Unsigned long integer Default: 60 seconds	The least amount of time, in seconds, between when a normally terminated connection ends and when the client objects and data that are associated with that connection are deleted from the repository.
AbnormalClientRetainTime	Unsigned long integer Default: 900 seconds (15 minutes)	The least amount of time, in seconds, between when an abnormally terminated connection ends and when the client objects and data that are associated with that connection are deleted from the repository.
RequestLogSize	Unsigned long integer Default: 25	Size of the request log for client connections for this application. By default, the 25 most recent requests and their responses are recorded. Setting RequestLogSize to 0 disables the collection of request information for client connections. Like the rest of the client data, the request information persists for a client connection until the associated SM_Client and its threads and queues are deleted.
LogClients	Boolean Default: TRUE	TRUE indicates that this application will log the client connects and disconnects in its log file.

Client data persists after connection terminates

When a connection terminates, the SM_ElementManager does not immediately delete the associated client objects and client data. Instead, the SM_ElementManager does a periodic scan and removes the client objects and the client data only after a certain amount of time has passed. *In this way, you, as an administrator, can examine details about a connection even after it has terminated*.

File descriptor usage warning event

The SM_ElementManager receives the file descriptor (socket number) for a new client connection. On Solaris and Linux hosts, there is a hard limit to the number of file descriptors that an application can use at any one time. Exceeding that limit is an irrecoverable condition that forces the application to exit with a fatal error.

SM_ElementManager keeps track of the file descriptors and uses its FDLimit and FDWarningHeadroom attribute values to determine when the number of file descriptors increases to an unacceptable level. When that limit is exceeded, the SM_ElementManager asserts the HighFDUsageWarning event.

NOTICE

The HighFDUsageWarning event is not supported for Windows.

Acquiring client information

You can get information about the clients for an application by going to the BASEDIR/smarts/bin directory of the application and entering dmctl command that include any of the following command options:

- **who** command option
  - **who**

    Lists the clients that are currently logged in to the application.
  - **who** ‹*n*›

    Provides information about the client with connection ID *n*.
  - **who is** ‹*n*›

    Provides information about the client with connection ID *n*.
  - **who am I**

    Provides information about the dmctl client that you am using.
  - **who is** [**ACTIVE** | **ENDED** | **FAILED**| **KILLING** | **KILLED** | **ABNORMAL**| **ALL**]

    Lists the clients whose state matches the one specified. The state names are not case-sensitive; for example, **who is ACTIVE** and **who is active** are both acceptable.
- **w** command option

  Uses same argument options as **who** command option. **who** and **w** are variants of the same command and show slightly different data: **who** is more focused on from whom or where a connection originates, and **w** is more focused on what a connection is doing. The **w** command option also shows summary data about the target application, such as the number of connections, uptime, and so on.
- **kill** ‹*connection ID*›

  Disconnects the client connection that is identified by the connection ID. Mainly used to terminate clients that have failed to disconnect gracefully.
- **getActivity** ‹*connection ID*›

  Displays the most recent requests and responses from the client with the specified connection ID.

Examples:

```
dmctl -s <application name> who is 14
```

```
dmctl -s <application name> w is active
```

```
dmctl -s <application name> kill 14
```

```
dmctl -s <application name> getActivity 14
```

The last example command returns as many requests as the target application holds for a client that has a connection ID of 14. By default, that number is 25, which is determined by the RequestLogSize attribute of the SM_ElementManager. For each returned request, the command also returns the application's resulting response/error to the request.

Here is an excerpt of the return for a **dmctl getActivity** command:

```
dmctl> getActivity 2887
Activity record starts at October 18, 2012 12:10:34 AM EDT +654ms

ELAPSED RSPTIME DIR  MESSAGE
:00.000         ---> invokeOperation ICS_ConfigurationManager
   ICS-ConfigurationManager exportConfigFor { ConsoleOperation }
:00.008 :00.008 <--- OK
.
.
.
:00.076         ---> get InCharge_Feature OI CreationClassName
:00.076 :00.000 <-!- MR-E-OBJECT_NOT_FOUND-Object class::name
   'InCharge_Feature::OI' not found; in file
   "/work/tancurrent/DMT-9.2.0.0/42/smarts/repos/mr/dyn_acc.c" at line 1590
.
.
.
:01.105         ---> getf ICS_NotificationList ICS_NL-Default listName -1
:01.105 :00.000 <--- OK

Activity record ends at October 18, 2012 12:10:35 AM EDT +760ms
```

The first and last lines show the period of time that the activity log for the connection spans. In between are the log entries.

Each log entry has an elapsed time (which is shown relative to the “Activity record starts” time), an optional response time, a direction, and a message. The direction is a right arrow for a request to the application, a left arrow for a response, and a left arrow with an exclamation point (`<-!-`) for an error response.

The message is the request summary for a request, “OK” for a response, or error text for an error response. The response time for a request is empty, and the response time for a response shows the time difference between the time of the request and the response.

Viewing SM_ElementManager attribute values

You can view SM_ElementManager attribute values by entering the following dmctl command from the BASEDIR/smarts/bin directory of the target application:

```
dmctl -s <application_ name> get
   SM_ElementManager::SM-ElementManager::<attribute_name>
```

For example:

```
dmctl -s INCHARGE-MPLS-TOPOLOGY get
   SM_ElementManager::SM-ElementManager::FDLimit
```

To view all values in one invocation, enter the following dmctl command:

```
dmctl -s <application_ name> get SM_ElementManager::SM-ElementManager
   | more
```

Changing SM_ElementManager attribute values

EMC recommends that you do not change SM_ElementManager attribute values. However, if you want to change an attribute value, you can do so by entering the following dmctl command from the BASEDIR/smarts/bin directory of the target application:

```
dmctl -s <application_ _name> put
   SM_ElementManager::SM-ElementManager::<attribute name> <attribute
   value>
```

For example:

```
dmctl -s INCHARGE-MPLS-TOPOLOGY put
   SM_ElementManager::SM-ElementManager::RequestLogSize 30
```

The command-modified parameters are persistent, which means that your modifications are saved to disk when the application is shut down and are *not* reset when the application is restarted. Two exceptions are RecentHighestFD and FDLimit, which are initialized at application startup.

CHAPTER 9
Managing Log Files

This chapter consists of the following sections:

- Overview of message logging 134
- Retrieving and setting log, error and trace levels at runtime 138
- Log files 139
- Starting a new log file 142
- Controlling the number of saved log files 144

Overview of message logging

Figure 4 on page 134 presents an overview of message logging.

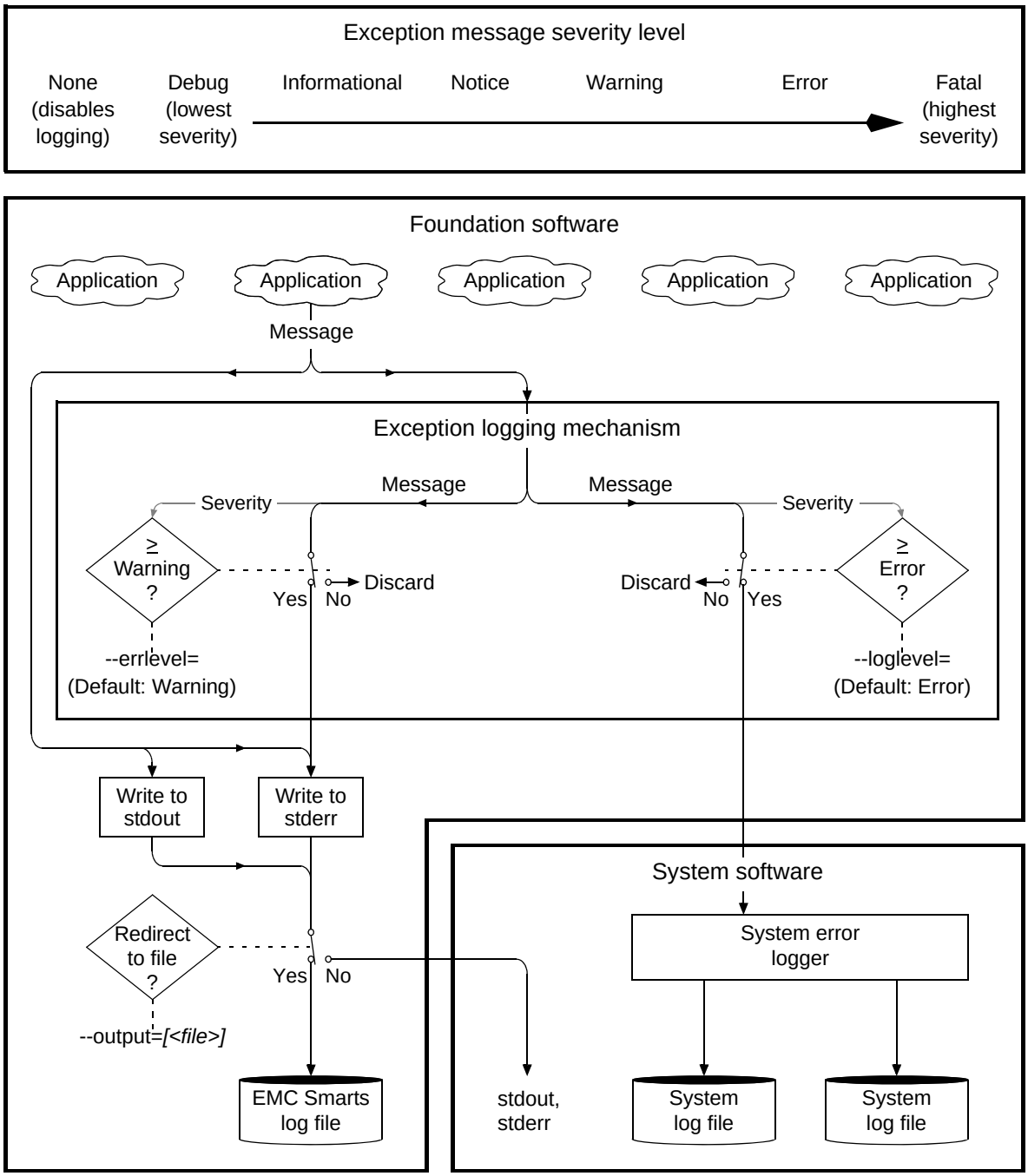

Figure 4 Message logging

All EMC Smarts Foundation processes (applications) such as Domain Managers, Global Managers, Adapter Platforms, and Adapters create log files that contain status information and error reports. Some processes use the exception logging mechanism that is shown in Figure 4, while others write directly to stdout or stderr.

When a process is started with the --output option, the status information and error reports are written to at least one log file. One log file is always created, but two different versions of the log file may be created, depending on the settings of the SM_LOCALE and SM_ENCODING_OUPUT environment variables.

The name of a log file has the following syntax:

*‹name›*_*‹locale›*_*‹encoding›*`.log`

where:

- *‹name›* in the log filename column is determined by the rules that apply to the --output command option, as described in "Command-line options that control messages" on page 137.
- *‹locale›* is based on the value of the SM_LOCALE environment variable.
- *‹encoding›* is based on the value of the SM_ENCODING_INPUT environment variable. "SM_LOCALE" on page 161 and "SM_ENCODING_*" on page 161 provide additional information.

A log file whose locale is en_US (default value of SM_LOCALE) and encoding is UTF-8 (default value of SM_ENCODING_INPUT) is always created when the --output option is provided.

- If the SM_LOCALE environment variable specifies the English locale (en_US), and the SM_ENCODING_OUTPUT environment variable specifies UTF-8, then the English log is the only log file produced.
- If the SM_LOCALE variable specifies a locale other than en_US, or the SM_ENCODING_OUTPUT environment variable specifies something other than UTF-8, a second log file is created. Log messages written to this second log are localized according to the setting of the SM_LOCALE variable, if such a localization exists.
- If SM_LOCALE specifies an invalid locale (one for which EMC Smarts software has not been localized), English log data will result in the process log file.

Table 25 on page 136 summarizes the filename, locale and encoding of the log files that are produced, including the system log containers.

Table 25 Log filenames, locales and encodings

Log use	Log filename (if –output is used)	Locale	Encoding
English log	*‹name›*_en_US_UTF-8.log For example, MYDM_en_US_UTF-8.log	en_US	UTF-8
Non-english log	*‹name›*_*‹locale›*_*‹encoding›*.log For example, MYDM_ja_JP_‹encoding›.log	Defined by SM_LOCALE	Specified by the SM_OUTPUT_ENCODING variable.
UNIX syslog		Defined by SM_LOCALE	UTF-8 (permitted and expected by the new IETF syslog-protocol RFC, still in a draft form)
Win Event Log		Defined by SM_LOCALE	UTF-16

Table 26 on page 136 provides several examples of SM_LOCALE and SM_ENCODING_OUTPUT variable settings and the resultant logs.

Table 26 SM_LOCALE and SM_OUTPUT_ENCODING examples

SM_LOCALE setting	SM_OUTPUT_ENCODING value	Logs that are produced
en_US	UTF-8	*‹name›*_en_US_UTF-8.log
en_US	ShiftJIS	*‹name›*_en_US_UTF-8.log *‹name›*_en_US_ShiftJIS.log
fr	UTF-8	*‹name›*_en_US_UTF-8.log *‹name›*_fr_UTF-8.log
zh	GB	*‹name›*_en_US_UTF-8.log *‹name›*_zh_GB.log

The volume of output to the system log containers is controlled by the --loglevel option. The volume of output to the log files is controlled by the `--errlevel` option, and by the environment variables that control filtering on specific error tables and error codes.

Log file management behavior applies to both log files. Log rolling is managed in a synchronized fashion so that data in multiple logs can be correlated. Thus, when the English log rolls, the non-English log, if it is being produced, will also roll.

Command-line options that control messages

EMC 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. Figure 4 on page 134 displays the range of security levels.

- Option --output[=*‹file›*]

  Write all logging output for an EMC Smarts application to a log file in the BASEDIR/smarts/local/logs directory. The name of the log file is *‹file›*_*‹locale›*_*‹encoding›*.log, or the --logname value if *‹file›* is omitted. If --output is not specified in the invocation line, then no logs are produced.

- Option --errlevel=*‹min_severity_level›*

  Used by the exception logging mechanism to determine the minimum exception level for writing messages to the log files. By default, --errlevel is set to warning. Exceptions below this level are not written to the log files.

- Option --loglevel=*‹min_severity_level›*

  Used by the 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.

- Environment variable SM_MAIN_OPTIONS

  Used to set the defaults for the standard options for all 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 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 an 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 log files from BASEDIR/smarts/local/logs to a user-specified location.

  Example

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

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

"The sm_service command line" on page 69 provides additional information about sm_service command options. Appendix A, "Environment Variables," provides additional information about environment variables.

Retrieving and setting log, error and trace levels at runtime

There are three computed attributes available to get and set the log, error, and trace levels of a Domain Manager at runtime. These computed attributes, described in Table 27 on page 138, are available on the SM_JIIM_Support object, and are therefore available from within JIIM code. The SM_System object is a subclass of SM_JIIM_Support and also inherits these computed attributes. The values of these computed attributes can be retrieved and set by using dmctl.

Table 27 Computed attributes to retrieve and set log, error, and trace levels at runtime

Computed Attributes	Description
logLevel	The minimum exception level for sending messages to the system error logger.The logLevel attribute is a string, and can be any one of the values set for the --loglevel command line option.
errLevel	The minimum exception level for writing messages to the log files. The errLevel attribute is a string, and can be any one of the values set for the --errlevel command line option.
traceLevel	Used to print a stack trace to the log file when an exception at this level or above occurs. Exceptions below this level do not write a stack trace. The traceLevel attribute is a string, and can be any one of the values set for the --tracelevel command line option.

Retrieving the current level

To retrieve the current levels, invoke the get method on SM_System::SM-System::logLevel, SM_System::SM-System::errLevel, or SM_System::SM-System::traceLevel. Invoking the get method will return a string representing the current level such as Warning, Error, or Fatal.

In this example the error level setting in the MYDM Domain Manager is retrieved, by using dmctl:

dmctl -s MYDM get SM_System::SM-System::errLevel

```
Warning
```

Setting the level

To change the current levels, invoke the put method on SM_System::SM-System::logLevel, SM_System::SM-System::errLevel, or SM_System::SM-System::traceLevel, and the level is changed appropriately.

In this example, the trace level setting is changed to None in the MYDM Domain Manager.

```
dmctl -s MYDM put SM_System::SM-System::traceLevel None
```

When you change the log, error or trace levels by using the put method on one of these computed attributes, dmctl does not return anything. However, a message is printed in the log file. The log message will appear similar to the following:

```
[April 8, 2009 5:03:41 PM EDT +122ms] t@1149000000 SM_ProtocolEngine-6
   JM_MSG-*-JM_TRACE_LEVEL_CHANGED-User 'user1', using remote dmctl
   client (id 6), on host host1 with credentials tpadmin1 has changed
   the Trace level to None; in file
   "/mypath/repos/jiim/SM_JIIM_Support_Impl.c" at line 458
```

Log files

When the --output option is specified for the sm_server sm_service command, each Manager stores its working history in a log file. Every time a Manager starts, it writes information to the log file when one of the following occurs:

- Error
- Connection is lost

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, as described in "SM_WRITEABLE" on page 170.

Also, the location where log files are saved can be separately controlled with the SM_LOGFILES environment variable, as described in "SM_LOGFILES" on page 170.

All log files have a .log file type. The default name of a log file is based on the Manager's name. For example, if the name of the Manager is MYDM, then the name of the English log file is MYDM_en_US_UTF-8.log. To change the default name, specify a filename with the --output option for the sm_server sm_service command.

The names of log files written by adapters are described in their respective user's guides.

EMC Smarts programs can maintain up to 999 different copies of backup log files for each log that is being generated. For example, if you are generating two log files, MYDM_en_US_UTF-8.log and MYDM_fr_UTF-8.log file, you can maintain up to 999 backup log files for the MYDM_en_US_UTF-8.log and 999 backup log files for the MYDM_fr_UTF-8.log.

The number of saved log files is determined by the value of the SM_BACKUP_FILE_LIMIT environment variable. When a Manager starts up, it renames a file that matches its log filenames and adds a .bak suffix to the name (for example, MYDM_en_US_UTF-8.log to MYDM_en_US_UTF-8.log.bak). If a file with this name already exists, it is renamed MYDM_en_US_UTF-8.log.NNN, and a new MYDM_en_US_UTF-8.log.bak is created. By default, programs save three log files, which does not include the active log file.

Log file management behavior applies to both log files. Log rolling is managed in a synchronized fashion so that data in multiple logs can be correlated. Thus, when the non-English log rolls, the English log, if it is being produced, will also roll.

Diagnostic header

Every log file (initial and after any log rolling) starts with a diagnostic header which indicates the startup and current information for a process.

The diagnostic header is as follows:

```
MAIN_MSG-*-STARTUP-The following lines are routine diagnostic output; this is not a crash.
==============================================================================
Start of process debug and runtime information:
==============================================================================
```

‹the standard header included with a stacktrace›

```
==============================================================================
End of process debug and runtime information
==============================================================================
```

The standard header for a stacktrace includes:

- Command line arguments of the process
- Which libraries are loaded
- The environment

Example

The diagnostic header will appear similar to the following example:

```
[July 15, 2009 10:58:41 AM MST +106ms] t@6084 platform
MAIN_MSG-*-STARTUP-The following lines are routine diagnostic output; this is not a crash.
==============================================================================
Start of process debug and runtime information:
==============================================================================
WinNT 200120:

C:/CMP/DMT90/bin/system/sm_server.exe -n TEST_OUT_M -c icf --broker=localhost:426 --output
DMT: V9.0.0.0(90743), 09-Jul-2009 12:01:53 Copyright 1995-2009, EMC Corporation - Build 1169
sm_clsapi.dll (90743) 09-Jul-2009 16:19:12 UTC

sm_rclt.dll (90744) 09-Jul-2009 16:52:37 UTC

sm_servif.dll (90744) 09-Jul-2009 16:52:45 UTC

sm_server.exe (90751) 09-Jul-2009 23:41:33 UTC

    time=Wed Jul 15 10:58:41 2009 (1247680721)
    pid=5986
    SM_HOME=C:/CMP/DMT90
    CWD=C:\CMP\DMT90\bin
Environment:
    ALLUSERSPROFILE=C:\Documents and Settings\All Users
    APPDATA=C:\Documents and Settings\user1\Application Data
    CLIENTNAME=Console
    CommonProgramFiles=C:\Program Files\Common Files
    COMPUTERNAME=COMP1
    ComSpec=C:\WINNT\system32\cmd.exe
    DEFLOGDIR=C:\Documents and Settings\All Users\Application Data\McAfee\DesktopProtection
    FP_NO_HOST_CHECK=NO
    HOMEDRIVE=H:
    HOMEPATH=\
    HOMESHARE=\\someDrive\user1$
    HummPATH=C:\Program Files\Hummingbird\Connectivity\10.00\Accessories\;
    JAVA_HOME=C:\jdk1.6.0_02
    LM_LICENSE_FILE=17955@smarts-lmgr92.smarts.com
    LOGONSERVER=\\CORP3
```

```
    MAVEN_HOME=C:\maven-2.0.7
    NUMBER_OF_PROCESSORS=2
    OS=Windows_NT

Path=C:/CMP/DMT90/local/bin/system;C:/CMP/DMT90/local/bin/system;C:/CMP/DMT90/bin/system;C:/C
MP/DMT90/local/lib;C:/CMP/DMT90/local/lib;C:/CMP/DMT90/lib;/opt/dev/foo/bin/system;/opt/dev/f
oo/lib;C:/jdk1.6.0_02/jre/bin;C:/jdk1.6.0_02/jre/bin/server;C:/maven-2.0.7/bin;C:/WINNT/syste
m32;C:/WINNT;C:/WINNT/System32/Wbem;C:/perl;C:/jdk1.6.0_02/bin;"C:/Program Files/Common
Files/EMC";C:/Program Files/Windows Imaging/
    PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
    PROCESSOR_ARCHITECTURE=x86
    PROCESSOR_IDENTIFIER=x86 Family 6 Model 23 Stepping 6, GenuineIntel
    PROCESSOR_LEVEL=6
    PROCESSOR_REVISION=1706
    ProgramFiles=C:\Program Files
    PROMPT=$P$G
    RoxioCentral=C:\Program Files\Common Files\Roxio Shared\9.0\Roxio Central33\
    SESSIONNAME=Console
    SM_AUTHORITY=<NONE>
    SM_BROKER=localhost:426
    SM_BROKER_DEFAULT=localhost:426
    SM_DEF=/opt/dev/foo
    SM_DEFHOME=/opt/dev/foo
    SM_FIXED=C:/CMP/DMT90;/opt/dev/foo
    SM_HOME=C:/CMP/DMT90
    SM_JAVA=C:\jdk1.6.0_02\jre\bin;C:\jdk1.6.0_02\jre\bin\server;
    SM_JAVAHOME=C:\jdk1.6.0_02\jre
    SM_JDK_HOME=C:\jdk1.6.0_02
    SM_LICENSE=17955@smarts-lmgr92.smarts.com
    SM_MAX_MONITORING_THREADS=1
    SM_PIPEDIR=\\\\.\\pipe\\smarts\\

SM_POP_1=PATH=C:\maven-2.0.7\bin;C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;C:\Program
Files\Common Files\Roxio Shared\DLLShared\;C:\Program
Files\Hummingbird\Connectivity\10.00\Accessories\;;c:\Program Files\Microsoft SQL
Server\90\Tools\binn\;C:\Program Files\Perforce;C:\perl;C:\jdk1.6.0_02\bin;"C:\Program
Files\Common Files\EMC";C:\Program Files\Windows Imaging\
    SM_POP_2=LM_LICENSE_FILE=
    SM_POP_3=SM_LIBPATH=
    SM_POP_4=SM_BROKER=
    SM_POP_5=SM_BROKER_DEFAULT=
    SM_POP_6=SM_PROCESS_PATH=
    SM_PROCESS_PATH=C:/CMP/DMT90/bin/system/sm_server.exe
    SM_SITEMOD=C:/CMP/DMT90/local;C:/CMP/DMT90;/opt/dev/foo/local;/opt/dev/foo

SM_SYMBOL_PATH=C:/CMP/DMT90/local/symbols;C:/CMP/DMT90/symbols;/opt/dev/foo/local/symbols;/op
t/dev/foo/symbols;C:/WINNT/symbols/dll;C:/WINNT/system32
    SM_WRITEABLE=C:/CMP/DMT90/local
    SystemDrive=C:
    SystemRoot=C:\WINNT
    TCL_LIBRARY=/opt/dev/foo/lib/tcl
    TEMP=C:\DOCUME~1\user1\LOCALS~1\Temp
    TMP=C:\DOCUME~1\user1\LOCALS~1\Temp
    UATDATA=C:\WINNT\system32\CCM\UATData\D9F8C395-CAB8-491d-B8AC-179A1FE1BE77
    USERDNSDOMAIN=someDomain.EMC.COM
    USERDOMAIN=someDomain
    USERNAME=user1
    USERPROFILE=C:\Documents and Settings\user1
    VS80COMNTOOLS=C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\
    VSEDEFLOGDIR=C:\Documents and Settings\All Users\Application Data\McAfee\DesktopProtection
    windir=C:\WINNT
==============================================================================
End of process debug and runtime information
==============================================================================
```

Date and timestamp, and thread id in each log message

The date and timestamp is written at the beginning of each log record. The thread id is also included in each log record, following the date and timestamp. For example:

```
[March 2, 2012 3:31:18 PM EDT +357ms] t@3012557744 SM_ProtocolEngine-1
SVIF_MSG-*-DSM-SM_DebugServer: purgeObserver()
```

Message at the end of a rolled log

When rolling a log, a message is printed at the bottom of the log file before it is rolled. The message is similar to the following example:

```
MAIN-N-Closing this log file at March 19, 2009 9:48:19 PM EDT;
   continuing in

    <BASEDIR>/local/logs/MYDM_en_US_UTF-8.log
```

Starting a new log file

You can request that a 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 or roll_log commands.
- For 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, copy or at least rename the .log.bak files as soon as they are created.

Log rolling is managed in a synchronized fashion so that data in multiple logs can be correlated. Thus, when the English log rolls, the non-English log, if it is being produced, will also roll.

The kill command method for UNIX systems

On UNIX systems only, it is possible to request that a 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 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.

NOTICE

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 Manager or adapter with administrative privileges. The syntax of the command is:

```
exec 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.

On UNIX, for example, to start a new log file for a Manager named MYDM, enter:

```
BASEDIR/smarts/bin/dmctl -s MYDM exec roll_log
```

On Windows, the equivalent command is:

```
BASEDIR\smarts\bin\dmctl -s MYDM exec roll_log
```

When roll_log is invoked, the Manager or adapter does the following:

1. Writes an informational message to the end of the current log files.
2. Repeats the steps it executed at startup.
3. Creates a new log file and a backup log file for each log file that was currently running.

All subsequent logging information is recorded to the new files.

Controlling the number of saved log files

EMC Smarts software can retain up to 999 rolled over log files for each log that is being generated. For example, if you are generating two log files, MYDM_en_US_UTF-8.log and MYDM_fr_UTF-8.log file, you can maintain up to 999 backup log files for the MYDM_en_US_UTF-8.log and 999 backup log files for the MYDM_fr_UTF-8.log.

The number of log files retained is determined by the value of SM_BACKUP_FILE_LIMIT. When a Manager starts up, it renames a file that matches its log filename, 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 5, the process is as follows:

1. A log file is created and it is named MYDM_en_US_UTF-8.log.
2. The MYDM_en_US_UTF-8.log file created in step 1 is rolled over:
   - The MYDM_en_US_UTF-8.log file created in step 1 is renamed to MYDM_en_US_UTF-8.log.bak.
   - A new MYDM_en_US_UTF-8.log file is created.
3. The MYDM_en_US_UTF-8.log file created in step 2 is rolled over:
   - The MYDM_en_US_UTF-8.log.bak file renamed in step 2 is now renamed to MYDM_en_US_UTF-8.log.001.
   - The MYDM_en_US_UTF-8.log file created in step 2 is renamed to MYDM_en_US_UTF-8.log.bak.
   - A new MYDM_en_US_UTF-8.log file is created.
4. The MYDM_en_US_UTF-8.log file created in step 3 is rolled over:
   - The MYDM_en_US_UTF-8.log.bak file renamed in step 3 is renamed to MYDM_en_US_UTF-8.log.002.
   - The MYDM_en_US_UTF-8.log file created in step 3 is renamed to MYDM_en_US_UTF-8.log.bak.
   - A new MYDM_en_US_UTF-8.log file is created.
5. The MYDM_en_US_UTF-8.log file created in step 4 is rolled over:
   - The MYDM_en_US_UTF-8.log.bak file renamed in step 3 is renamed to MYDM_en_US_UTF-8.log.003.
   - The MYDM_en_US_UTF-8.log file created in step 4 is renamed to MYDM_en_US_UTF-8.log.bak.
   - A new MYDM_en_US_UTF-8.log file is created.
6. Because the limit of MYDM_en_US_UTF-8.log.NNN files that are retained is set to 3, when the MYDM_en_US_UTF-8.log file created in step 5 is rolled over:
   - The oldest log file, MYDM_en_US_UTF-8.log.001 from step 3 , is deleted.
   - The MYDM_en_US_UTF-8.log.bak file renamed in step 5 is renamed MYDM_en_US_UTF-8.log.004.
   - The MYDM_en_US_UTF-8.log file created in step 5 is renamed MYDM_en_US_UTF-8.log.bak.
   - A new MYDM_en_US_UTF-8.log is created.

APPENDIX A
Environment Variables

This appendix describes the environment variables that are used by EMC Smarts Foundation software. It consists of the following sections:

- How variable values are interpreted 146
- Methods for setting environment variables..... 147
- Environment variables for users 149

How variable values are interpreted

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

How integer variables are interpreted

When the value of an environment variable is to be expressed as an integer, the 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 28 on page 146 provides examples of numeric values and how they are interpreted by the software.

Table 28 Examples of numeric values

Numeric value	Meaning
"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 “0X”. 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, the 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

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 an EMC Smarts product and the other sets the environment variable for a particular program.

Setting an environment variable product-wide

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

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

Here is an example procedure for adding the SM_AUTHORITY environment variable to the runcmd_env.sh file:

1. Go to the BASEDIR/smarts/bin directory and enter the following command to open the runcmd_env.sh file:

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

2. Use the following syntax to add the **SM_AUTHORITY** variable and its value:

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

3. Save and close the file.

   The modified version of the runcmd_env.sh file is saved to the BASEDIR/smarts/local/conf directory.

   At this point, any EMC Smarts program that is started from this installation directory will use the applicable environment variables that are specified in the runcmd_env.sh file.

4. EMC Smarts programs that were previously launched from this installation directory need to be restarted for any new environment variable to take effect. Chapter 5, “Controlling Startup,” describes how to start services.

Setting an environment variable for a program

You can also set an environment variable so that it only affects a particular program. Typically, this is done by adding the environment variable to the program's sm_service command line. For example, if you want a 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. You must have administrative privileges when typing this command.

   ```
   BASEDIR/smarts/bin/sm_service show --cmdline ic-am-server
   ```

   Output:

   ```
   BASEDIR/smarts/bin/sm_service install
      --name=ic-am-server
      --description="EMC Smarts IP Availability Manager"
      --startmode=runonce
      BASEDIR/smarts/bin/sm_server
      --name=INCHARGE-AM
      --config=icf
      --bootstrap=bootstrap-am.conf
      --port=0
      --subscribe=default
      --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.

   ```
   BASEDIR/smarts/bin/sm_service install
      --force
      --name=ic-am-server
      --description="EMC Smarts IP Availability Manager"
      --startmode=runonce
      --env=SM_SERVERCONNECT=BASEDIR/smarts/conf/serverConnect_IP.conf
      BASEDIR/smarts/bin/sm_server
      --name=INCHARGE-AM
      --config=icf
      --bootstrap=bootstrap-am.conf
      --port=0
      --subscribe=default
      --ignore-restore-errors
      --output
   ```

3. Do one of the following:

- Start the program.
- Stop and restart the program.

Environment variables for users

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

- “SM_DISPLAY”
- “SM_EDITOR”
- “SM_SNMP_BUG_COMPATIBLE”
- “SM_ENABLE_ESM_EVENT_MAPPING”
- “SM_ENABLE_SNMP_SET”
- “SM_MAIN_OPTIONS”
- “SM_OUTGOING_PROTOCOL”
- “SM_INCOMING_PROTOCOL”
- “SM_FIPS140”
- “SM_DISABLE_KEEPALIVES”
- “SM_IP_VERSIONS”
- “SM_MAX_MONITORING_THREADS”
- “SM_AUDIT_CLIENT_ACTIONS”
- “SM_AUDIT_ALL_CONNECTIONS”
- “SM_NO_LINE_FOLD”
- “SM_FOLD_WIDTH”
- “SM_HOSTSFILE”
- “SM_MAX_SESSION_VERSION”
- “SM_MIN_PKTSIZE_COMPRESS”
- “SM_SNMP_ALLOW_DUP_ENGINEID”
- “SM_SNMP_MAX_MESSAGE_SIZE”
- “SM_ASL_MAX_STACK_DEPTH”
- “SM_REGISTER_ADDR”
- “MALLOC_ARENA_MAX_”
- “SM_LOCALE”
- “SM_ENCODING_*”
- “SM_JAVA_PACKAGE_ROOT”
- “SM_JDK_HOME”
- “SM_JAVA”
- “SM_JAVAHOME”

- “SM_BROKER_DEFAULT”
- “SM_BROKER”
- “SM_BROKER_NTHREADS”
- “LM_LICENSE_FILE”
- “SM_LICENSE”
- “SM_LMGRD_LICENSE_FILE”
- “SM_LMGRD92_LICENSE_FILE”
- “SM_AUTHORITY”
- “SM_KEYFILE”
- “SM_CLIENTCONNECT”
- “SM_OKLOGIN_LOGLEVEL”
- “SM_SERVERCONNECT”
- “SM_DATETIME_FORMAT”
- “TZ”
- “LC_TIME”
- “SM_LOG_ROLL_SIZE”
- “SM_BACKUP_FILE_LIMIT”
- “SM_LOG_NAME_CUSTOM”
- “SM_LOGFILES”
- “SM_WRITEABLE”
- “SM_RULESET_PATH”
- “SM_COUNT_REPOS_LOCKS”
- “SM_REPOS_LOCKS_LIMIT”
- “SM_SUBSCRIPTION_TRACE”
- “SM_IGNORE_DYNMODEL_ERRORS”

SM_SITEMOD

The SM_SITEMOD variable provides a directory search list that Smarts software uses to locate files. This list is used to find files that users can, and in some cases must, modify to use Smarts at their site. Such files include configuration files, ASL files, and scripts. For locating a file, the directories listed in SM_SITEMOD are searched in the same order as they are specified. The components of the list are separated by colons (:) on UNIX and semicolons (;) on Windows.

The default value of SM_SITEMOD is ‹smarts›/local:‹smarts›. For specifying a new search directory, one only has to set SM_SITEMOD to the new directory. The "runcmd" wrapper program that is used to launch all Smarts executables, internally constructs the actual search path, by appending the known search paths to the user specified path.

It is strongly recommended that SM_SITEMOD be only specified in the service installation command line. Setting SM_SITEMOD in runcmd_env.sh is not recommended, as it sometimes causes the internally constructed search path to get overridden, which may cause certain components to not work correctly.

The Console does not currently support SM_SITEMOD.

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 following editors:

- Edit editor on UNIX systems
- WordPad editor on Windows systems

SM_SNMP_BUG_COMPATIBLE

Implementation of the SNMP protocol is strict in that the software only sends or receives SNMP messages that conform to the SNMP standard. By default, the software only accepts conforming SNMP messages. Unfortunately, not all SNMP implementations are as strict. The result is that EMC Smarts programs may receive nonconforming SNMP messages.

In some cases, the software can successfully interpret and handle nonconforming messages. If you set the SM_SNMP_BUG_COMPATIBLE environment variable, the software will accept the nonconforming SNMP messages in two 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.

When the SM_SNMP_BUG_COMPATIBLE environment variable is set, any invalid SNMP message that is accepted by the software is logged at a warning level to pinpoint the source of the invalid SNMP message and to determine the reason why it is invalid.

SM_SNMP_BUG_COMPATIBLE is a Boolean.

SM_ENABLE_ESM_EVENT_MAPPING

This environment variable is set to TRUE by default in the runcmd_env.sh file. It is applicable only to Service Assurance Manager and Adapter Platform (OI Server) installations.

This variable ensures that the Audit History is preserved post upgrade for active VMwareESX and vNIC notifications. In case some ESX or Virtual Interface instances remain in the alert state before and post upgrade, Server Manager will notify them as alerts for Hypervisor and VirtualInterface instances. Service Assurance Manager will map them to VMwareESX and vNIC if they were active before the upgrade. After the problem clears, the notifications will be archived. This allows you to keep a complete history of the problem and the user actions performed on the notification. Post archival, new outages are displayed under the new class name.

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_MAIN_OPTIONS

The environment variable SM_MAIN_OPTIONS enables you to set default values for the standard options supported by the software, as described in Table 29 on page 153. You can override the values specified by SM_MAIN_OPTIONS by providing alternative values on the command line.

For example, if you set SM_MAIN_OPTIONS to --errlevel=fatal, programs will record all errors with a level of fatal or higher.

Table 29 Standard options supported by SM_MAIN_OPTIONS (page 1 of 2)

Standard option	Definition
--accept=*‹hosts›*	Specify a list of hosts, by name or IP address, from which programs accept connections. Multiple entries should be separated by commas. By default, programs accept connections from any host. Programs that accept connections are dmstart, sm_server, sm_adapter, and sm_trapd. Note that sm_adapter and sm_trapd accept only connections when started with the --name option. To limit connections to the host on which the software is installed, specify the name or IP address of the host instead of localhost.
--daemon	Run the program as a daemon (UNIX only). Messages generated from internal logging calls are directed to the log file. Log files are written to the BASEDIR/smarts/local/logs directory.
--facility=*‹facility›*	Instructs the program to log any syslog messages with the specified syslog facility (UNIX/Linux only). The default is Daemon. The *‹facility›* setting can be one of the following: • Kern • User • Mail • Daemon • Auth • Lpr • News • Uucp • Cron • Local0 • Local1 • Local2 • Local3 • Local4 • Local5 • Local6 • Local7
--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 29 Standard options supported by SM_MAIN_OPTIONS (page 2 of 2)

Standard option	Definition
--logname=*‹name›*	Specify the name used to identify the 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›*]	Messages generated from internal logging calls are directed to a log file. Log files are written to the BASEDIR/smarts/local/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 program should use as the source address for outgoing ICMP and SNMP packets and the destination address for which it accepts incoming ICMP and SNMP packets. *‹IP Address›* must be a *v4* address.
--version	Print the version of the program.

SM_OUTGOING_PROTOCOL

This environment variable controls the kinds of connections that a program that is acting as a client is allowed to initiate, when not using FIPS 140 mode. 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.

SM_INCOMING_PROTOCOL

This environment variable controls the kinds of connections that a program that is acting as a server is allowed to accept, when not using FIPS 140 mode. 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.

SM_FIPS140

This environment variable controls enables and disables FIPS mode:

- TRUE = Enable FIPS 140 mode
- FALSE = Disable FIPS 140 mode (default)

When FIPS 140 mode is enabled, the software uses Transport Layer Security v1.2, and the settings of SM_OUTGOING_PROTOCOL and SM_INCOMING_PROTOCOL are ignored.

SM_DISABLE_KEEPALIVES

This environment variable controls whether KeepAlive messages for the EMC Smarts proprietary inter-process communication (IPC) protocol are used. To disable KeepAlive messages, specify a nonzero number. To enable KeepAlive messages, specify a zero (0). KeepAlive messages are enabled by default.

The EMC Smarts proprietary IPC uses KeepAlive messages to test TCP connections between EMC Smarts clients and Managers. In order for a Manager to detect whether a client is temporarily unresponsive or has permanently dropped the TCP connection, the client sends a KeepAlive message as an out-of-band byte of TCP urgent data every 90 seconds to indicate that the client is still connected. If the Manager does not receive periodic KeepAlive messages, it concludes that the client has dropped the connection and shuts down the server end of the connection.

NOTICE

Bear in mind that in some cases, not receiving KeepAlive messages from the client could be due to other software applications and hardware *not* supporting TCP urgent data, not the result of the connection being down.

Under certain circumstances, the client's operating system might incorrectly insert the urgent data into the regular data stream, which corrupts the regular data stream and causes problems in the connection. If the urgent data byte leaks in at the point where the next command is expected, the leak can result in an invalid handler code and generate additional messages in the Manager's log file.

In rare cases when communication issues exist, you may disable KeepAlives by specifying a nonzero number for the SM_DISABLE_KEEPALIVES environment variable. Because this workaround might cause Managers (whose clients have disconnected) to experience memory growth and run out of memory, work with your EMC customer support team to optimize your network operation environment if the problem persists.

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 Guide*.

NOTICE

The SM_DISABLE_KEEPALIVES environment variable has no effect on version 9.2 and later connections that use Transport Layer Security.

SM_IP_VERSIONS

The SM_IP_VERSIONS environment variable enables you to control the address family used for hostname resolution. It affects the following:

- EMC Smarts utilities that use a command line (for example, sm_snmpwalk, dmctl, sm_tpmgr)
- Name resolution during discovery
- ASL scripts
- Ping
- DNS lookup

SM_IP_VERSIONS also controls which ports a Domain Manager listens on:

- If SM_IP_VERSIONS is set to "v4" or "v6", a Domain Manager will only open sockets and listen on that address family.
- If SM_IP_VERSIONS is set to "v4v6" or "v6v4", a Domain Manager will listen on both address families.

The SM_IP_VERSIONS setting is also dependent on the address family as follows:

- If you are running on a system which only supports a single address family, it is recommended that SM_IP_VERSIONS be set to "v4" or "v6", as appropriate.
- If you are running on a system which supports both address families, set it to the value which best describes your administrative preferences.

The variable can be set depending on the order in which you want to do hostname resolution.If the variable is not set, and the address family is not explicitly provided on the command line (for example, frame.someDomain.emc.com:v4), the default behavior is to resolve the hostname as an IPv4 hostname. The variable should be set to the address family that is predominate for the network.

To set this variable, add it to the runcmd_env.sh file, which is located in the BASEDIR/smarts/local/conf directory of the product.

The syntax of the environment variable is:

```
SM_IP_VERSIONS="<ip_value>"
```

Table 30 on page 156 lists acceptable values for the SM_IP_VERSIONS environment variable.

Table 30 Acceptable values for the SM_IP_VERSIONS environment variable

ip_value	Description
"V4"	The hostname is resolved to an IPv4 hostname (default).
"V6"	The hostname is resolved to an IPv6 hostname.
"V4V6"	The hostname is resolved to an IPv4 hostname. If that fails, the Domain Name System server tries to resolve the hostname to an IPv6 hostname.
"V6V4"	The hostname is resolved to an IPv6 hostname. If that fails, the Domain Name System server tries to resolve the hostname to an IPv4 hostname.
Note: The acceptable value can also be lowercase ("v4", "v6", "v4v6" or "v6v4").	

Detailed instructions about setting environment variables and information about the runcmd_env.sh file is provided in the "Methods for setting environment variables" on page 147.

SM_MAX_MONITORING_THREADS

The SM_MAX_MONITORING_THREADS environment variable represents the maximum number of monitoring threads that the monitoring system will use for monitoring changes. The default value is 3.

The recommended values are described in Table 31 on page 157.

Table 31 Recommended values for SM_MAX_MONITORING_THREADS

Platform	Recommended value
Windows 2003 Windows 2008	1
All other supported platforms	Up to 3, but no more than the number of CPUs on the machine.

SM_AUDIT_CLIENT_ACTIONS

Setting the SM_AUDIT_CLIENT_ACTIONS environment variable equal to 1 produces the log if the client requests it. However, even if the client requests it, auditing does not occur if it has not been enabled on the server with either the SM_AUDIT_CLIENT_ACTIONS variable, or the SM_AUDIT_ALL_CONNECTIONS environment variable, which is described in "SM_AUDIT_ALL_CONNECTIONS" on page 157.

SM_AUDIT_ALL_CONNECTIONS

Setting the SM_AUDIT_ALL_CONNECTIONS environment variable equal to 1 forces the logging even if the client does not request it. If you set SM_AUDIT_ALL_CONNECTIONS equal to 1, then you do not have to set SM_AUDIT_CLIENT_ACTIONS, described in "SM_AUDIT_CLIENT_ACTIONS" on page 157.

If you set SM_AUDIT_ALL_CONNECTIONS, you do not need to modify the client. However, if you suspect a specific client you might still want to use the by-request-only access provided by SM_AUDIT_CLIENT_ACTIONS to reduce the amount of logging coming from other sources.

SM_NO_LINE_FOLD

The environment variable SM_NO_LINE_FOLD can be set to prevent the line wrapping in log files. Add the following lines to the BASEDIR/local/conf/runcmd_env.sh file:

```
export SM_NO_LINE_FOLD
SM_NO_LINE_FOLD="TRUE"
```

You can use any string value for SM_NO_LINE_FOLD, it is only if it has no string associated with it that lines fold.

SM_FOLD_WIDTH

The SM_FOLD_WIDTH environment variable removes the default line wrapping of exception messages in the log file. The lines will only wrap if SM_FOLD_WIDTH=X where X specifies the column at which to fold the lines.

SM_HOSTSFILE

On 32-bit Solaris builds of the software, a bug in the Solaris system libraries prevents /etc/hosts from being read properly if more than 256 file descriptors are in use. If you set SM_HOSTSFILE=1 in the environment, then /etc/hosts is read using application code, bypassing the system libraries as a work-around to this problem.

NOTICE

This work-around only handles IPv4 addresses in /etc/hosts. The work-around does not work for IPv6 addresses, and will not read /etc/ipnodes (or /etc/inet/ipnodes). IPv6 addresses obtained from DNS are handled properly in all cases. If it is necessary to obtain IPv6 addresses from /etc/ipnodes, you should use the 64-bit version of the product.

SM_HOSTSFILE is only available on Solaris versions of the software. It operates in both 32 and 64-bit builds, but since the 64-bit system libraries work properly, it is recommended that it *not* be set on 64 bit builds.

SM_MAX_SESSION_VERSION

SM_MAX_SESSION_VERSION is an environment variable that controls the maximum version of the protocol session that is negotiated by the peer. The valid levels are described inTable 32 on page 158,

Table 32 Valid levels for SM_MAX_SESSION_VERSION

Level	Description
0	Legacy. No KeepAlives. No encryption
1	Encryption. No KeepAlives.
2	Encryption and KeepAlives. This level is the default value.
3	Encryption and Inband KeepAlives. Also supports compression.

NOTICE

Encryption negotiation also uses SM_INCOMING_PROTOCOL and SM_OUTGOING_PROTOCOL to determine exactly the level that is negotiated.

If the previous behavior of out-of-band (OOB) KeepAlives is required, set this variable to 2.

SM_MIN_PKTSIZE_COMPRESS

The SM_MIN_PKTSIZE_COMPRESS environment variable enables compression for flow frames ›= ‹*size*›. By default, compression is disabled.

- If unset or equal to 0, SM_MIN_PKTSIZE_COMPRESS enables compression with a reasonable default size.
- To disable compression, set SM_MIN_PKTSIZE_COMPRESS to a large value (›65536).

SM_SNMP_ALLOW_DUP_ENGINEID

The SNMP V3 component will not interact with two devices with the same engine Id in the managed domain in accordance to RFC 2574 (User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)), namely section 2.2: 2.2. Replay Protection. Each SNMP engine maintains three objects: snmpEngineID, which (at least within an administrative domain) uniquely and unambiguously identifies an SNMP engine.

When the environment variable SM_SNMP_ALLOW_DUP_ENGINEID is set, the SNMP Lcd will index the specific engine data by ip:port instead of engineId thus allowing interaction with networks not conforming with section 2.2 of RFC2574. The recommended way of enabling this feature is to add SM_SNMP_ALLOW_DUP_ENGINEID=1 in the local/conf/runcmd_env.sh.

SM_SNMP_MAX_MESSAGE_SIZE

SM_SNMP_MAX_MESSAGE_SIZE is an environment variable that controls the maximum size of the SNMP datagram that is processed. The default value is 8k (8192 bytes).

The default size can be overridden by setting the environment variable SM_SNMP_MAX_MESSAGE_SIZE to the required maximum size. For example, setting SM_SNMP_MAX_MESSAGE_SIZE equal to 16384 sets the maximum to 16k, which is double the default value.

If you try to set SM_SNMP_MAX_MESSAGE_SIZE to less than 1472 (the recommended minimum for UDP/IPv4 per RFC-3417, section 3.2), it will be silently adjusted to be exactly 1472.

NOTICE

Although SM_SNMP_MAX_MESSAGE_SIZE can be used to set the maximum SNMP datagram size to less than the default 8k, this is *strongly* discouraged.

If you try to set SM_SNMP_MAX_MESSAGE_SIZE greater than 64k (65536 bytes), it will be set to exactly 65536. A value of 65536 bytes is the largest possible datagram size supported by UDP.

NOTICE

Setting SM_SNMP_MAX_MESSAGE_SIZE to large values will increase server memory usage. This is especially true for trap processing during periods of high trap reception rate. Therefore, this option should only be used when it is needed to communicate with SNMP devices sending unusually large datagrams. Only increase the setting of this environment variable on the specific Domain Managers which manage those devices.

SM_ASL_MAX_STACK_DEPTH

In order to avoid a recursive ASL script to crash the server, a limit of 700 calls has been placed on the recursion of ASL scripts. When this happens a message is logged similar to:

```
ASL-W-ERROR_PARSE-While parsing rule 'START'
 ASL-PARSER_STACK_OVERFLOW-ASL engine has exceeded the stack limit of
   700 during execution
 ASL-CALL_STACK_HEADING-Generating execution trace of RuleSet:
   /emc/ferren/DEV/p-dmt90rel/smarts/ga/linux_rhAS40-x86-64/debug/tasl
   stackoverflow.asl
  ASL-CALL_STACK_RULE-   RuleName: R1, Line: 19
  ASL-CALL_STACK_RULE-   RuleName: R1, Line: 19
  ASL-CALL_STACK_RULE-   RuleName: R1, Line: 19
```

If you have a valid (non-recursive) case where the stack depth of ASL needs to be over 700, the limit can be changed by setting the environment variable SM_ASL_MAX_STACK_DEPTH.

If the value is too high, process stack overflow can occur resulting in server crash.

SM_REGISTER_ADDR

When a server is running on a machine which has more than one IPv4 and/or more than one IPv6 address, it selects one address from each protocol family to register with the Broker. The selection of which address is arbitrary. The selection algorithm is not documented and subject to change.

On a fully-connected network, the arbitrary selection is usually satisfactory, but there may be situations where it is required to force a specific address to be registered. This can be accomplished by setting the environment variable SM_REGISTER_ADDR. The value can be an IPv4 address literal, an IPv6 address literal, or one of each, delimited by a comma.

If you specify an address for one protocol family, the other one will revert to using the original arbitrary selection algorithm.

There is no checking to verify that the specified address is configured or reachable. Specifying an incorrect value can result in a non-functional server. It is recommended that this option only be used in specific situations where the default behavior is unacceptable.

An IPv4 example:

```
SM_REGISTER_ADDR=10.5.24.105
```

An IPv6 example:

```
SM_REGISTER_ADDR=fe80::218:8bff:fe48:50b1
```

Both v4 and v6. The order is not important:

```
SM_REGISTER_ADDR=fe80::218:8bff:fe48:50b1,10.5.24.105
```

MALLOC_ARENA_MAX_

The MALLOC_ARENA_MAX_ environment variable controls the maximum number of arenas that can be created for a process using the ptmalloc library. An arena is a data structure used by ptmalloc for maintaining reuse of free memory.

When the ptmalloc library is loaded and initialized it determines whether the MALLOC_ARENA_MAX_ environment variable has been defined in the environment variable set:

- When the MALLOC_ARENA_MAX_ is not defined in the environment variable set, there is no maximum number of arenas. This is the default.
- If MALLOC_ARENA_MAX_ is set to any positive integer, then that number defines the number of arenas.

Variables that control the locale and encoding

These variables specify the local and encoding used by the programs.

SM_LOCALE

The SM_LOCALE environment variable specifies the appropriate language and region in which you want to see output. In general, SM_LOCALE must be specified consistently with advice in IETF RFC 1766/3066. If you do not specify a value for SM_LOCALE, it defaults to en_US.

ICU allows for some flexibility on the case (upper and lower) of the language and country codes and for the separator character used to join the two.

EMC Smarts software will convert the value specified in SM_LOCALE to a normalized form that can be recognized by all ICU library functions which take a locale argument. The normalization (done by calling the ICU function, uloc_canonicalize) enables ICU library functions to perform predictably.

```
export SM_LOCALE=ja_JP
sm_server --name=Tokyo
```

SM_ENCODING_*

There are four SM_ENCODING_* environmental variables that can be set.

- SM_ENCODING_SHELL indicates the encoding used by the shell, primarily useful for specifying what encoding is used for command line arguments.
- SM_ENCODING_SYSTEM indicates the encoding used for internal interaction with the operating system.
  - Primarily dictates the encoding used for file system access.
  - Secondarily indicates the encoding used when internally accessing the environment.
- SM_ENCODING_INPUT indicates the encoding used by stdin.
- SM_ENCODING_OUTPUT indicates the encoding used by stdout and stderr.

Tuning the SM_ENCODING_INPUT and SM_ENCODING_OUTPUT values is primarily useful if you are redirecting or piping text somewhere other than the shell.

The behaviors of these variables is different on Windows and UNIX.

On Windows the behavior is:

- If SM_ENCODING_SHELL is not set, then its value will default to the current Windows "Active Code Page."
- SM_ENCODING_SYSTEM will always be ignored since there is only one valid possible value, and it is set internally.
- If SM_ENCODING_INPUT is not set, then its value will default to the current Windows "OEM Code Page."
- If SM_ENCODING_OUTPUT is not set, then its value will mirror SM_ENCODING_INPUT.

NOTICE

Separate values are necessary due to the way the Windows command shell (cmd.exe) behaves.

On UNIX the behavior is:

- If SM_ENCODING_SHELL is not set, then its value will default to "UTF-8."
- If SM_ENCODING_SYSTEM is not set, then its value will mirror SM_ENCODING_SHELL.
- If SM_ENCODING_INPUT is not set, then its value will mirror SM_ENCODING_SHELL.
- If SM_ENCODING_OUTPUT is not set, then its value will mirror SM_ENCODING_INPUT.

NOTICE

In UNIX it generally makes sense to set all four values identically.

The following example sets the encoding to Shift_JIS.

```
export SM_ENCODING_INPUT=Shift_JIS
export SM_ENCODING_OUTPUT=Shift_JIS
export SM_ENCODING_SHELL=SHIFT_JIS
export SM_ENCODING_SYSTEM=Shift_JIS
sm_server --name=Osaka
```

Variables that control Java settings

These variables specify the Java settings.

SM_JAVA_PACKAGE_ROOT

The SM_JAVA_PACKAGE_ROOT environment variable can be used to specify the java package root.

SM_JDK_HOME

The SM_JDK_HOME environment variable is used to indicate the location of the JDK, if present. If you provide the location of the JDK while running the appropriate setup script, the SM_JDK_HOME variable will be defined in the runcmd_env.sh file.

NOTICE

If you are using a 64-bit version of the software, then you should set SM_JDK_HOME to the location of a 64-bit version of the JDK. Likewise, if you are using a 32-bit version of the software, then you should set SM_JDK_HOME to indicate the location of a 32-bit version of the JDK.

SM_JAVA

The SM_JAVA 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). On Windows, the value of SM_JAVA is added to the PATH.

This variable is optional. If left unset, runcmd_env.sh will assume standard paths relative to the value of SM_JAVAHOME.

SM_JAVAHOME

The SM_JAVAHOME environment variable is used in conjunction with SM_JAVA to specify a non-standard JRE. It specifies the location of the JRE, and defines the value of the JVM's "java.home" system property.

- If SM_JAVAHOME is left unset and SM_JDK_HOME was specified, then SM_JAVAHOME will effectively be set to SM_JDK_HOME/jre.
- If left unset and SM_JDK_HOME is also unset, and if the BASEDIR/jre directory exists, then SM_JAVAHOME will effectively be set to BASEDIR/jre.

Variables that control the Broker

These variables control the location of the Broker.

SM_BROKER_DEFAULT

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

The software follows these steps to determine the Broker's location:

1. Verifies whether 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.

NOTICE

The hostname smarts-broker is usually an alias, such as a DNS CNAME.

SM_BROKER

This environment variable 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.

SM_BROKER_NTHREADS

This environment variable determines the maximum number of threads that the Broker can use. Valid values are 20 through 128, with a default value of 20.

If a user sets SM_BROKER_NTHREADS to an invalid string or a value less than 20, the value is set to 20 (default). If a user sets the variable to a value greater than 128, the value is set to 128.

Variables that control FlexNet Publisher licensing

These variables control the FlexNet Publisher licensing.

LM_LICENSE_FILE

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

The user does not have to set this environment variable; instead, use SM_LICENSE. The "runcmd" wrapper program that is used to launch all Smarts executables internally sets LM_LICENSE_FILE based on SM_LICENSE_FILE.

SM_LICENSE

The value of the SM_LICENSE 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 Publisher variable is the vendor-specific environment variable that specifies the location of the license files or the license servers. When both LM_LICENSE_FILE and SM_LMGRD_LICENSE_FILE are set, the SM_LMGRD_LICENSE_FILE environment variable takes precedence.

The *FlexNet Publisher License Administration Guide* provides additional information about the LM_LICENSE_FILE and SM_LMGRD_LICENSE_FILE environment variables. The *FlexNet Publisher License Administration Guide* is automatically installed with the EMC Smarts documentation into the BASEDIR/smarts/doc/pdf directory. The name of the file is fnp_licAdmin.pdf.

SM_LMGRD92_LICENSE_FILE

Similar to LM_LICENSE_FILE, this FlexNet Publisher variable is the vendor-specific environment variable that specifies the location of the license files or the license servers. When both LM_LICENSE_FILE and SM_LMGRD_LICENSE_FILE are set, the SM_LMGRD_LICENSE_FILE environment variable takes precedence.

The *FlexNet Publisher License Administration Guide* provides additional information about the LM_LICENSE_FILE and SM_LMGRD_LICENSE_FILE environment variables. The *FlexNet Publisher License Administration Guide* is automatically installed with the EMC Smarts documentation into the BASEDIR/smarts/doc/pdf directory. The name of the file is fnp_licAdmin.pdf.

The user does not have to set this environment variable. The FlexNet licensing software automatically derives a value for this based on the LM_LICENSE_FILE variable.

Variables that control security

These variables specify the security used by EMC Smarts programs.

SM_AUTHORITY

This environment variable determines the selection of the authority programs used by the software for different parts of the authentication process. For example:

```
SM_AUTHORITY=IDENTIFY=<program1>,AUTHENTICATE=<program2>
```

where:

- *‹program1›* is the program granting credentials
- *‹program2›* is the program validating credentials.

  NOTICE

  Program1 and program2 may refer to the same program.

- SM_AUTHORITY=‹STD› means:

  ```
  SM_AUTHORITY=IDENTIFY=sm_authority,AUTHENTICATE=sm_authority
  ```

- SM_AUTHORITY=‹NONE› means:

  ```
  SM_AUTHORITY=IDENTIFY=sm_authnone,AUTHENTICATE=sm_authnone
  ```

- SM_AUTHORITY=someAuthority means:

  ```
  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 filename. 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 ERROR severity, rather than NOTICE.

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 filename. The default location is the BASEDIR/smarts/local/conf/serverConnect.conf directory.

Variables that control date and time formatting

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 consoles are formatted by the consoles themselves and are not controlled by these variables.

Additional information can be found on many systems, such as:

- man pages on UNIX systems
- Microsoft Developer Network (MSDN) on Windows systems

If these are not available on your system, various sites on the Web publish this information.

SM_DATETIME_FORMAT

The SM_DATETIME_FORMAT environment variable determines the format of timestamps in server log files. The SM_DATETIME_FORMAT environment variable can be set to several different values. The default is SI_DAT_LONG.

The following examples show the output, based on the setting of SM_DATETIME_FORMAT:

```
SM_DATETIME_FORMAT=SI_DAT_LOCALE_NEUTRAL
  1969/12/31 20:00:42

SM_DATETIME_FORMAT=SI_DAT_SHORT
  31/12/69 20:00

SM_DATETIME_FORMAT=SI_DAT_MEDIUM
  31 déc. 1969 20:00:42

SM_DATETIME_FORMAT=SI_DAT_LONG
  31 décembre 1969 20:00:42 HMG-05:00

SM_DATETIME_FORMAT=SI_DAT_FULL
  mercredi 31 décembre 1969 20:00:42 États-Unis (New York)

SM_DATETIME_FORMAT=SI_DAT_DEFAULT
  31 décembre 1969 20:00:42 HMG-05:00

SM_DATETIME_FORMAT=
  31 décembre 1969 20:00:42 HMG-05:00

SM_DATETIME_FORMAT=SI_DAT_UTC (similar to SI_DAT_LOCALE_NEUTRAL but in
   UTC regardless of the setting of TZ)
  1970/01/01 01:00:42

SM_DATETIME_FORMAT=SI_DAT_ISO8601
  1970-01-01T01:00:42.000Z
```

EMC has created a new value, SI_DAT_CUSTOM, which enables the Foundation software to recognize the old SM_FORMAT_* environment variables (SM_FORMAT_DATE, SM_FORMAT_TIME, and SM_FORMAT_DATETIME). The SM_FORMAT_* environment variables were deprecated in Foundation 9.0 because their date-time formats are not internationalization (I18N) capable.

Dropping support for the SM_FORMAT_* variables affected custom parsing scripts that were based on the date-time format of those variables. To enable users to continue using their custom parsing scripts, EMC created SI_DAT_CUSTOM.

Because the date-time formats for the SM_FORMAT_* variables are only meaningful under English locales, EMC recommends that you use the other date-time formats that are available through the SM_DATETIME_FORMAT variable.

If SM_DATETIME_FORMAT is not set to SI_DAT_CUSTOM, the old SM_FORMAT_* variables will not be considered by the Foundation software.

TZ

This system environment variable determines which time zone is used when formatting a time. As such, it controls the conversion from an internal time base 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, 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.

Variables that control reading and writing files

These variables control the location of output, such as log files and repository. They also control the location from which EMC Smarts reads files, such as ASL scripts.

SM_LOG_ROLL_SIZE

This environment variable determines the size to which a server log file may grow before the server creates a new log file. The size can be any value between 5 megabytes (5,000,000 bytes) and 2 gigabytes (2,000,000,000 bytes), with a default value of 200 megabytes.

A user appends a suffix letter, *k* for kilobytes, *M* for megabytes, or *G* for gigabytes, to a user-specified integer value, to indicate that the integer value is to be multiplied by 1024, 1048576, or 1073741824, respectively. As examples, 200M indicates a value of 200 megabytes, 1G indicates a value of 1 gigabyte, and 2G indicates a value of 2 gigabytes.

If a user sets SM_LOG_ROLL_SIZE to a value less than 5 megabytes, the value is set to 5 megabytes. If a user sets SM_LOG_ROLL_SIZE to a value greater than 2 gigabytes, the value is set to 2 gigabytes.

The number of retained log files is determined by the value of SM_BACKUP_FILE_LIMIT. If you set SM_LOG_ROLL_SIZE to a value such that the log rolls frequently, you should also set SM_BACKUP_FILE_LIMIT to a higher value than the default value of 3. You should keep at least seven days worth of log files.

NOTICE

Log rolling by cron jobs is still supported.

SM_BACKUP_FILE_LIMIT

This environment variable determines the number of log files that are saved by the server. A server creates a new log file when the server is restarted or when the log file is rolled over. "Managing Log Files" on page 133 provides additional information, including an example.

Valid values are 0 through 999, with a default value of 3. If SM_BACKUP_FILE_LIMIT is set to 0, only the current log file and a backup of the previous log file are saved. If a user sets the variable to an invalid string or a negative value, the value is set to 3(default). If a user sets the variable to a value greater than 999, the value is set to 999.

The value for SM_BACKUP_FILE_LIMIT applies to *each* version of a log file that a server is configured to generate. For log files MYDM_en_US_UTF-8.log and MYDM_fr_UTF-8.log, for example, the EMC Smarts software can retain up to 999 rolled over log files for each of these log files.

SM_LOG_NAME_CUSTOM

This environment variable SM_LOG_NAME_CUSTOM enables you to specify the earlier log naming convention for your locale specific log, that is, use pre-Foundation V9.X log naming. The ‹locale›_‹encoding› string is not appended to the base log name.

When SM_LOG_NAME_CUSTOM is set to TRUE, ‹locale›_‹encoding› will not be attached to the non-English logs, or, if there is only one log (English log), ‹locale›_‹encoding› will not be attached to that log. The ‹locale›_‹encoding› will be attached to the English log if there are two logs (local and English).

Examples

If the server is started with option --output=name with SM_LOCALE=en_US and SM_ENCODING_OUTPUT=UTF-8, only name.log is created in the BASEDIR/local/logs directory.

If the server is started with option --output=name with SM_LOCALE=fr_FR and SM_ENCODING_OUTPUT=UTF-8, name.log and name_en_US_UTF-8.log files are created in the BASEDIR/local/logs directory.

If the server is started with option --output=name with SM_LOCALE=en_US and SM_ENCODING_OUTPUT=ISO-8859-15, name.log and name_en_US_UTF-8.log files are created in the BASEDIR/local/logs directory.

NOTICE

If you use --output without a ‹name›, the output uses the server name to create the log file.

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_WRITEABLE

This environment variable specifies the location where the 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 33 on page 170 lists the directories whose location is controlled by SM_WRITEABLE:

Table 33 Directories controlled by SM_WRITEABLE

Subdirectory	Contents
/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_RULESET_PATH

By default, 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 EMC 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

These variables control thread locking in the programs.

SM_COUNT_REPOS_LOCKS

A Manager crashes whenever the stack used by a thread exceeds the space allocated. Threads acquire locks to access instances in the 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 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 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.

Variables that control log subscription changes and Dynamic Modeling errors

These variables control the tracing of subscription changes and the ignoring of DynModel errors.

SM_SUBSCRIPTION_TRACE

This environment variable controls the tracing of subscription changes. Subscription tracing is enabled by default, which means that subscription changes will appear in the server log by default. Similar messages will appear when new clients connect to the server and issue subscriptions.

Valid values are 0 and 1, with a default value of 1. Disabling subscription tracing is not recommended, but can be achieved by setting SM_SUBSCRIPTION_TRACE=0. To enable it again, set SM_SUBSCRIPTION_TRACE=1.

SM_IGNORE_DYNMODEL_ERRORS

This environment variable enables the server to ignore DynModel errors and continue to run. In earlier releases, if you tried to refine an existing attribute, on finding the DynModel error that is not recoverable, the server application (Manager) would stop and write an error message to its log. This behavior is the default behavior in this release.

Valid values are TRUE and FALSE, with a default value of FALSE. Setting SM_IGNORE_DYNMODEL_ERRORS=TRUE causes the server to ignore DynModel errors and continue to run.

APPENDIX B
Wildcards

This appendix shows you how to use a wildcard pattern in a text field to match a number of objects instead of specifying each object by name. It consists of the following sections:

- Types of wildcard patterns 174

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 34 on page 174. 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. Table 35 on page 175 describes compound wildcard patterns.

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.”

NOTICE

Spaces are interpreted as characters and are subject to matching even if they are adjacent to operators like “&”.

Table 34 Basic wildcard patterns (page 1 of 2)

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.

Table 34 Basic wildcard patterns (page 2 of 2)

Character	Description
‹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 35 on page 175.

Table 35 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

A

Access privileges
- All 91
- Monitor 91
- None 91
- Ping 91

Additional safeguards 112
- File permissions 112
- Limiting access to servers 112

Alternate security configuration files
- Specifying 98

ASL script
- SM_RULESET_PATH 171

Audit file 25

Authentication
- record format 89, 91

Authority programs
- Specify selection 166

Availability Manager
- see IP Availability Manager

B

brcontrol utility 119

Broker
- brcontrol 119
- Clients locating 164
- Manager
  - Process ID 118
  - State last changed 118
  - Status 118
- Ping access privilege 91
- Registry 118
- Secure 110, 115
- Security 87
- TCP port 119

BROKER token 90, 92, 95

brokerConnect.conf 87
- Description 91
- Encrypted passwords 104
- Syntax 88

C

Client process locale 15

Client session locale 15

clientConnect.conf
- BROKER token 92
- DEFAULT token 92
- Description 91
- Encrypted passwords 104
- login user field 92
- password field 92
- PROMPT token 92
- record format 91
- Syntax 88
- target field 92
- username field 92
- USERNAME token 92

Configuration files
- runcmd_env.sh 20

Country codes 16

D

Default parameters for services 77

DEFAULT token 90, 92

Diffie Helman-Advanced Encryption Standard 84, 85

Disabling security 100

Displaying installed services 73

dmctl utility 25
- roll_log 143

Domain Manager
- Running as non root 81

E

EDITOR 21

ELOOP error 171

Enabling security 100

Encoding
- Setting for internal interaction with operating system 161
- Setting for shell 161
- Setting for stdin 161
- Used by stdout and stderr 161

Encrypted connections 106
- Configuring 109

Encrypted passwords 104

Encryption 101

encryption standard
- Federal Information Processing Standard (FIPS) Publication 140-2 84, 85

Environment variables
- Boolean values 147
- Date and time formatting 167
- EDITOR 21
- Hexadecimal values 146
- How values are interpreted 146
- Integer values 146
- LC_TIME 168
- LM_LICENSE_FILE 165
- Octal values 146
- runcmd_env.sh 20, 120
- Setting 147
- SM_AUDIT_ALL_CONNECTIONS 157
- SM_AUDIT_CLIENT_ACTIONS 157
- SM_AUTHORITY 100, 166
- SM_BACKUP_FILE_LIMIT 139, 144, 169
- SM_BROKER 164

SM_BROKER_DEFAULT 119, 120, 164
SM_BROKER_NTHREADS 164
SM_CLIENTCONNECT 98, 166
SM_COUNT_REPOS_LOCKS 171
SM_DISABLE_KEEPALIVES 155
SM_DISPLAY 151
SM_EDITOR 21, 151
SM_ENABLE_SNMP_SET 152
SM_ENCODING_INPUT 161
SM_ENCODING_OUTPUT 161
SM_ENCODING_SHELL 161
SM_ENCODING_SYSTEM 161
SM_FOLD_WIDTH 157
SM_HOSTSFILE 158
SM_IGNORE_DYNMODEL_ERRORS 172
SM_INCOMING_PROTOCOL 108, 154
SM_IP_VERSIONS 155
SM_JAVA 163
SM_JAVA_PACKAGE_ROOT 163
SM_JAVAHOME 163
SM_JDK_HOME 163
SM_KEYFILE 166
SM_LICENSE 36, 165
SM_LMGRD_LICENSE_FILE 165
SM_LOCALE 161
SM_LOG_NAME_CUSTOM 170
SM_LOG_ROLL_SIZE 169
SM_LOGFILES 26, 137, 170
SM_MAIN_OPTIONS 137, 152
SM_MAX_MONITORING_THREADS 157
SM_NO_LINE_FOLD 157
SM_OKLOGIN_LOGLEVEL 166
SM_OUTGOING_PROTOCOL 108, 154
SM_REPOS_LOCKS_LIMIT 171
SM_RULESET_PATH 23, 171
SM_SERVERCONNECT 98, 167
SM_SNMP_BUG_COMPATIBLE 151
SM_SUBSCRIPTION_TRACE 171
SM_WRITEABLE 25, 139, 170
TZ 168
VISUAL 21
Error level
Retrieve at runtime 138
Set at runtime 138
Evaluation license file 39
Exception messages
Line wrapping in log file 157

F

Federal Information Processing Standard (FIPS) Publication 140-2 encryption standard 84, 85
Firewall
One-way connection 41
flexlm.log 36, 57
FlexNet Publisher 34
Document (PDF) 34, 52
License server 35, 44
LM_LICENSE_FILE 165
log file 36, 57
Tools 52
FlexNet Publisher SM_LMGRD_LICENSE_FILE 165
Force logging 157

G

Global Console
Client 86
Security prompting 87

H

Host ID
And IP address 58

I

ic-serviced 67
Indicating the location of the JDK 163
install_license script 43, 50
Installing programs as services 71
Integer values 146
IP Availability Manager
Discovery and volume licensing 56
ISO 639 Language Codes 16

J

Java package root specification 163

K

KeepAlive messages 155
kill command 142

L

Language codes 16
LC_TIME 168
Licensing
Block of licenses 54
Components 35
Determining number of licenses in use 58
Discovery and volume licensing 56
Examples of license files 37
flexlm.log file 36, 57
FlexNet Publisher license server 35, 44
FlexNet Publisher software 34
host ID, host_ID 38
Hostname 59
How to obtain
A license file 40
install_license script 43, 50
Installation upgrade 34
Installing license server 43, 50
Multiple sites 47
New installation 34
Number of licenses 38
Permanent license file 34
Renew a temporary license file 40
Requirements 34
For temporary license file 39
Rules for
Permanent license file 37

Running old license files 48
Set up
Permanent 34
SM_LICENSE 36
smarts.lic 37
Starting license server 43, 53
Stopping license server 53
System IP address 58
Temporary license file 39
Tools 52
Travelling license file 39
Troubleshooting 57
Types of daemons 35
Uninstalling license server 51
Upgrade 48
Utilities for volume licensing 54
Verifying version of license server 52
Volume licensing 54
LM_LICENSE_FILE 165
Locale
Client process 15
Client session 15
Definition 14
Process 15
Support 16
Locating the Broker 164
Lock counting 171
Log files 134
Adapter 139
Audit file 25
Controlling the number 139, 144
Date and timestamp 142
Diagnostic header 140
FlexNet Publisher log file 36
Line wrapping of exception messages 157
Manager log file 139
Message at end of rolled log 142
Naming convention 139
Prevent line wrapping 157
Retrieve error level at runtime 138
Retrieve log level at runtime 138
Retrieve trace level at runtime 138
Rolling over 136, 139, 142
Setting error level at runtime 138
Setting log level at runtime 138
Setting trace level at runtime 138
Starting a new one 142
Statistics log file 25
Thread id 142
UNIX syslog 136
Win Event 136
Log level
Retrieve at runtime 138
Set at runtime 138
Log successful logins 166
login user 92

M

Manager
Status
Dead 118
Running 118
Unknown 118
Matching
Pattern 174
Maximum number of monitoring threads in monitoring system 157
Modifying service parameters 76
monitoring performance 157

O

Operator
Wildcard 174

P

Passwords
Encrypting 104
Pattern 174
Pattern matching 174
performance of monitoring system 157
Permanent license file 34
privopen option 81
process ID 142
Process locale 15
PROMPT token 92

R

Removing services 76
roll_log command 143
runcmd_env.sh 20, 120, 147
Running as non root 81

S

Script
install_license 50
Secret phrase 101
Changing 102
Locking 103
Using for multiple suites and sites 103
Security
Access privileges
All 91
Monitor 91
None 91
Ping 91
Automatic authentication 87
Broker 87
brokerConnect.conf
Description 91
Syntax 88
clientConnect.conf
Description 91
Syntax 88
Diffie Helman-Advanced Encryption Standard 84, 85
Enabling or disabling 100
Encryption 88, 101
Global Console 87
Network account authentication 94

serverConnect.conf
Description 89
Syntax 88
Security configuration
Examples 113
Security feature 84
Authentication 84
Encrypted passwords 84
Seed file
Encrypted 106
serverConnect.conf
BROKER token 90, 95
DEFAULT token 90
Description 89
Encrypted passwords 104
password field 90
privilege field 90
record format 89
Syntax 88
SYS token 90
target field 90
username field 90
Setting environment variables 147
sm_adapter 79
common options 79
SM_AUDIT_ALL_CONNECTIONS 157
SM_AUDIT_CLIENT_ACTIONS 157
SM_AUTHORITY 100, 166
Setting the variable 100
sm_authority program 26
SM_BACKUP_FILE_LIMIT 139, 144, 169
SM_BROKER 164
SM_BROKER_DEFAULT 119, 120, 164
SM_BROKER_NTHREADS 164
SM_CLIENTCONNECT 98, 166
SM_COUNT_REPOS_LOCKS 171
SM_DISABLE_KEEPALIVES 155
SM_DISPLAY 151
sm_edit utility 19, 20, 21, 151
Encrypting passwords 105, 106
SM_EDITOR 21, 151
SM_ENABLE_SNMP_SET 152
SM_ENCODING_INPUT 161
SM_ENCODING_OUTPUT 161
SM_ENCODING_SHELL 161
SM_ENCODING_SYSTEM 161
SM_FOLD_WIDTH 157
SM_HOSTSFILE 158
SM_IGNORE_DYNMODEL_ERRORS 172
SM_INCOMING_PROTOCOL 108, 154
SM_IP_VERSIONS 155
SM_JAVA 163
SM_JAVA_PACKAGE_ROOT 163
SM_JAVAHOME 163
SM_JDK_HOME 163
SM_KEYFILE 166
SM_LICENSE 36, 165
SM_LMGRD_LICENSE_FILE 165
SM_LOCALE 15, 161
SM_LOG_NAME_CUSTOM 170
SM_LOG_ROLL_SIZE 169
sm_logerror 26
SM_LOGFILES 26, 137, 170
SM_MAIN_OPTIONS 137, 152
SM_MAX_MONITORING_THREADS 157
SM_NO_LINE_FOLD 157
SM_OKLOGIN_LOGLEVEL 166
SM_OUTGOING_PROTOCOL 108, 154
SM_REPOS_LOCKS_LIMIT 171
SM_RULESET_PATH 23, 171
sm_server
common options 77, 80
SM_SERVERCONNECT 98, 167
sm_service
actions 70
install action 71
install options 71
remove action 76
show action 73
standard options 69
start action 74
start options 74
stop action 74
stop options 74, 75
sm_serviced 67
SM_SNMP_BUG_COMPATIBLE 151
SM_SUBSCRIPTION_TRACE 171
sm_tpmgr command 55
SM_WRITEABLE 25, 139, 170
smarts.lic 37
SNMP 151
Specify non-standard JRE 163
Starting and stopping services 74
Starting license server 43, 53
statistics log file 25
Stopping license server 53
Support locale 16
SYS token 90

T

TCP
Broker port 119
Threads
Maximum number of monitoring threads 157
Time zone 168
Trace level
Retrieve at runtime 138
Set at runtime 138
Trial license file 39
TZ 168

U

Unicode 16
UNIX syslog 136
USERNAME token 92
Utilities
brcontrol 119
dmctl
roll_log 143

sm_edit 19, 20, 151

V

VISUAL 21
Volume licensing 54

W

Wildcard 174
Chart of operators 174
Win Event Log 136
Windows Domain Manager 94
Work-around for 32-bit Solaris library bug 158