

VMware Smart Assurance Security Configuration Guide

VMware Smart Assurance 10.1.7

You can find the most up-to-date technical documentation on the VMware by Broadcom website at:

<https://docs.vmware.com/>

VMware by Broadcom

3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Copyright © 2024 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, go to <https://www.broadcom.com>. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies. [Copyright and trademark information.](#)

Contents

- 1 Overview 5**
 - Security features 5
 - Security enhancements 5
 - How security is implemented 6
 - Server and client programs for Smarts 7
 - Server authentication 8
 - Client authentication 8
 - Encryption 9

- 2 Secure deployment and usage settings 11**
 - A basic Smarts secure deployment 11
 - Secure Deployment Settings 12
 - Security mechanism and communication protocols 12
 - Considering security and firewalls 13

- 3 Security Configuration Settings 14**
 - Access Control Settings 14
 - Authentication 14
 - Authorization 25
 - Component Access Control 28
 - Log settings 28
 - Log filenames, locales and encodings 28
 - Retrieving and setting log, error and trace levels at runtime 29
 - Retrieving the current level 29
 - Setting the level 30
 - Communication Security Settings 30
 - Communication protocols overview 31
 - Enabling FIPS 140 32
 - Disabling FIPS 140 33
 - Verify FIPS-140 status 34
 - Re-encrypting security files and seed files 34
 - Using the secret across large number of sites 35
 - Using the secret phrase for multiple products and sites 36
 - Locking the secret phrase 36
 - Encrypted passwords 36
 - Encrypting seed files 38
 - Encrypted connections 38
 - Port usage 42

Disabling the RC4 algorithm with a cipher suite list	43
Data Security Settings	45
Encryption of data at rest	45
Additional Security Considerations	45
Preventing Elasticsearch from being vulnerable to dynamic scripting	45
Importing TLS certificates	45
Pseudo-random number generators (PRNG) algorithm	46
Configuring a secure Broker	46
Setting the SM_AUTHORITY variable	47
Specifying alternate security configuration files	48
Options to run as non-root on Linux systems	49
4 Security Maintenance	51
Routine password maintenance	51
Change the secret phrase	51
5 Physical Security Controls	52
Physical Security at customer site	52

Overview

1

This guide provides overview of security configuration settings available in the product, secure deployment and usage settings, secure maintenance and physical security controls needed to ensure secure operation of the product. This chapter includes the following topics:

Read the following topics next:

- [Security features](#)
- [Security enhancements](#)
- [How security is implemented](#)

Security features

Smart Assurance software provides multiple levels of security. The administrators can configure the software security to secure the system and control access to it as appropriate for their needs.

The various levels of security can be implemented by any one of these ways:

- Authentication and user privileges
- Encryption
- Secure communication

The security features are enabled with default settings when you install the software. The default security settings permit access to the system. Therefore, change the default usernames, passwords, and the secret phrase to enforce access control to the software. In addition, restrict access to the security configuration files.

Security enhancements

This sections provides information on security enhancements in VMware Smart Assurance.

The following components are upgraded for VSA Domain Managers:

- Common-fileupload has been upgraded to 1.4.
- Log4J has been upgraded to 2.17.0.

- Mujs has been upgraded to 1.1.3.
- Netty has been upgraded to 4.1.67.
- Quartz has been upgraded to 2.3.2.
- Snappy-java has been upgraded to Glibc compatible version 2.33.
- Slf4j has been upgraded to 1.7.32.
- Snappy-java has been upgraded to Uclibc compatible version 2.33.
- Zlib has been upgraded to 1.2.11.
- Java has been upgraded to 11.0.12.
- Icu4J has been upgraded to 52.2.

How security is implemented

Smart Assurance security features apply to authentication and user privileges, encryption and secure communication.

Authentication and user privileges

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 to the server. If the connection is allowed, then privileges are granted to the client.

Encryption

All Smart Assurance passwords are encrypted. 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 Smart Assurance software installation. Encryption is based on a secret phrase, common to all of the Smart Assurance applications that must interact, and is used to encrypt password fields in the authentication records. SNMPv3 passwords can also be encrypted in seed files.

Secure communication

Secure connections are implemented through:

- 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.

A FIPS 140 enabled Broker is able to communicate with a FIPS 140 enabled Domain Manager as well as a non-FIPS 140 enabled Smarts Domain Manager. In addition, a FIPS 140 enabled Smarts Domain Manager is able to communicate with a FIPS 140 enabled Domain Manager as well as a non FIPS 140 enabled Domain Manager.

Server and client programs for Smarts

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

Table 1-1. Smart Assurance server and client programs

Server Programs	Client Programs
Broker	SAM Failover System
	Command-line utilities such as dmctl and brcontrol
	Global Console users
	Global Manager
	Notification Adapters
	Service Assurance Adapter Platform
	Tools
	Report Manager
Global Manager	SAM Failover System
	Broker
	Command-line utilities (dmctl)
	Global Console users
	Notification Adapters
	Tools
	Global Manager
	Report Manager
Smart Assurance Manager Adapter Platform	Broker
	Command-line utilities (dmctl)
	Global Manager
	sm_ems
	SNMP Trap Adapter

Table 1-1. Smart Assurance server and client programs (continued)

Server Programs	Client Programs
	Syslog Adapter
Smart Assurance IP Availability Manager	Global Console
	Global Manager
Smart Assurance IP Performance Manager	Global Console
	Global Manager

Server authentication

A server authenticates a connection request by comparing the username and password it receives from a client with the authentication records in the file.

When a client program initiates a connection to a server, it provides the server with a username and password. The server authenticates the client, by comparing this username and password with the authentication records in the `serverConnect.conf` file. 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 that any changes to the file take effect immediately.

The Global Console always displays a log-on dialog box where a user must enter a 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, then Service Assurance Manager accepts the connection and grants 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

Client authentication involves sending the client authentication information, stored in the file, to the server. For the , the client connection file is the.

Other than the Global Console, most clients automatically send authentication information to the server by default. 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 communicates the access privilege to the client.
- 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. You can edit the configuration files any time and the changes take effect immediately.

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

For example, when a Domain Manager (such as an Smart Assurance IP Availability Manager) registers with a Broker, the Domain Manager sends a username and password from its `clientConnect.conf` file to the Broker. The Broker checks the username and password against the records in its `serverConnect.conf` file. Based on the results, it grants or denies a connection.

However, when the Broker checks whether a registered Domain Manager is alive by pinging the Manager, the Broker authenticates 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 grants or denies the permission to ping it.

Encryption

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

The transformation of the site secret into the `imk.dat` files differs, depending on whether encryption is set to FIPS 140 mode.

Note Smart Assurance programs in FIPS 140 mode encryption cannot use the same `imk.dat` file that is used by Smart Assurance programs in non-FIPS 140 mode.

Therefore, all clients and servers using an `imk.dat` file must be set to the same FIPS 140 mode.

The Smart Assurance programs use the site secret to:

- Encrypt passwords in the configuration files.
- Encrypt passwords for SNMP v3 devices in the seed files.
- Encrypt connections between programs.

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

Not a secret

The `imk.dat` file can be copied. The `imk.dat` is located in the `BASEDIR/smarts/local/conf` directory.

Note 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, change the secret phrase by using the `sm_rebond` utility. Thereafter, change the secret phrase periodically to maintain a secure system.

Note Treat the secret phrase with the same care as a root password or highest level system administration password.

Secure deployment and usage settings

2

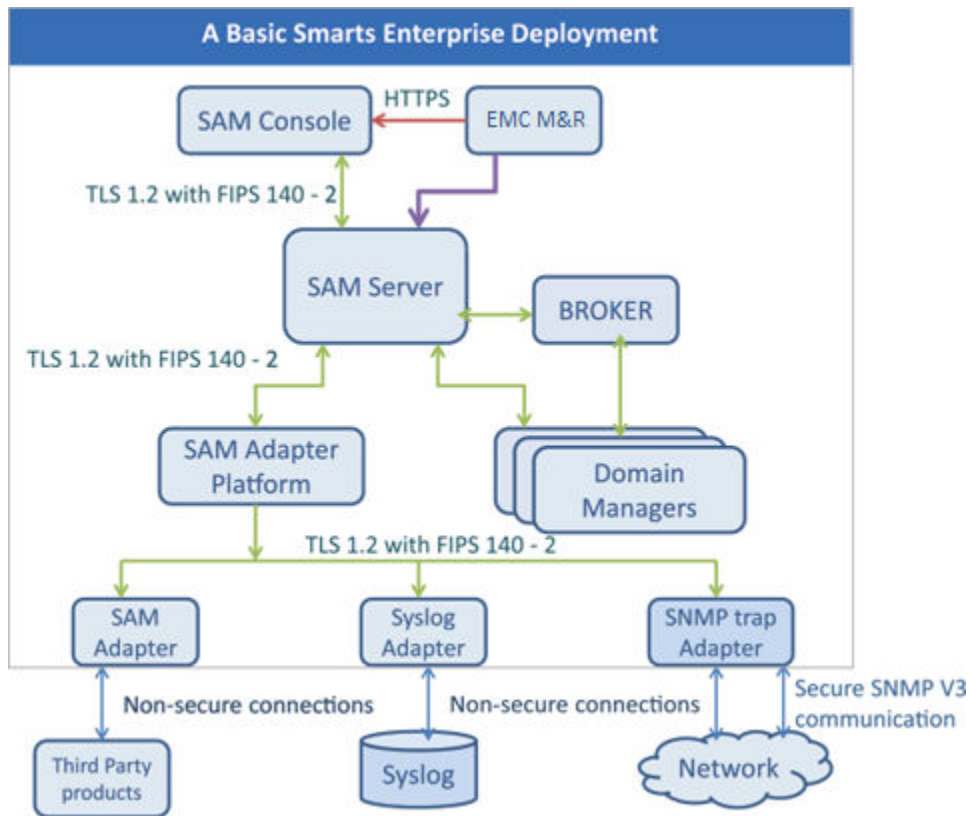
Read the following topics next:

- A basic Smarts secure deployment
- Secure Deployment Settings
- Security mechanism and communication protocols
- Considering security and firewalls

A basic Smarts secure deployment






This topic provides an overview of a basic deployment.

Figure 2-1. A basic Smarts secure Deployment



This is a basic diagram of an Smart Assurance deployment. It shows various Smarts components. The arrows depict the communication protocol and their security. [Table 2-1. Depiction of basic Smarts secure deployment](#) provides details.

Table 2-1. Depiction of basic Smarts secure deployment

Representation	Depiction
	Transport Layer Security (TLS) communication. TLS communication can be configured with FIPS 140 mode.
	SNMP v3 communication
	Non-secure communication
	HTTPS communication
	Non-encrypted communication. Authentication required.

Secure Deployment Settings

The table Secure Deployment Settings describes the default settings of products.

Table 2-2. Secure Deployment Settings

Smart Assurance version	Default settings	Secure Deployment settings	Pros and Cons of Secure Deployment settings
10.1.7, 10.1.5, 10.1.2, 10.1.0, and 10.0.0	<p>Communication is possible in the following ways:</p> <ul style="list-style-type: none"> ■ Transport Layer Security. ■ Communication over Smarts crypto channels, for interoperability with older versions of clients/domains. ■ Authenticated, un-encrypted (Cleartext) mode for interoperability with older versions of clients/domains. 	<p>Restrict usage of the less secure communication modes such as Cleartext, keeping the interoperability requirements in mind.</p>	<p>Pros: Provides better security in communication.</p> <hr/> <p>Cons:</p> <ul style="list-style-type: none"> ■ Incompatible with the previous releases of the software ■ Slower than cleartext.
	<p>Non-FIPS mode</p>	<p>The cryptographic settings used in non-FIPS 140 mode is secure. If FIPS validated cryptography is required, enable FIPS 140 mode.</p>	<p>Pros: Has all the advantages of encryption levels provided for non- FIPS mode.</p>

Security mechanism and communication protocols

The table [Security mechanism and communication protocols](#) lists the communication protocols and security mechanism adopted by Smart Assurance products.

Table 2-3. Security mechanism and communication protocols

Smart Assurance Domain Manager	Protocols	Security mechanism
IP Availability Manager, Server Manager, Network Protocol Manager, MPLS Manager, VoIP Manager	<ul style="list-style-type: none"> ■ CLI (Telnet, SSH) ■ SNMP v1, v2, v3 <p>Note Recommended to use SNMP v3 for enhanced security</p>	All SNMP v3 passwords are stored in the seed files.
	RestAPI (IP server)	Used for OpenStack feature.
Smart Assurance Server Manager	WMI	Uses Windows Management Instrumentation for managing Microsoft Clusters.
	RestAPI	<ul style="list-style-type: none"> ■ Used for OpenStack feature. ■ Used for Scale IO feature.
EMC M&R	HTTP, HTTPS	For EMC M&R to Service Assurance Manager Console.
	CAS	For authentication and authorization.

Considering security and firewalls

When planning your deployment, consider the following security and firewall settings:

- Configure security policies (rules) to enable a two-way connections between the Broker and the various Domain Managers. For communication between Managers across firewalls, plan on opening a hole in the firewall for the Smart Assurance communications. Certain UDP and TCP ports must be opened for proper communications:
 - Broker: Port 426
 - Manager: one port each, which can be configured
 - Adapters, including SNMP Trap Adapter and Syslog Adapter.
- Consider proxy servers when communicating with Smart Assurance applications that reside behind firewalls. Using a proxy server reduces the number of firewall ports that need to be opened to one firewall port.
- If access lists are used, plan on deploying the IP addresses of hosts that include Managers to the access list of devices that will be managed. Smart Assurance applications must have full access to browse the MIBs of the devices. The *VMware Smart Assurance IP Manager User Guide* lists the specific MIBs. Depending on the network size and complexity, this may require scheduling to obtain support from the organization's network personnel.
- You must have a listing of SNMP versions and related security parameter values that are used by specific devices in the organization's network.

Security Configuration Settings

3

This chapter provides an overview of the settings available in the product to ensure a secure operation of the product. Security settings are split into the following categories:

Read the following topics next:

- [Access Control Settings](#)
- [Log settings](#)
- [Communication Security Settings](#)
- [Disabling the RC4 algorithm with a cipher suite list](#)
- [Data Security Settings](#)
- [Additional Security Considerations](#)

Access Control Settings

Access control settings enable the protection of resources against unauthorized access.

Access control settings include:

- [Authentication](#)
- [Authorization](#)
- [Component Access Control](#)

Authentication

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

Default user account

[Default user account](#) lists the default user accounts and passwords.

Table 3-1. Default user account

Default account	Password	Description
admin	changeme	Default account for Smart Assurance server. Privilege - ALL
BrokerNonsecure	Nonsecure	Default account for Broker. Privilege - ALL

User action performed without authentication

[User action performed without authentication](#) lists the user actions that do not require authentication.

Table 3-2. User actions performed without authentication

User action	Description of component
View Smart Assurance documents	You can connect to the Smart Assurance server through a browser by providing the server port details in the browser to view documents.
Applets (java class, jar file); class libraries (jar files)	Serve an HTTP request for applets

Controlling authentication

The environment variable controls the authentication security features provided by the software. You can set the value of this variable to configure authentication. If is not specified, is assumed.

Note The authentication features are enabled by default.

The environment variable `SM_AUTHORITY` has the following syntax:

```
SM_AUTHORITY=IDENTIFY=<program1>,AUTHENTICATE=<program2>
```

Where *<program1>* is the program granting credentials and *<program2>* is the program validating the credentials. 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. The angle brackets (< >) are required.

Authenticating users with a valid system account

The default authentication record in `serverConnect.conf` file permits users with a valid login account on the system where the server is running, to connect.

A client who uses this authentication record must enter a valid system username and password.

Note For authentication, the system username becomes the Smart Assurance username. This record provides monitoring privileges.

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

Authentication prompt

Prompt for a username and password is required for authentication of non-console applications, which can be set in the file.

The corresponding fields in `clientConnect.conf` file must contain <PROMPT> for the username and password 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`.

Authenticating users interactively

The authentication records in `clientConnect.conf` file allows an interactive way to establish connection by prompting the user to provide a username and password.

This authentication record in the `clientConnect.conf` prompts users to enter an Smart Assurance 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. Non-interactive programs skip this record because they cannot prompt for information.

This record could match the following `serverConnect.conf` authentication records:

- * : maint : maint : Monitor
- * : oper : oper : Monitor
- * : * : <SYS> : Monitor

Server authentication

A server authenticates a connection request by comparing the username and password it receives from a client with the authentication records in the file.

When a client program initiates a connection to a server, it provides the server with a username and password. The server authenticates the client, by comparing this username and password with the authentication records in the `serverConnect.conf` file. 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 that any changes to the file take effect immediately.

The Global Console always displays a log-on dialog box where a user must enter a 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, then Service Assurance Manager accepts the connection and grants the privileges that are specified by the authentication record. If the username and password do not match an authentication record, the connection is refused.

`serverConnect.conf` file

The 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.

Format of a record in `serverConnect.conf`

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

Field description of parameters in `serverConnect.conf` file

Table 3-3. Field descriptions for `serverConnect.conf`

Field	Definition	Value
<code><target></code>	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: <ul style="list-style-type: none"> ■ <code><BROKER></code> indicates that this record applies only to the Broker. ■ <code>~<BROKER></code> indicates this record applies to all servers except the Broker.
<code><Smart Assurance username></code>	Username for the client requesting a connection.	Can be a matching pattern with wildcards or the following special value: <ul style="list-style-type: none"> ■ <code><DEFAULT></code> is provided for legacy clients that cannot send a username. ■ <code><AUTO></code> is provided for site-specific credentials.
<code><password></code>	Password for the user requesting a connection.	Can be a password or one of the following special values: <ul style="list-style-type: none"> ■ <code><SYS></code> 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. ■ <code><DEFAULT></code> is provided for legacy clients that cannot send a password. ■ <code><AUTO></code> is provided for site-specific credentials.
<code><privilege></code>	Access privileges of the client.	Valid values include: <ul style="list-style-type: none"> ■ All ■ Monitor ■ None ■ Ping

Note During the authentication process, the server receives a connection target, Smart Assurance 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.

Client authentication

Client authentication involves sending the client authentication information, stored in the file, to the server. For the , the client connection file is the.

Other than the Global Console, most clients automatically send authentication information to the server by default. 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 communicates the access privilege to the client.
- 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. You can edit the configuration files any time and the changes take effect immediately.

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

For example, when a Domain Manager (such as an Smart Assurance IP Availability Manager) registers with a Broker, the Domain Manager sends a username and password from its `clientConnect.conf` file to the Broker. The Broker checks the username and password against the records in its `serverConnect.conf` file. Based on the results, it grants or denies a connection.

However, when the Broker checks whether a registered Domain Manager is alive by pinging the Manager, the Broker authenticates 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 grants or denies the permission to ping it.

clientConnect.conf and brokerConnect.conf files

The `clientConnect.conf` and `brokerConnect.conf` files contain authentication information about the client and the broker respectively.

By default, the `clientConnect.conf` and `brokerConnect.conf` 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.

Note The Broker only uses `brokerConnect.conf` to send authentication to other processes so that it can ping them.

Format of a record in `clientConnect.conf` or `brokerConnect.conf`

```
<login user>:<target>:<ITOps username>:<password>
```

Field description of parameters in `clientConnect.conf` and `brokerConnect.conf`

Table 3-4. Field descriptions for `clientConnect.conf`

Field	Definition	Value
<code><login user></code>	System login name of the person or process attempting a connection.	Can be a matching pattern with wildcards.
<code><target></code>	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: <ul style="list-style-type: none"> ■ <code><BROKER></code> indicates that this record applies only to the Broker. ■ <code>~<BROKER></code> indicates this record applies to all servers except the Broker.
<code><Smart Assurance username></code>	Username that is sent to the server for authentication.	Can be a username or one of the following special values: <ul style="list-style-type: none"> ■ <code><USERNAME></code> indicates that the username under which the current process is logged in, is sent as the Smart Assurance username. ■ <code><PROMPT></code> indicates that the client program asks the user to provide an Smart Assurance username. ■ <code><AUTO></code> is provided for site-specific credentials. ■ <code><DEFAULT></code> mimics legacy client authentication.
<code><password></code>	Password that is sent to the server for authentication.	Can be a password or one of the following special values: <ul style="list-style-type: none"> ■ <code><PROMPT></code> indicates that the client program asks the user to provide a password. ■ <code><AUTO></code> is provided for site-specific credentials. ■ <code><DEFAULT></code> mimics legacy client authentication.

Note A Smart Assurance program runs under the login name of the user who started it. This can have the following 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.
- The Smart Assurance Broker listens on port 426 by default.

Note 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 Smart Assurance username in the `clientConnect.conf` file or an Smart Assurance username and password will not be sent to a server for authentication. In the `clientConnect.conf` record, a user's login name and Smart Assurance username do not have to be identical.

Special authentication values

Special authentication involves the configuration of the , and by using the special values <SYS>, <PROMPT>, <AUTO>, and <DEFAULT>. These values control system, site-specific and legacy authentication.

The special authentication values are associated with the following types of authentication:

- System authentication
- Network Account Authentication
- Authentication prompt
- Site-specific authentication
- Legacy system authentication

System authentication

System authentication uses the username and password of the operating system to authenticate clients. System authentication is configured in file.

The System authentication method gives every console operator an account on the host on which the server runs. By using this method:

- Each console operator has a unique username and password.
- Access to the system can be traced to a particular user.
- Access can be individually revoked.

Note 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 the `serverConnect.conf` file:

```
* : 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 who provide the correct password are granted `Monitor` access.

You could even define a class of administrative users. For example, usernames that start with `ADM` are provided `All` access. All other users are provided `Monitor` access.

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

In console applications, the passwords are displayed as masked characters (*) to avoid displaying the passwords in cleartext.

Note In order for console applications to prompt, the value `<PROMPT>` must be used for the Smart Assurance username and password in `clientConnect.conf` file.

Network account authentication

Network account authentication involves username authentication based on its password.

In UNIX systems, the system authenticates a username based on its password.

Authentication prompt

Prompt for a username and password is required for authentication of non-console applications, which can be set in the file.

The corresponding fields in `clientConnect.conf` file must contain `<PROMPT>` for the username and password 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`.

Legacy system authentication

Legacy system authentication provides the ability to interoperate with older software that may not support authentication, by providing a 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 as any other username/password combination validation.

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

Authenticating site-specific connections

Site-specific authentication involves the use of a site's secret to authenticate connections.

When corresponding records in `serverConnect.conf` and `clientConnect.conf` contain the `<AUTO>` value for both the Smart Assurance username and password, the client generates a password by using its secret. The server validates the password to accept or reject the connection.

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

Procedure

- 1 For connections to a particular Domain Manager, for example, Service Assurance Manager, define a record in the `serverConnect.conf` file.

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

- 2 Define a corresponding record in the `clientConnect.conf` file on the different hosts connecting to the server:

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

Default authentication records

Default authentication records are enabled by default. These are user records in , and automatic authentication setup in and .

This includes:

- Legacy software authentication
- Automatic client authentication to the Broker
- Automatic Broker authentication to servers
- Administrative users authentication

Authenticating Global Console users

This topic lists the two records in file that provide the with monitoring access to servers.

The first record authenticates console users who provide the Smart Assurance username "maint" and the password "maint".

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

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

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

Consoles do not use `clientConnect.conf` for authentication as there is no automatic authentication. The console prompts for a username and password and passes the information to the server for authentication.

Note User profile restrictions can be used to further limit certain Global Console operations.

Authenticating legacy software

You can authenticate legacy software connection in the file.

The Broker uses the following record in `serverConnect.conf` file to authenticate legacy client connections and provide full access. The target field, with the value `<BROKER>`, identifies that the Broker is the sole target for authentication.

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

Note There are no corresponding entries in the `clientConnect.conf` or `brokerConnect.conf`.

In this configuration, the Global Console does not prompt for a username and password when connecting to the Broker. The default configuration defines a `nonsecure Broker`, equivalent in security to Smart Assurance software prior to version 5.0.

Authenticating administrative users

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

This record authenticates clients that provide the Smart Assurance 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.

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

Note 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 after you have set up a corresponding administrative account in the Global Manager. Do not remove or disable an administrative account in the Global Manager.

Automatic Broker authentication to servers

The automatic authentication record in file handles automatic 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
```

Syntax and content of security files

This topic describes the syntax of the , and files.

Each file consists of one or more authentication records, each of which contains four fields:

- A line that starts with two forward slashes (`//`) or a pound sign (`#`) is considered a comment and ignored.
- The fields of each authentication record are separated by colons (`:`).

- Any white space before, after, or between fields, is ignored. If the field value contains a space, you need to add an escape character, the backslash (\), before the space. When a backslash is encountered, the following character loses any special significance and is used as is.

Note The first line of each of the configuration files contains encryption information. Do not change this line unless you want to disable encryption. To comment out the line, add a second pound sign.

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.

Authorization

User authorization settings control rights or permissions that are granted to enable a user to access a resource managed by the product. Only authenticated users are allowed to access an product and network. The authenticated users must be granted the appropriate permissions (or privileges) for authorization to access the application's features and functions. Access to the application is through the user interface (UI) or is a CLI based access.

Users and security

When defining the functional categories for users, consider the security implementation. For each user category, determine the following:

- List of the specific Smart Assurance managers that must be accessed by the users in the category. If necessary, divide a category. For example, in a category of network administrators, some users require access to managers in Asia, other users require access to managers in Europe, and still other users require access to all managers. In this case, the network administrators category could be divided into three categories to match these user needs while maintaining tighter security.
- The functionality that must be accessed by users in this category. Smart Assurance software is designed in such a way that users can be classified into any one of the following access levels:

Impersonate

An access level that allows a login to perform actions on behalf of other users. It is used only by the Smart EDAA. This access level has all the powers of All, plus the ability to execute actions as other users.

All

An access level where users can access all Global Console functionality available for one or more Smart Assurance Managers, if their user profile permits it.

This access level 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 Smart Assurance provides additional information about using user profiles to restrict access to console operations.

Monitor

An access level at which users can access only Global Console monitoring functionality, and not administrative functionality, for one or more Smart Assurance Managers, if their user profile permits it.

This access level 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`).

Ping

An access level normally reserved for Smart Assurance processes.

This access level allows processes ping the hosts where other Smart Assurance processes are installed to determine if the hosts are running. The access level 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 this is sufficient to allow the `dmctl` command line utility to connect to a server and execute the ping command.

None

An access level that specifically excludes access to the Global Console server. This privilege can be used to explicitly prevent a user from accessing the server.

These types of security access are defined in the `serverConnect.conf` file located in the `BASEDIR/smarts/local/conf` directory on the servers where Smart Assurance software is installed.

User profiles

User profiles combine access to notification lists, console operations, custom console layouts, and specific tools. It is a functional grouping of users and their requirements.

You can create a profile for each category of user that you support. Groups of users with similar needs can then be assigned the same user profile.

If needed, you can further customize a generic user profile by copying it and then modifying it for more specific needs. For example, an administration user profile could be customized for less experienced administrators by restricting access to some administrative console operations and tools. Other possible user profiles could include regional or customer-specific consoles.

Smarts Console and user types

consoles have different capabilities. For each type of user and their equipment, determine which console is most appropriate:

- Global Console provides all functions and all views to operators. Administrators can restrict console operations so that operators have only the specific functionality that they require.
- Web Console provides all functions and all views that are available from the Global Console, except the ones from Topology Browser Console. As with the Global Console, most console operations can be restricted.
- Business Dashboard provides access to the results of Smart Assurance analysis through one or more Dashboard viewlets, implemented as Java applets. A Dashboard viewlet is a Java application that shows a particular aspect of Service Assurance information such as notifications, containment, or maps. Dashboard viewlets provide operational functions, but no administrative functions.

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.

Limiting access to the configuration files

Configure security in such a way that each security file is readable only by those programs or users that require it. The security configuration files installed with software, which should be edited after installation, are readable by anyone.

The Managers and the Broker typically run with administrative privileges. Therefore, ensure that the `serverConnect.conf` and `brokerConnect.conf` files are readable by users with administrative privileges only.

Procedure

- 1 Create a secure setup for users and client programs by providing two separate `clientConnect.conf` files.

Note One `clientConnect.conf` file, can remain readable by anyone, should only contain entries that make client programs prompt for passwords. This `clientConnect.conf` file will not contain passwords.

- 2 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 option to the command 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 option before authenticating.

Using file permissions

You can limit access to the security configuration files by using the permissions feature of the operating system. Allow only selected users to access the files. The users must include those who launch the applications requiring access to the authentication records.

Component Access Control

Component access control settings define control over access to the product by external and internal systems or component.

Log settings

A log is a chronological record of system activities sufficient to enable the reconstruction and examination of the sequence of environments and activities surrounding or leading to an operation, procedure, or event in a security-relevant transaction, from inception to final results.

Log settings include:

- [Log filenames, locales and encodings](#)
- [Retrieving and setting log, error and trace levels at runtime](#)
- [Retrieving the current level](#)
- [Setting the level](#)

Log filenames, locales and encodings

This table summarizes the filename, locale and encoding of the log files that are produced, including the system log containers.

Table 3-5. Log filenames, locales and encodings

Log use	Log filename (if <code>-output</code> 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

Retrieving and setting log, error and trace levels at runtime

This topic describes the computed attributes available for 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 the table [Table 3-6. Computed attributes to retrieve and set log, error, and trace levels at runtime](#) are available on the `SM_JIIM_Support` object, and are therefore available from within `JIIIM` 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 3-6. Computed attributes to retrieve and set log, error, and trace levels at runtime

Computed Attributes	Description
<code>logLevel</code>	The minimum exception level for sending messages to the system error logger. The <code>logLevel</code> attribute is a string, and can be any one of the values set for the <code>--loglevel</code> command line option.
<code>errLevel</code>	The minimum exception level for writing messages to the log files. The <code>errLevel</code> attribute is a string, and can be any one of the values set for the <code>--errlevel</code> command line option.
<code>traceLevel</code>	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 <code>traceLevel</code> attribute is a string, and can be any one of the values set for the <code>--tracelevel</code> command line option.

Retrieving the current level

This topic provides the procedure to retrieve the current log, error, and trace levels.

Procedure

- ◆ 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.

Example

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
```

Setting the level

This topic provides the procedure to change the current log, error, and trace levels.

To change the current log, error, and trace levels,

Procedure

- ◆ 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.

Example

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:

```
[September 27, 2013 2:04:11 AM EDT +281ms] t@1158101312
SM_ProtocolEngine-5203 21295 dmctl JM_MSG-*--JM_TRACE_LEVEL_CHANGED-User
'sebasg', using remote dmctl client (id 5203), on host killian.lss.emc.com
with credentials admin has changed the Trace level to None; in file "/emc/
sebasg/svn/main/smarts/repos/jiim/SM_JIIM_Support_Impl.c" at line 459
```

Communication Security Settings

Communication security settings allow the establishment of secure communication channels between product components as well as between product components and external systems.

Communication security settings include:

- [Encryption](#)
- [Enabling FIPS 140](#)

- Disabling FIPS 140
- Verify FIPS-140 status
- Re-encrypting security files and seed files
- Locking the secret phrase
- Changing passwords in security configuration files
- Encrypting seed files
- Encrypting connections in FIPS 140 mode
- Encrypting connections in non-FIPS 140 mode
- Disabling the RC4 algorithm with a cipher suite list

Communication protocols overview

Smarts domain managers have the capability to communicate over many transport protocols. Smarts and Watch4net supports 1.0, 1.1 and 1.2 versions of TLS (Transport Layer Security). SSLv3 is strictly not allowed due to security vulnerabilities such as the POODLE vulnerability.

The latest Smarts version uses TLSv1.2, while some of the past versions used TLSv1.1 and TLSv1.0. Before that, Smarts client server interaction was done over proprietary implementations of some of the popular cryptographic protocols such as Diffie-Hellman key exchange, AES, and so on, as well as in "cleartext" mode (with no encryption). For enhanced security, the user should configure Smart Assurance to choose the most secure protocols and disable the weak protocols, keeping backward compatibility in mind. This can be achieved by using a combination of the following environment variables:

SM_TLS_PROTOCOLS

Configure which versions of TLS protocols to allow and disallow. Without this flag, only TLS1.2 is allowed. For example, only to illustrate the point and recommend a configuration, `SM_TLS_PROTOCOLS=-TLSv1.2,+TLS1.0,+TLS1` will disallow TLS1.2 and allow TLSv1.1 and TLSv1.0.

SM_ALLOW_LEGACY_CRYPTO

Set this environment variable to TRUE to allow communication in non-FIPS mode using the proprietary protocols. Without this flag, all proprietary protocols including cleartext mode are disabled. Once enabled, it is possible to choose different levels of security within the set of proprietary protocols using the `SM_INCOMING_PROTOCOL` and `SM_OUTGOING_PROTOCOL` environment variables. The VMware Smart Assurance System Administration Guide provides details on these protocols. This setting has no effect in FIPS mode since the proprietary protocols are strictly disabled in FIPS mode.

SM_TLS_SUITE_LIST

A colon-separated list of the cipher suites that are allowed in TLS communication. A cipher suite is a suite of cryptographic algorithms used to provide encryption, integrity

and authentication. This environment variable needs to be used only if some vulnerability is found on some cryptographic function, which must be disabled. By default, Smart Assurance supports many cipher suites. The current preferred cipher suite uses RSA algorithm for Key-Exchange, RSA algorithm for Signature, AES-GCM(256) for encryption and SHA384 for message digest. The supported list of cipher suites, and the order in which they are prioritized, may change with each release. Contact customer support to get information on other supported cipher suites.

As of release 10.1, you can specify a cipher suite list as an alternative to the RC4 algorithm used for TLS communication. Using a cipher suite list disables the RC4 algorithm. To do so, perform the procedure described in [Disabling the RC4 algorithm with a cipher suite list](#).

As of release 9.4, the following environment variables are set by default in new product installations in the `BASEDIR/smarts/local/conf/runcmd_env.sh` file:

- `SM_ALLOW_LEGACY_CRYPTO=TRUE`
- `SM_TLS_PROTOCOLS=+TLSv1.1`

Depending on your deployment, ensure that the `BASEDIR/smarts/local/conf/runcmd_env.sh` file includes the environment variables, `SM_TLS_PROTOCOLS` and `SM_ALLOW_LEGACY_CRYPTO`.

Use `SM_TLS_PROTOCOLS` set to the `+TLSv1.1` value only if you need to interoperate with Smart Assurance products based on Foundation 9.0.0.0 Build 1345 through 9.2.x.

Use `SM_ALLOW_LEGACY_CRYPTO` set to `TRUE` only if you need to interoperate with Smart Assurance products based on Foundation versions prior to 9.0.0.0 Build 1345 and if your deployment includes EMC M&R.

Enabling FIPS 140

FIPS 140 mode is disabled after installation of any product. You can enable FIPS 140 on a clean installation or on an upgrade, and before the broker is started.

Prerequisites

Installation of any Smart Assurance product.

Procedure

- 1 Back up the `imk.dat`, `brokerConnect.conf`, `serverConnect.conf`, and `clientConnect.conf` files from the existing installation.

These files are located in the `<BASEDIR>/smarts/local/conf` directory.

Note The backup is necessary in case you need to disable FIPS 140 mode and remove FIPS 140-2 encryption.

- 2 Run the following command at the command line prompt: `sm_rebond --upgrade --basedir=/opt/InCharge/<product>/smarts`

The path must be set to the default install path.

Note Invoke the `sm_rebond` command from the BASEDIR where the software is installed and not from any other product installation area which may have the `sm_rebond` utility, regardless of the FIPS 140 state.

- 3 When prompted, type `Not a secret` as the password phrase password to regenerate the `imk.dat` file.
- 4 Download and install the Java 8 Unlimited Strength Jurisdiction Policy JAR files. These JAR files are required for the FIPS 140 mode for the console, web server, and anything else using Java. The policy files used with earlier releases will not work.

Note Manual download of Java 8 Unlimited Strength Jurisdiction Policy JAR files `local_policy.jar` and `US_export_policy.jar` is not required for anything in the 9.4.x release including the FIPS 140 mode for the console or web server. This manual step is needed only for deployments that use NAS discovery in IP domain manager. For more details refer to NAS chapter in the installation guide. The policy files used with earlier releases will not work.

- 5 Set `SM_FIPS140=TRUE` in the `runcmd_env.sh` file located in the `<BASEDIR>/smarts/local/conf` directory.

Note If you install the server as a service on Linux platforms, the services will start automatically after you issue the `sm_rebond` command. First stop the services, modify `SM_FIPS140=TRUE` in the `runcmd_env.sh` file, and then manually start the services.

- 6 After you enable FIPS 140 mode, start the Broker, and then the server.

The following message may appear in the server log: "CI-W-NOCGSS-No certificate loaded for <Smarts product>, generating self-signed certificate".

Note Since FIPS 140 requires secure communication which can be achieved by SSL, a certificate is required. If a certificate is not available, the <Smarts product> generates a self-signed certificate.

Disabling FIPS 140

You can disable FIPS 140 mode.

Prerequisites

Replace the `imk.dat`, `brokerConnect.conf`, `serverConnect.conf`, and `clientConnect.conf` files in the `<BASEDIR>/smarts/local/conf` folder, with the copies saved from prior to enabling FIPS 140.

Note If you do not have a copy of these files saved, contact technical support.

Procedure

- 1 Set the value for the `SM_FIPS140` parameter to `FALSE` in the `runcmd_env.sh` file.
- 2 Downgrade the `imk.dat` and the `*Connect.conf` files with encrypted passwords.

Note RPS files started under FIPS mode cannot be re-used in non-FIPS mode. Domains will have to be started either from scratch or pre FIPS RPS files can be used in cases where topologies have not changed. Restoring from older RPS files may not be productive as it will not contain any recent topology.

- 3 Restart all processes, such as the domain, Broker, and the console.

Note Do not remove the Unlimited Strength Jurisdiction Policy JAR files.

Verify FIPS-140 status

You can verify the FIPS 140 status by using a command.

Procedure

- ◆ To know the FIPS 140 status, type the following command in `dmctl` mode: `dmctl -s <server name> get SM_System::SM-System::FIPS`

The output should be `TRUE`.

- ◆ `TRUE` implies that FIPS140 is enabled
- ◆ `FALSE` implies that FIPS140 is disabled.

When the broker and server starts on FIPS 140 mode, the following information is written in the broker and server log files: `RSA BSAFE: MES 3.2.4 26-November-2010, Nov 26 2010 11:25:56`

`FIPS: RSA BSAFE Crypto-C Micro Edition FIPS 140-2 Module 3.0.0.0, May 31 2008 13:40:49`

Re-encrypting security files and seed files

The utility re-encrypts the security files and the seed files, then restarts the processes once the phrase and encryption changes are made. Use the utility to change the old secret phrase and re-encrypt the files affected by the secret.

The `sm_rebond` utility shuts down all of the processes run from the suite that were started by using `sm_service` or `sm_serviced`, and that use the same `imk.dat` file. All other processes should be shut down manually before running the utility and restarted once the utility is finished.

The `sm_rebond` 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. The utility then generates an `imk.dat` file and updates all the files containing encrypted information. The `sm_rebond` utility affects all the applications that run on the same host and use the same `imk.dat` file

Note 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 technical support.

Procedure

- 1 Use the `sm_rebond` utility.

You must enter this command with administrative privileges. For the <base directory>, enter the path for the directory in which the software that is to be rebonded is installed.

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

- 2 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.

- 3 Re-enter the new secret to confirm it.
- 4 Close the utility.

Using the secret across large number of sites

You can use the secret phrase across multiple sites.

Prerequisites

Install an Smart Assurance product at a primary site.

Procedure

- 1 Use the `sm_rebond` utility to change the secret phrase.

Note Do not use the `sm_rebond` utility on the other Manager if the files are moved to the Manager before it is started.

- 2 Use the `sm_edit` utility to update the security configuration files and seed files of the suite as needed.

- 3 Copy the `imk.dat` file, as well as the configuration files and seed files to the other installations of the software.

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

Using the secret phrase for multiple products and sites

You can use the secret phrase across multiple products and sites.

Prerequisites

Install one or more Smart Assurance products on one or more hosts.

Procedure

- 1 Stop the Manager if short periods of interrupted communications are not detrimental.
However, if the products also use cleartext connections, you can use the `sm_rebond` utility without stopping the Manager. Only that product will lose connections for a short period of time.
- 2 Use the `sm_rebond` utility to change the secret phrase for each suite on each host.

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

- 3 Use the `sm_edit` utility to update the security configuration files and seed files of the suites as needed.
- 4 Restart the Manager if applicable.

Locking the secret phrase

VMware provides the option to lock the file. If access to encrypted data is a critical issue for your organization, you can opt to lock the 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 VMware Global Services.

Encrypted passwords

The encryption process uses the secret phrase to encrypt passwords contained in the security configuration files. The first line of the `sm_rebond`, `sm_edit`, and `sm_reseed` files indicates the appropriate field to which encryption is applied.

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

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.

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

Changing passwords in security configuration files

The password is the only field in the security configuration file that is encrypted. You can change the password in the security configuration files.

The configuration files are located in the `BASEDIR/smarts/local/conf` directory of a product.

Procedure

- 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.

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
#
```

Note 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.

Note The appropriate values are encrypted as part of the process.

- 5 Repeat the steps for all of the security configuration files.

6 Restart the Manager if applicable.

Note 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>`

Encrypting seed files

The encryption of a seed file applies only to certain information that is applicable to SNMP V3 devices. For these devices, you can encrypt the keyword that contains the authentication password.

Prerequisites

Obtain administrative privileges to encrypt seed files.

Procedure

- 1 The first line of the seed file must begin with a pound (#) sign and read as follows:

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

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

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

Encrypted connections

Software components communicate over TCP connections by using the Remote API. Clients who are using Remote API connections, authenticate themselves to servers by sending credentials, normally 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.

Encrypted connections do not work with the following products:

- Perl API
- Smart Assurance Adapter for NetIQ AppManager

Encrypting connections in FIPS 140 mode

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

The environment variable `SM_FIPS140` controls FIPS 140 status.

Procedure

- ◆ Set the environmental variable `SM_FIPS140` to `TRUE` to enable FIPS 140 mode.

Note 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.

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

Encryption for connections in FIPS 140 mode

Any encryption based on the site secret should only be used once the secret phrase has been changed by using the utility.

The Global Console supports cleartext (Level 0), Diffie Helman-Advanced Encryption Standard (Level 1), and TLSv1.2 encrypted connections. This table lists the encryption level that is provided in FIPS 140 mode.

Table 3-7. Encryption for connections in FIPS mode

Encryption level	Advantages	Disadvantages
TLS v1.2 with FIPS 140 cryptography	<ul style="list-style-type: none"> ■ Standards-based (TLS and FIPS 140 standard) ■ Compatible with FIPS 140-2 compliant cryptography 	<ul style="list-style-type: none"> ■ Incompatible with previous releases of the software ■ Slower than cleartext

Encrypting connections in non-FIPS 140 mode

applications that are based on Foundation 9.1 or later software, and running in non-FIPS 140 mode use TLSv1.2 encrypted connections by default. For applications that are based on pre-9.1 Foundation software, the encrypted connections are configured by using the two environment variables, and .

Note 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.

To set the environment variables so that they can be used by the programs of an Smart Assurance product, edit the `runcmd_env.sh` file, which is located in the `BASEDIR/smarts/local/conf` directory of that product.

Procedure

- 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.

To raise security to the next highest level, change 1 to 2:

```
SM_INCOMING_PROTOCOL=2,1,0
```

```
SM_OUTGOING_PROTOCOL=2,1,0
```

- 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 Manager if applicable.

Encryption levels for connections in non-FIPS 140 mode

VMware recommends the following encryption levels for connections in non-FIPS 140 mode.

Table 3-8. Encryption levels for connections for non-FIPS 140 mode

Security level	Description	Advantages	Disadvantages
0. CLEAR, or CLEARTXT	No encrypted communication	<ul style="list-style-type: none"> ■ Backward compatibility ■ No configuration (default behavior) 	<ul style="list-style-type: none"> ■ No security ■ Passwords passed to servers as clear texts
1	DH-AES	<ul style="list-style-type: none"> ■ No site secret needed ■ No configuration (default behavior for new installations) ■ Protection against eavesdropping 	<ul style="list-style-type: none"> ■ Slower connection than cleartext or level 2 security ■ Not secure against active attacks
2	Encryption based on site secret	<ul style="list-style-type: none"> ■ Protection against eavesdropping and active attack ■ Almost as fast as cleartext 	Must set site secret and keep it common across all communicating entities

Table 3-8. Encryption levels for connections for non-FIPS 140 mode (continued)

Security level	Description	Advantages	Disadvantages
3	DH-AES and site secret	Protection against eavesdropping and active attack, even by those who know the site secret	<ul style="list-style-type: none"> ■ 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	<ul style="list-style-type: none"> ■ Incompatible with previous releases of the software ■ Slower than cleartext

Suggested encrypted connections in non-FIPS 140 mode

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

Table 3-9. Suggested encryption connections in non-FIPS 140 mode

Connections for	Recommended Encryption	Conditions	Additional Information
Broker	<ul style="list-style-type: none"> ■ Cleartext ■ Encryption in <code>SM_INCOMING_PROTOCOL</code> and <code>SM_OUTGOING_PROTOCOL</code> ■ TLS v1.2 	<p>If client supports only cleartext.</p> <p>Note The Broker need not support cleartext if all clients can make encrypted connections.</p>	<p>This is a required configuration since the Broker acts as a client as well as a server.</p> <p>The Broker should be able to communicate with every component in the system.</p>
Domain Manager	Set <code>SM_OUTGOING_PROTOCOL</code> to cleartext as well as encryption	When the Domain Manager must connect to a client that supports only cleartext.	None
Adapters	TLS v1.2	For Adapters based on Foundation 9.1	None
	Set <code>SM_INCOMING_PROTOCOL</code> to encryption	If you have Adapters that accept incoming connections from clients that are not TLS capable.	Adapters that register with the Broker, can accept incoming connections.

Table 3-9. Suggested encryption connections in non-FIPS 140 mode (continued)

Connections for	Recommended Encryption	Conditions	Additional Information
	Add cleartext option to the appropriate variable	If you have Adapters that support only cleartext.	None
Components running on network outside the management domain	<ul style="list-style-type: none"> ■ Set <code>SM_INCOMING_PROTOCOL</code> to encryption ■ Set <code>SM_OUTGOING_PROTOCOL</code> to encryption ■ TLS v1.2 	To configure any components that must run on networks outside the management domain.	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.

Port usage

This topic lists the ports and protocols that are used by components.

Table 3-10. Port usage

Product	Releases	Port	Protocol	Source	Destination	Service Description	Purpose	Classification
VMware Smart Assurance	10.1.7	426	TCP	Smarts Broker	VSA Domain managers	Used to listen to client requests	It is the default port that communicates with the VSA Broker.	Sensitive
VMware Smart Assurance	10.1.7	161	UDP	VSA Application	Device	SNMP Communication	SNMP communication port between NCM and Device	Sensitive
VMware Smart Assurance	10.1.7	162	UDP	Device	VSA Application	Used for receiving SNMP Traps.	For SNMP Traps.	Sensitive
VMware Smart Assurance	10.1.7	22	TCP	VSA Application	Device	SSH communication	For SSH communication.	Sensitive
VMware Smart Assurance	10.1.7	23	TCP	VSA Application	Device	Telnet communication	For Telnet communication	Sensitive
VMware Smart Assurance	10.1.7	8080	TCP	VSA Application (Tomcat)	VSA client Application	Tomcat service	Used for accessing VSA application by the clients.	Sensitive
VMware Smart Assurance	10.1.7	58080	TCP	VSA Application (Tomcat)	EMC M&R	M&R service.	Used for communication between M&R and VSA application.	Sensitive

Table 3-10. Port usage (continued)

Product	Releases	Port	Protocol	Source	Destination	Service Description	Purpose	Classification
VMware Smart Assurance	10.1.7	9200	TCP	VSA Application (Elastic Search)	EDAA ingestion	Elastic search communication	Used for interacting with EDAA ingestion	Sensitive
VMware Smart Assurance	10.1.7	15672	TCP	VSA Application (RabbitMQ)	SAM notification publisher	RabbitMQ service	Used for SAM notification publisher	Sensitive
VMware Smart Assurance	10.1.7	8443	TCP	VSA Application (Tomcat)	EDA API	Secured Tomcat service	Used for accessing VSA application by the clients (SSL enabled).	Sensitive and SSL enabled
VMware Smart Assurance	10.1.7	58443	TCP	EMC M&R (Watch4net) Tomcat server	EMC M&R	M&R service.	Used for communication between M&R and VSA application (SSL enabled).	Sensitive and SSL enabled
VMware Smart Assurance	10.1.7	9091,9092,9093	TCP	Kafka broker	Domain managers	Kafka service	VSA application communication with Kafka process	Sensitive
VMware Smart Assurance	10.1.7	8443	TCP	VSA application	DCF	DCF service	Communication between DCF and VSA application	Sensitive
VMware Smart Assurance	10.1.7	514	UDP	Device	VSA Application	VSA syslog adapter service	For Syslog messages	-

Disabling the RC4 algorithm with a cipher suite list

You can specify a list of cipher suites as an alternative to the RC4 algorithm used for TLS communication. The RC4 algorithm is a weaker cipher and vulnerable to attacks. If you want to disable the RC4 algorithm from Smart Assurance, you can use a cipher suite list.

A cipher suite is a suite of cryptographic algorithms used to provide encryption, integrity and authentication. Cipher suite lists and the `SM_TLS_SUITE_LIST` environment variable are described in [Communication protocols overview](#). Security Advisory “ESA-2016-115” provides more information about the fixed vulnerabilities for the RC4 algorithm.

Introduced with the 9.4.2 release, this feature is supported for the following Smart Assurance products: SAM, IP Manager, ESM, MPLS, NPM, OTM, VoIP AM, and the SAM Global Console. It is not supported for EMC M&R. If your deployment includes NCM, consult the *VMware Smart Assurance Network Configuration Manager Security Configuration Guide* for information about using ciphers.

To disable the RC4 algorithm and specify a cipher suite list, follow this procedure.

Procedure

- 1 For each Manager and SAM Global Console, add the `SM_TLS_SUITE_LIST` environment variable to the `runcmd_env.sh` file.

- a Go to the `<BASEDIR>/smarts/bin` directory and enter this command to open the `runcmd_env.sh` file:

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

- b Add the `SM_TLS_SUITE_LIST` variable and specify one or more cipher suites. Use a colon (`:`) to separate multiple cipher suites. For example:

```
SM_TLS_SUITE_LIST=AES256-GCM-SHA-384
```

In this example, two cipher suites are listed:

```
SM_TLS_SUITE_LIST=AES256-GCM-SHA-384:AES128-GCM-SHA256
```

- c Save and close the file.
 - d Restart the Manager.
- 2 For each SAM Global Console, perform these steps to allow the console to communicate with the Broker:
 - a Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8 from the Oracle website.
 - b Extract the `local_policy.jar` and `US_export_policy.jar` files from the downloaded zip file into a temporary directory.
 - c Go to the `<BASEDIR>/smarts/jre/lib/security` directory and then back up the existing policy files in this directory.
 - d Overwrite the `local_policy.jar` and `US_export_policy.jar` files in the `<BASEDIR>/smarts/jre/lib/security` directory.
 - e Restart the SAM Console Tomcat server and Global Consoles (`sm_gui` applications) for the changes to take effect.

Data Security Settings

Data security settings enable definition of controls to prevent data permanently stored by the product to be disclosed in an unauthorized manner.

Encryption of data at rest

All data and passwords are encrypted and stored in the repository (.rps) files.

Additional Security Considerations

You can consider the following additional security measures to make your deployment secure:

- [Importing TLS certificates](#)
- [Pseudo-random number generators \(PRNG\) algorithm](#)
- [Configuring a secure Broker](#)
- [Setting the SM_AUTHORITY variable](#)
- [Specifying alternate security configuration files](#)
- [Options to run as non-root on Linux systems](#)

Preventing Elasticsearch from being vulnerable to dynamic scripting

If you configure Elasticsearch for access from a remote host, arbitrary code can execute. You can prevent this vulnerability.

The default configuration in the version of Elasticsearch used in this release enables dynamic scripting, which potentially allows for execution of arbitrary code. While we ship Elasticsearch configured for access from the local host only, it is best to disable the dynamic scripting feature, which this product does not use: add the line `script.disable_dynamic: true` to the `conf/elasticsearch/elasticsearch.yml` file and restart the elastic-search service. To restart the service on Linux, use the command `./sm_service start smarts-elasticsearch`.

Importing TLS certificates

By default, an generates a self-signed TLS certificate that is presented to the incoming TLS connections. However, you can configure a certificate to use by performing the following step:

Procedure

- 1 Place the certificate and key files in `$(SM_SITEMOD)/conf/tls/$(DOMAIN).cert` and `$(SM_SITEMOD)/conf/tls/$(DOMAIN).key`

Here, `$(DOMAIN)` is the name of the domain used to register with the Broker.

Note The files must be in PKCS#8, PEM encoded form with no password.

The self-signed certificates generated by Smart Assurance are stored in the process memory.

- 2 To regenerate the default self-signed TLS certificates, stop and restart the Smart Assurance processes.

Pseudo-random number generators (PRNG) algorithm

Global Console SSL uses (HMACDRBG256) as the default PRNG algorithm.

Configuring a secure Broker

You can configure the to run in a secure manner.

The use of a secure Broker results in the following changes:

- The consoles prompt for a username and password to connect to the Broker. Without a secure Broker, consoles connect to the Broker without authenticating.
- The 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`.

Procedure

- 1 Choose a unique Smart Assurance 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.

You could use the username `SecureBroker` and the password `Secure`. Choose a unique Smart Assurance 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 Smart Assurance 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.

```
#*:<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. In this example, it involves replacing the password `Secure` with `<PROMPT>`.

```
*: <BROKER> : SecureBroker : <PROMPT>
```

- 3 Use `sm_edit` to make the following changes to the local `serverConnect.conf` file used by the Broker:

- a Delete the line granting `<DEFAULT>/<DEFAULT>` access to the Broker.
- b Change the `BrokerNonsecure/Nonsecure` line to grant `Ping` access rather than `All` access. Do not, however, delete this authentication record.
- c Add a new authentication record that grants `All` access to the `SecureBroker/Secure` credentials. This new record must be below the `BrokerNonsecure/Nonsecure` record.

```
<BROKER>:BrokerNonsecure:Nonsecure:Ping
```

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

Setting the SM_AUTHORITY variable

The environment variable controls the authentication security features provided by the software. You can add and set the variable in the file.

To add and set the `SM_AUTHORITY` environment variable:

Procedure

- 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.

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

4 Restart the Smart Assurance programs that were previously launched from this installation directory for the environment variable to take effect.

Specifying alternate security configuration files

You can define separate files on hosts where multiple servers or clients are running. The environment variables that allows you to specify distinct configuration files are `SM_SERVERCONNECT` and `SM_CLIENTCONNECT`.

By defining separate files on hosts, you can 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 the servers and/or clients operate differently.

The following steps set the value of the `SM_SERVERCONNECT` environment variable to point to the `server_Connect_IP.conf` file.

Prerequisites

Specify the absolute path to the alternate security file for each variable. Because you specify the filename, you can have multiple files in the same directory. For server programs, specify an alternate `serverConnect.conf` file, by setting the `SM_SERVERCONNECT` variable in the command line that starts the server.

Procedure

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

Note You must have administrative privileges when typing the following command.

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

```
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.

Note 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.

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

Options to run as non-root on Linux systems

This topic lists the options that are used in conjunction with each other to run as non-root user. These two options are only available on Linux systems. These options are available for the `and` commands.

Options used to run as not-root user

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.

Table 3-11. Additional options for running as non-root on Linux systems

Options	Description
<pre>--privopen=<arg>[, <arg>]</pre>	<p>This option is only used in conjunction with the <code>--run-as-user</code> option to run as non root.</p> <p>Opens privileged sockets. The <code><arg></code> parameter has the following syntax:</p> <pre><type>[:<family>]:<port>[, #<count>]</pre> <p>where:</p> <ul style="list-style-type: none"> ■ <code><type></code> is one of the following: <ul style="list-style-type: none"> ■ TCP (for a TCP connection) ■ UDP (for a datagram) ■ IP (for a raw socket) ■ <code><family></code> is one of the following: <ul style="list-style-type: none"> ■ :v4 (IPv4 address family) ■ :v6 (IPv6 address family) <p>If <code><family></code> is not specified, the address family defaults to IPv4.</p> <ul style="list-style-type: none"> ■ <code><port></code> is one of the following: <ul style="list-style-type: none"> ■ The required privileged port for a TCP socket ■ The required privileged port for a UDP socket ■ The protocol for IP ■ <code><count></code> is the number of sockets of the type, family and port. The default count is 1.
<pre>--run-as-user=<username></pre>	<p>Note This option is only used in conjunction with the <code>--privopen</code> option to run as non-root user.</p> <p>Specifies a valid user name.</p>

Example: Opening multiple ports by repeating the `<arg>` parameter. Each instance is separated by a comma (,).

```
--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.”

Security Maintenance

4

This chapter includes the following topics:

Read the following topics next:

- [Routine password maintenance](#)
- [Change the secret phrase](#)

Routine password maintenance

To maintain your security, change the passwords to access your servers and s regularly by using the utility.

Change the secret phrase

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

Smart Assurance programs use the site secret to encrypt:

- Passwords in the configuration files.
- Passwords for SNMP v3 devices in seed files.
- Connections between programs.

Physical Security Controls

5

This chapter includes the following topics:

Read the following topics next:

- [Physical Security at customer site](#)

Physical Security at customer site

Physical security controls enable the protection of resources against unauthorized physical access and physical tampering.

Since the software is installed on existing customer hardware, VMware is neither responsible nor liable for establishing norms and operations to secure the physical storage environment at customer sites.