AppManager for SNMP Traps

Management Guide

December 2018

Legal Notice

Contents

About this Book and the Library

The NetIQ AppManager product (AppManager) is a comprehensive solution for managing, diagnosing, and analyzing performance, availability, and health for a broad spectrum of operating environments, applications, services, and server hardware.

AppManager provides system administrators with a central, easy-to-use console to view critical server and application resources across the enterprise. With AppManager, administrative staff can monitor computer and application resources, check for potential problems, initiate responsive actions, automate routine tasks, and gather performance data for real-time and historical reporting and analysis.

Intended Audience

This guide provides information for individuals responsible for installing an AppManager module and monitoring specific applications with AppManager.

Other Information in the Library

The library provides the following information resources:

Installation Guide for AppManager

Provides complete information about AppManager pre-installation requirements and step-by-step installation procedures for all AppManager components.

User Guide for AppManager Control Center

Provides complete information about managing groups of computers, including running jobs, responding to events, creating reports, and working with Control Center. A separate guide is available for the AppManager Operator Console.

Administrator Guide for AppManager

Provides information about maintaining an AppManager management site, managing security, using scripts to handle AppManager tasks, and leveraging advanced configuration options.

Upgrade and Migration Guide for AppManager

Provides complete information about how to upgrade from a previous version of AppManager.

Management guides

Provide information about installing and monitoring specific applications with AppManager.

Help

Provides context-sensitive information and step-by-step guidance for common tasks, as well as definitions for each field on each window.

The AppManager library is available in Adobe Acrobat (PDF) format from the AppManager Documentation page of the NetIQ Web site.

About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

Enabling critical business services, better and faster

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

Our Philosophy

Selling intelligent solutions, not just software

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate — day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

Driving your success is our passion

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with — for a change. Ultimately, when you succeed, we all succeed.

Our Solutions

- Identity & Access Governance
- Access Management
- Security Management
- Systems & Application Management
- Workload Management
- Service Management

Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

Worldwide:	www.netiq.com/about_netiq/officelocations.asp
United States and Canada:	1-888-323-6768
Email:	info@netiq.com
Web Site:	www.netiq.com

Contacting Technical Support

For specific product issues, contact our Technical Support team.

Worldwide:	www.netiq.com/support/contactinfo.asp
North and South America:	1-713-418-5555
Europe, Middle East, and Africa:	+353 (0) 91-782 677
Email:	support@netiq.com
Web Site:	www.netiq.com/support

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. If you have suggestions for improvements, click **comment on this topic** at the bottom of any page in the HTML versions of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

NetIQ Communities, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, NetIQ Communities helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit http://community.netiq.com.

1 Introducing AppManager for SNMP Traps

This chapter introduces AppManager for SNMP Traps, providing an overview of the module and describing how you can use AppManager to monitor SNMP traps sent by remote devices to a NetIQ Trap Receiver server. This module also enables you to improve the AppManager event messaging generated by trap source objects.

1.1 Features and Benefits

The following list describes the features and benefits of monitoring SNMP traps with AppManager:

- A new Discovery_SNMPTraps Knowledge Script, which you can use to discover known devices that forward SNMP traps to a NetIQ Trap Receiver server. This script discovers devices that generate traps using SNMP version 1, 2, or 3. For more information, see “Discovering SNMP Traps Resources” on page 19.
- A new SNMPTraps_AddMIB Knowledge Script, which you can use to:
  - Add management information base (MIB) files that you can use to convert trap object identifiers (OIDs) into object descriptive names (ODEs) to make monitored traps more readable with the SNMPTraps_TrapMonitor Knowledge Script.
  - Reload all MIB files in the MIB directory.
- A new SNMPTraps_TrapMonitor Knowledge Script, which you can use to:
  - Monitor SNMP v1, v2, and v3 traps sent by remote devices in real time.
  - Display trap source devices in an AppManager console on demand.
  - Filter the list of devices monitored, with inclusion filters based on OID, ODE, and MIB subtree, and exclusion filters based OID, ODE, varbind value, and MIB subtree.
  - Enable custom formatting for AppManager events that correspond to SNMP traps in the `SNMPTraps_AlarmMappings.csv` file that comes with this module.
  - Monitor NetIQ Trap Receiver availability.
  - Monitor traps from unknown devices.
- Improved and more readable event detail messaging.
- Ability to customize short message event format using the `SNMPTraps_AlarmMappings.csv` file.

1.2 Devices with Custom Event Formatting in this Module

While the AppManager for SNMP Traps module can potentially support any trap forwarded by any SNMP device, the module also provides custom event messaging and severities for traps forwarded by the following SNMP devices:

- Acme Packet Net-Net; for a list of relevant MIBs, see “MIB Files for Acme Packet Net-Net Devices” on page 10.

- ApplianX Protocol Converter
- Audiocodes Media Gateway
- Avaya Application Enablement Services (AES)
- Avaya Call Management System (CMS)
- Avaya CallPilot
- Avaya Definity Audix
- Avaya Intuity Audix
- Avaya Firewall
- Avaya Media Application Server
- Avaya Meeting Exchange
- Avaya Message Network Server
- Avaya Modular Messaging
- Avaya one-X Client Enablement Services
- Avaya one-X Portal
- Avaya Communication Server 1000
- Avaya Session Border Controller (SBC) (Sipera)
- Avaya Session Manager
- Avaya System Manager
- Avaya Telephony Manager 3
- Avaya Voice Portal
- Avaya Web Conferencing
- Check Point Firewall
- Cisco Adaptive Security Appliance (ASA) 5510
- Cisco Catalyst devices; for a list of relevant MIBs, see “MIB Files for Cisco Catalyst Devices” on page 11.
- Exinda WAN Optimizers
- Extreme LAN Switch
- NICE Recorder
- Radware Load Balancer
- Sophos Management Console

You can also add custom event messaging and severities for traps forwarded from additional devices by editing the `SNMPTraps_AlarmMappings.csv` file that comes with this module. For more information, see Section 3.3, “Customizing AppManager Events for Trap Source Devices,” on page 40.

MIB Files for Acme Packet Net-Net Devices

This module supports traps forwarded by Acme Packet Net-Net devices defined in the following MIB files:

- APSYSMGMT-MIB
- APLICENSE-MIB
- APSWINVENTORY-MIB

- ACMEPACKET-ENVMON-MIB
- APSYSLOG-MIB
- APEMS-MIB
- APAGENTCAP-MIB
- APSECURITY-MIB

MIB Files for Cisco Catalyst Devices

This module also supports traps forwarded by Cisco Catalyst devices defined in the following MIB files:

- BGP4-MIB
- BRIDGE-MIB
- CISCO-AUTH-FRAMEWORK-MIB
- CISCO-BGP4-MIB
- CISCO-BULK-FILE-MIB
- CISCO-CLUSTER-MIB
- CISCO-CONFIG-COPY-MIB
- CISCO-CONFIG-MAN-MIB
- CISCODHCP-SNOOPING-MIB
- CISCO-ENERGYWISE-MIB
- CISCO-ENTITY-SENSOR-MIB
- CISCO-ENVMON-MIB
- CISCO-ERR-DISABLE-MIB
- CISCO-FLASH-MIB
- CISCO-HSRP-MIB
- CISCO-IF-EXTENSION-MIB
- CISCO-IPMROUTE-MIB
- CISCO-MAC-NOTIFICATION-MIB
- CISCO-OSPF-TRAP-MIB
- CISCO-PAE-MIB
- CISCO-PIM-MIB
- CISCO-PING-MIB
- CISCO-PORT-SECURITY-MIB
- CISCO-PORT-STORM-CONTROL-MIB
- CISCO-POWER-ETHERNET-EXT-MIB
- CISCO-PROCESS-MIB
- CISCO-RESILIENT-ETHERNET-PROTOCOL-MIB
- CISCO-RTTMON-MIB
- CISCO-STACK-MIB
- CISCO-STP-EXTENSIONS-MIB
- CISCO-SYSLOG-MIB

- CISCO-UDLDP-MIB
- CISCO-VLAN-MEMBERSHIP-MIB
- CISCO-VTP-MIB
- ENTITY-MIB (RFC2737)
- IF-MIB (RFC1573)
- LLDP-MIB
- OSPF-TRAP-MIB
- PIM-MIB
- POWER-ETHERNET-MIB
- RMON-MIB
- SNMPv2-MIB

2 Installing AppManager for SNMP Traps

This chapter provides installation instructions and describes system requirements for AppManager for SNMP Traps.

This chapter assumes you have AppManager installed. For more information about installing AppManager or about AppManager system requirements, see the *Installation Guide for AppManager*, which is available on the AppManager Documentation page.

2.1 System Requirements

For the latest information about supported software versions and the availability of module updates, visit the AppManager Supported Products page. Unless noted otherwise, this module supports all updates, hotfixes, and service packs for the releases listed below.

AppManager for SNMP Traps has the following system requirements:

Software/Hardware	Version
NetIQ AppManager installed on the AppManager repository (QDB) computers, on the proxy agent computers you want to monitor (agents), and on all console computers	8.0.3, 8.2, 9.1, 9.2, 9.5, or later One of the following AppManager agents are required: ◆ AppManager agent 7.0.4 with hotfix 72616 or later ◆ AppManager agent 8.0.3, 8.2, 9.1, 9.2, 9.5, or later
Microsoft Windows operating system on the agent computers	One of the following: ◆ Windows Server 2016 ◆ Windows 10 (32-bit or 64-bit) ◆ Windows Server 2012 R2 ◆ Windows Server 2012 ◆ Windows 8 (32-bit or 64-bit) ◆ Windows Server 2008 R2 ◆ Windows Server 2008 (32-bit or 64-bit) ◆ Windows 7 (32-bit or 64-bit)
Microsoft .NET Framework on the proxy agent computers	3.5 or later
Microsoft SQL Server Native Client 11.0 **(for TLS 1.2 support)**	11.3.6538.0 or later **NOTE:** The SQL Server Native client can be installed from this Microsoft download link.

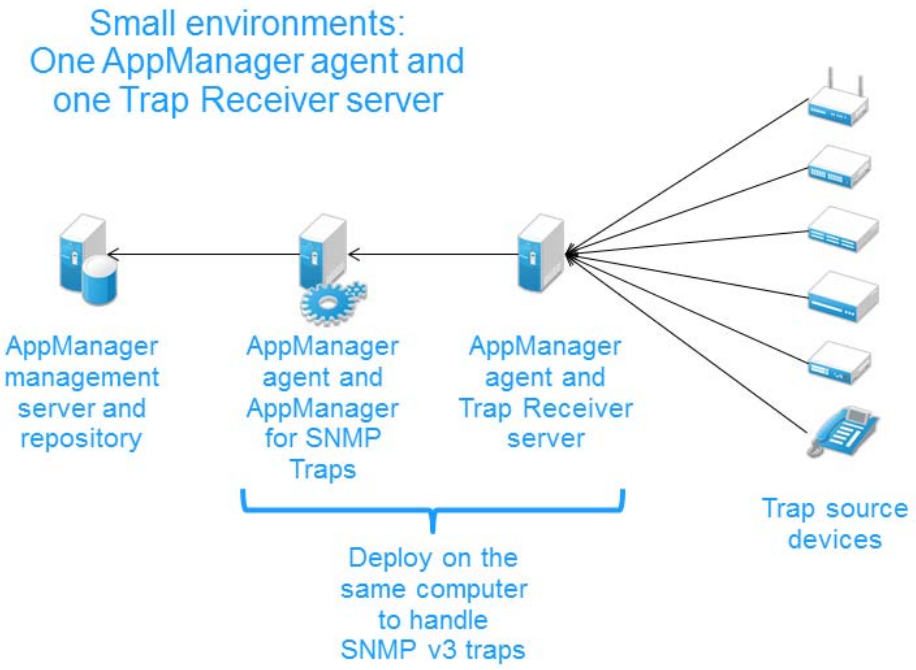
NOTE: If you want TLS 1.2 support and are running AppManager 9.1 or 9.2, then you are required to perform some additional steps. To know about the steps, see the article.

2.2 Pre-installation Considerations

On the AppManager agent computer, AppManager for SNMP Traps installs the module and the NetIQ Trap Receiver server. The installation includes a service called *NetIQ Trap Receiver* that is not started by default; you must start this service to enable the Trap Receiver server on the AppManager agent.

Small Environments

If you have a small number of trap source devices you plan on monitoring, the most basic deployment consists of one AppManager agent and one Trap Receiver server. All trap source devices forward traps to the single Trap Receiver server.



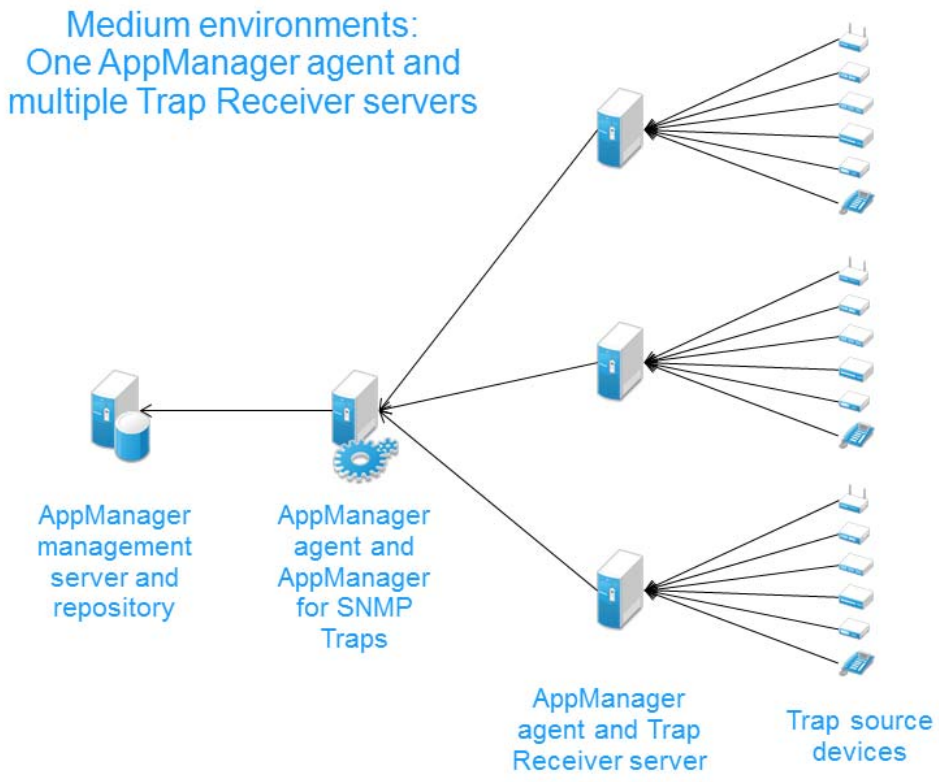
If you plan to monitor SNMP v3 traps, install the Trap Receiver server and the AppManager agent on the *same* computer to prevent malicious users from gaining secure access to the information in these traps.

By default, the Trap Receiver server and the AppManager agent are installed on the same computer.

Medium-sized Environments

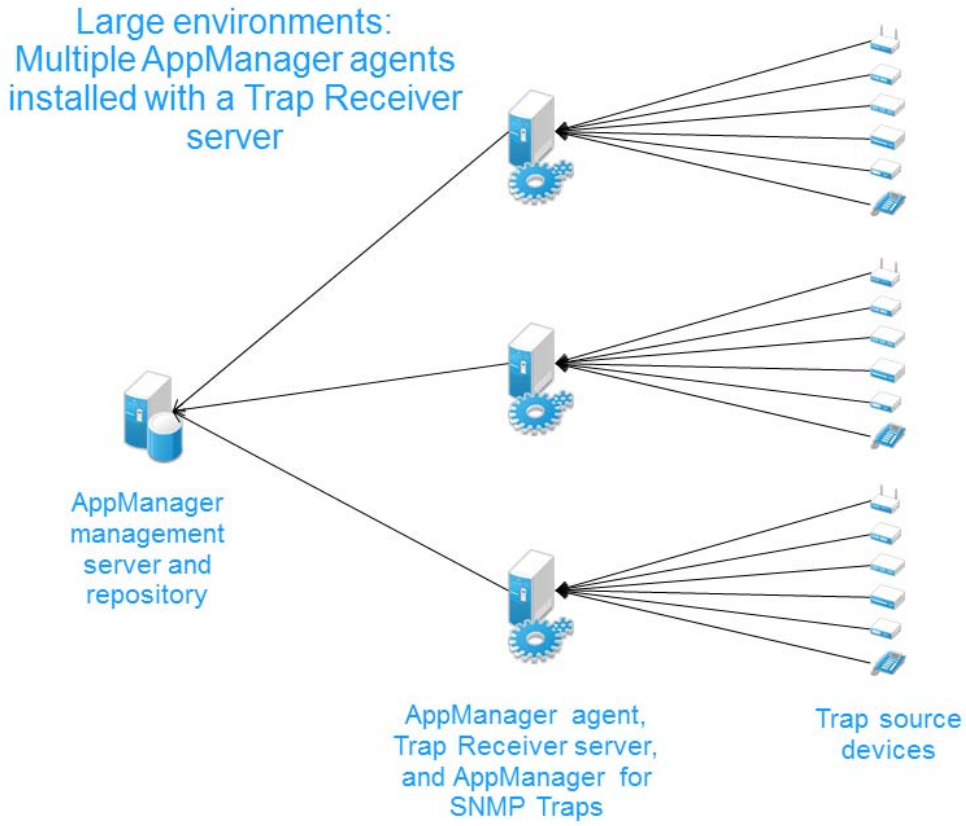
For medium-sized environments, you can load-balance the trap reception by installing the module and the Trap Receiver server on more than one AppManager agent. Launch the Trap Receiver server by starting the NetIQ Trap Receiver service on multiple AppManager agents after the installation completes.

The following configuration uses just one AppManager agent to process all of the traps received by multiple Trap Receiver servers.



Large Environments

For large environments, install and run the module and the Trap Receiver server together on more than one AppManager agents. This configuration load-balances the AppManager agents and the Trap Receiver servers so you can process more traps in a single AppManager environment, and it prevents a single agent from having to process all the traps if you have many trap source devices.



2.3 Installing the Module

To install the AppManager agent components, run the module installer on the AppManager agents that will monitor traps sent by remote devices. To install the Help and console extensions, run the module installer on all console computers.

Access the `AM70-SNMPTraps-8.x.x.0.msi` module installer from the `AM_SNMPTraps_8.x` self-extracting installation package on the AppManager Module Upgrades & Trials page.

For Windows environments where User Account Control (UAC) is enabled, install the module using an account with administrative privileges. Use one of the following methods:

- Log in to the server using the account named Administrator. Then, run the module installer `.msi` file from a command prompt or by double-clicking it.
- Log in to the server as a user with administrative privileges and run the module installer `.msi` file as an administrator from a command prompt. To open a command-prompt window at the administrative level, right-click a command-prompt icon or a Windows menu item and select **Run as administrator**.

You can install the Knowledge Scripts into local or remote AppManager repositories (QDBs). The module installer installs Knowledge Scripts for each module directly into the QDB instead of installing the scripts in the `\AppManager\qdb\kp` folder as in previous releases of AppManager.

You can install the module manually, or you can use Control Center to deploy the module to a remote computer where an agent is installed. For more information, see Section 2.4, “Deploying the Module with Control Center,” on page 17. However, if you use Control Center to deploy the module, Control Center only installs the *agent* components of the module. The module installer installs the QDB and console components as well as the agent components on the agent computer.

To install the module manually:

1. Double-click the module installer `.msi` file.
2. Accept the license agreement.
3. Review the results of the pre-installation check. You can expect one of the following three scenarios:
   - **No AppManager agent is present:** In this scenario, the pre-installation check fails, and the installer does not install agent components.
   - **An AppManager agent is present, but some other prerequisite fails:** In this scenario, the default is to not install agent components because of one or more missing prerequisites. However, you can override the default by selecting Install agent component locally. A missing application server for this particular module often causes this scenario. For example, installing the AppManager for Microsoft SharePoint module requires the presence of a Microsoft SharePoint server on the selected computer.
   - **All prerequisites are met:** In this scenario, the installer installs the agent components.
4. To install the Knowledge Scripts into the QDB:
   - **4a** Select **Install Knowledge Scripts** to install the repository components, including the Knowledge Scripts, object types, and SQL stored procedures.
   - **4b** Specify the SQL Server name of the server hosting the QDB, as well as the case-sensitive QDB name.
5. (Conditional) If you use Control Center 7.x, run the module installer for each QDB attached to Control Center.
6. (Conditional) If you use Control Center 8.x or later, run the module installer only for the primary QDB. Control Center automatically replicates this module to secondary QDBs.
7. Run the module installer on all console computers to install the Help and console extensions.
8. (Conditional) If you are monitoring SNMP trap source devices that require the use of SNMP version 3, run the Discovery_SNMPTraps Knowledge Script on all agent computers from which you plan to monitor those source devices. For more information, see Section 2.6, “Discovering SNMP Traps Resources,” on page 19.

After the installation has completed, the `SNMPTraps_Install.log` file, located in the `\NetIQ\Temp\NetIQ_Debug\`*`ServerName`* folder, lists any problems that occurred.

2.4 Deploying the Module with Control Center

You can use Control Center to deploy the module to a remote computer where an agent is installed. This topic briefly describes the steps involved in deploying a module and provides instructions for checking in the module installation package. For more information, see the *Control Center User Guide for AppManager*, which is available on the AppManager Documentation page.

Deployment Overview

This section describes the tasks required to deploy the module on an agent computer.

To deploy the module on an agent computer:

1. Verify the default deployment credentials.
2. Check in an installation package. For more information, see “Checking In the Installation Package” on page 18.
3. Configure an e-mail address to receive notification of a deployment.
4. Create a deployment rule or modify an out-of-the-box deployment rule.
5. Approve the deployment task.
6. View the results.

Checking In the Installation Package

You must check in the installation package, `AM70-SNMPTraps-8.x.x.xml`, before you can deploy the module on an agent computer.

To check in a module installation package:

Log on to Control Center using an account that is a member of a user group with deployment permissions.

1. Navigate to the **Deployment** tab (for AppManager 8.x or later) or **Administration** tab (for AppManager 7.x).
2. In the Deployment folder, select **Packages**.
3. On the Tasks pane, click **Check in Deployment Packages** (for AppManager 8.x or later) or **Check in Packages** (for AppManager 7.x).
4. Navigate to the folder where you saved `AM70-SNMPTraps-8.x.x.xml` and select the file.
5. Click **Open**. The Deployment Package Check in Status dialog box displays the status of the package check in.

2.5 Silently Installing the Module

To silently (without user intervention) install a module using the default settings, run the following command from the folder in which you saved the module installer:

```
msiexec.exe /i "AM70-SNMPTraps-8.x.x.0.msi" /qn
```

where *x.x* is the actual version number of the module installer.

To create a log file that describes the operations of the module installer, add the following flag to the command noted above:

```
/L* "AM70-SNMPTraps-8.x.x.0.msi.log"
```

The log file is created in the folder in which you saved the module installer.

NOTE: To perform a silent install on an AppManager agent running Windows Server 2008 R2 or Windows Server 2012, open a command prompt at the administrative level and select **Run as administrator** before you run the silent install command listed above.

To silently install the module to a remote AppManager repository, you can use Windows authentication or SQL authentication.

Windows authentication:

```
AM70-SNMPTraps-8.x.x.0.msi /qn MO_B_QDBINSTALL=1 MO_B_MOINSTALL=0
MO_B_SQLSVR_WINAUTH=1 MO_SQLSVR_NAME=SQLServerName MO_QDBNAME=AM-RepositoryName
```

SQL authentication:

```
AM70-SNMPTraps-8.x.x.0.msi /qn MO_B_QDBINSTALL=1 MO_B_MOINSTALL=0
MO_B_SQLSVR_WINAUTH=0 MO_SQLSVR_USER=SQLLogin MO_SQLSVR_PWD=SQLLoginPassword
MO_SQLSVR_NAME=SQLServerName MO_QDBNAME=AM-RepositoryName
```

2.6 Discovering SNMP Traps Resources

Use the Discovery_SNMPTraps Knowledge Script to discover known devices that forward SNMP traps to a NetIQ Trap Receiver server. You can discover devices that generate traps that use SNMP version 1, 2, or 3.

This script creates trap source device objects in the Navigation pane or TreeView for devices that can be polled with SNMP as well as devices that cannot be polled with SNMP. The display name format of all trap source device objects created by this script use the following format:

```
Trap Source: Device Name [Device IP Address]
```

For example:

```
Trap Source: MyRouter [10.22.120.67]
```

You can specify one or more sets of mappings that pair a device name to an IP address, which enables you to customize how the list of discovered SNMP Traps device objects display in the Navigation pane or TreeView.

By default, this script runs once. You can run each iteration of a Discovery_SNMPTraps job on just one NetIQ Trap Receiver server. If you have multiple trap receiver servers, run one Discovery_SNMPTraps job for each Trap Receiver server.

If you delete or add a resource object, or if you make any other kind of change that might affect the monitoring of your devices, run the Discovery_SNMPTraps Knowledge Script again to update your list of resource objects.

Prerequisite

Before running the Discovery_SNMPTraps script, configure AppManager Security Manager with the community string and version information for each device you want to monitor. Security Manager entries for SNMP v1 and v2 are optional, but SNMP v3 traps require a Security Manager entry.

If you already use other modules that monitor SNMP traps, such as AppManager for Avaya Communication Manager or AppManager for Network Devices, you can continue to use any existing SNMPTrap Security Manager entries.

The type of Security Manager information you configure varies according to the version of SNMP implemented on the device. AppManager for SNMP supports SNMP versions 1, 2, and 3.

Configuration for SNMP Versions 1 and 2

To set up Security Manager for SNMP v1 or SNMP v2 traps, complete the following fields on the **Custom** tab in Security Manager:

Field	Description
Label	`SNMPTraps` This script also supports Security Manager entries labeled `SNMPTrap`, which is a label used by other modules that you might have already installed, such as AppManager for Avaya Communication Manager or AppManager for Network Device,
Sub-label	Specify whether the community string is used for a single device or for all devices: ◆ For a single device, list the IP address for the community string. ◆ For all devices, enter `default`.
Value 1	Specify the community string for the device or devices.
Value 2	Leave this field blank.
Value 3	Leave this field blank.

Configuration for SNMP Version 3

AppManager for SNMP supports the following modes for SNMP version 3 (SNMP v3):

- No authentication; no privacy
- Authentication; no privacy
- Authentication and privacy

In addition, the module supports the following protocols for SNMP v3:

- MD5 (Message-Digest algorithm 5, an authentication protocol)
- SHA (Secure Hash Algorithm, an authentication protocol)
- DES (Data Encryption Standard, an encryption protocol)
- AES (Advanced Encryption Standard, an encryption protocol, 128-bit keys only)

Configure SNMP v3 information for each device monitored by each proxy computer.

If you plan to monitor SNMP v3 traps, install the NetIQ Trap Receiver and the AppManager agent on the *same* computer to prevent malicious users from gaining secure access to the information in these traps. The Discovery_SNMPTraps script notifies you if an SNMP v3 trap source device's corresponding NetIQ Trap Receiver IP address does not match the IP address of the AppManager agent monitoring it.

The Discovery_SNMPTraps script does not fully validate SNMP v3 credentials retrieved from Security Manager for a particular device or set of devices, and the script does not notify you if these credentials do not match. As a result, the Discovery_SNMPTraps script might miss some SNMP v3 traps if you do not enter the Security Manager credentials properly.

For SNMP v3 configuration, complete the following fields in the **Custom** tab of Security Manager for the proxy agent computer.

Field	Description
Label	`SNMPTraps` This script also supports Security Manager entries labeled `SNMPTrap`, which is a label used by other modules that you might have already installed, such as AppManager for Avaya Communication Manager or AppManager for Network Devices.
Sub-label	Specify the IP address, or enter `default` for all devices that do not have a specific IP address entry.
Value 1	Specify the SNMP user name, or *entity*, configured for the device. All SNMP v3 modes require an entry in this field.
Value 2	Specify the name of the context associated with the user name or entity entered in **Value 1**. A *context* is a collection of SNMP information that is accessible by an entity. If possible, enter a context that provides access to all MIBS for a device. If the device does not support context, type an asterisk (*). All SNMP v3 modes require an entry in this field.
Value 3	Specify the combination of protocol and password appropriate for the SNMP v3 mode you have implemented. ◆ For *no authentication/no privacy mode*, leave this field blank. ◆ For *authentication/no privacy mode*, enter `md5` or `sha` and the password for the protocol, separating each entry with a comma. For example, enter `md5,abcdef` ◆ For *authentication/privacy mode*, enter `md5` or `sha` and the associated password, and then enter `des` and the associated password, separating each entry with a comma. For example, enter `sha,hijklm,des,nopqrs`

Setting Parameter Values

Set the **Values** tab parameters as needed.

Description	How to Set It
General Settings	

Description	How to Set It
Job Failure Notification	
Event severity if discovery job fails unexpectedly	Set the event severity level, from 1 to 40, to reflect the importance when this script fails unexpectedly. The default is 5.
Event Details	
Event detail format	Select whether to view event details in an HTML table or in plain text. The default is HTML Table.
Additional Settings	
Tracing (for advanced users only)	**Note** Use the tracing settings in this section only with the help of NetIQ Technical Support.
Raise event with job execution log?	Select **Yes** to raise an event when the job execution log is created. The default is unselected.
Logging level	Select the logging level you want to monitor. The options are Off, Fatal, Error, Warn, Info, Debug, or All. Use these settings only with the help of Technical Support. The default is Warn.
Derive event severity from most severe event log entry?	Select **Yes** to calculate the event severity for the *Raise event with job execution log* parameter based on the most severe event log entry. The default is Yes.
Event severity (if automatic severity computation not selected above)	If you did not select **Yes** for the *Derive event severity from most severe event log entry* parameter, set the event severity level, from 1 to 40, to reflect the importance of the event raised with the creation of the job execution log.The default is 40.
Discover SNMP Trap Devices	
Raise event if discovery succeeds?	Select **Yes** to raise an event when this script successfully discovers devices that forward traps to trap receivers. The default is Yes.
Event severity when discovery succeeds	Set the event severity level, from 1 to 40, to reflect the importance when this script successfully discovers devices that forward traps to trap receivers. The default is 25.
Raise event if discovery fails?	Select **Yes** to raise an event when this script fails to discover devices that forward traps to trap receivers. The default is Yes.
Event severity when discovery fails	Set the event severity level, from 1 to 40, to reflect the importance when the script fails to discover devices that forward traps to trap receivers. The default is 5.
Update the TreeView object name if the device name changed since the previous discovery?	Select **Yes** if you changed the name of a device after initially discovering it, and you want to update the name of the Navigation pane or TreeView object with the new name. The renamed device should have the same IP address, and after the script updates the Navigation pane or TreeView object with the new name, the script monitors the new object and stops monitoring the old object. The default is unselected. If you select No for this parameter, and you change the name of a device after initially discovering it, and then you run discovery again on the device, the script will not create a new Navigation pane or TreeView object. The new name of the device does not display in the Navigation pane or TreeView.

Description	How to Set It
Name of the device to populate in the TreeView	Specify the name of the device that forwards traps to a trap receiver. Use this parameter and the *IP address of the device to populate in the Treeview* parameter if you only want to discover one device. If you want to discover multiple devices, use the *File containing a list of device name/IP address pairs to populate in the TreeView* parameter. This parameter only supports characters allowed in a hostname or fully qualified domain name (FQDN).
IP address of the device to populate in the TreeView	Specify the IP address for the device you want to monitor. This script does not support the discovery of devices that use IPv6 addresses.

File containing the list of device name/IP address pairs to populate in the TreeView	Specify the path to a text file containing a list of mappings that pair device names to IP addresses. This script does not support the discovery of devices that use IPv6 addresses.
	
	For example: `c:\DeviceList.txt`
	
	In the file, separate each mapping with a comma, no spaces, with each pair on a single line. All mappings must be formatted properly for the job to run successfully.
	
	For example:
	
	`Intuity,10.41.5.30`
	
	`AvayaOneX,10.41.5.20`
	
	Place the file in a location that is accessible by the account under which the `NetIQmc` service is running on the agent. This script supports UNC shares if the agent's parent account has authority to access the share.

Trap Receiver IP address	Specify the IP address for the NetIQ Trap Receiver (NTR) Server. This script does not support IPv6 addresses. The default is `localhost`.
Trap Receiver TCP port	Specify the TCP port for the NTR Server. The default is `2735`.

3 SNMPTraps Knowledge Scripts

AppManager provides the following Knowledge Scripts for monitoring SNMP Traps resources. From the Knowledge Script view of Control Center, you can access more information about any Knowledge Script by selecting it and clicking **Help**. In the Operator Console, select any Knowledge Script in the Knowledge Script pane and press **F1**.

Knowledge Script	What It Does
AddMIB	Adds management information bases (MIBs) for monitoring by the SNMPTraps_TrapMonitor Knowledge Script.
TrapMonitor	Monitors for incoming SNMP traps from devices forwarded by NetIQ Trap Receiver. Raises events when traps are received and for Trap Receiver availability.

3.1 AddMIB

Use this Knowledge Script to add management information base (MIB) files, enabling you to convert trap object identifiers (OIDs) into object descriptive names (ODEs) to make monitored traps more readable with the SNMPTraps_TrapMonitor Knowledge Script. The MIB files should be `ASN.1` text files with a `.txt`, `.my`, or `.mib` file extension, and not compiled MIB files.

Use this script to copy a MIB file from a location you specify to the MIB tree located in the `netiq/AppManager/bin/MIBs` folder. If you select Yes for the *Reload MIB tree?* parameter, you can also reload all MIBs in the tree without restarting the AppManager agent. A restart of the AppManager agent automatically reloads the MIB tree.

In This Scenario	Set These Parameters
You want to add a MIB file to the MIB tree, but do not want the addition to take effect until after the next restart of the AppManager agent.	*Install additional MIB files?*: Select **Yes**. *Full path to MIB files* and *List of MIB files*: Provide location and name of MIB file you want to add. *Reload MIB tree?*: Set to **No** (unselected).
You manually copied a MIB file to the MIB directory and want to reload all MIBs in the directory.	*Install additional MIB files?*: Set to **No** (unselected). *Full path to MIB files* and *List of MIB files*: Leave blank. *Reload MIB tree?*: Select **Yes**. *MIB reload timeout*: Set new timeout value or accept default of 10 seconds.
Due to compiler errors, you edited some MIBs in the MIB directory. Now you want to reload the MIBs to ensure the errors have been fixed.	*Install additional MIB files?*: Set to **No** (unselected). *Full path to MIB files* and *List of MIB files*: Leave blank. *Reload MIB tree?*: Select **Yes**. *MIB reload timeout*: Set new timeout value or accept default of 10 seconds.

Resource Objects

- NT_MachineFolder

Default Schedule

By default, this script runs once.

Setting Parameter Values

Set the **Values** tab parameters as needed.

Parameter	How to Set It
General Settings	
Job Failure Notification	
Event severity if AddMIB job fails unexpectedly	Set the event severity level, from 1 to 40, to reflect the importance when this script fails unexpectedly. The default is 5.
Additional Settings	
Event Details	
Event detail format	Select whether to view event details in an HTML table or in plain text. The default is HTML Table.
Tracing (for advanced users only)	Use the tracing settings in this section only with the help of NetIQ Technical Support.
Raise event with job execution log?	Select **Yes** to raise an event when the job execution log is created. The default is unselected.
Logging level	Select the logging level you want to monitor. The options are Off, Fatal, Error, Warn, Info, Debug, or All. Use these settings only with the help of Technical Support. The default is Warn.
Derive event severity from most severe event log entry?	Select **Yes** to calculate the event severity for the *Raise event with job execution log* parameter based on the most severe event log entry. The default is Yes.
Event severity (if automatic severity computation not selected above)	If you did not select **Yes** for the *Derive event severity from most severe event log entry* parameter, set the event severity level, from 1 to 40, to reflect the importance of the event raised with the creation of the job execution log. The default is 40.
MIB Load Configuration Settings	
Raise event with the list of currently installed MIBs?	Select **Yes** to raise an informational event that provides a list of all MIBs installed in the MIB tree. The default is Yes.
Event severity for the list of currently installed MIBs	Set the event severity level, from 1 to 40, to reflect the importance of an event that provides a list of all MIBs in the MIB tree. The default is 25.
Install additional MIB files?	Select **Yes** to install additional MIB files. The default is Yes. If you select Yes for this parameter, but do not enter any value for both the *Full path to MIB files* and the *List of MIB files* parameters, the script loads or reloads all MIB files in the `NetIQ\AppManager\bin\MIBs` folder.

Parameter	How to Set It
Raise event if installation of MIB files succeeds?	Select **Yes** to raise an event if the installation of the MIB files succeeds. The default is Yes.
Event severity when installation of MIB files succeeds	Set the severity level, from 1 to 40, to indicate the importance of an event in which the installation of the MIB files succeeds. The default is 25.
Raise event if installation of MIB files fails?	Select **Yes** to raise an event if the installation of the MIB files fails. The default is Yes.
Event severity when installation of MIB files fails	Set the severity level, from 1 to 40, to indicate the importance of an event in which the installation of MIB files fails. The default is 10.
Full path to MIB files	Specify the full path to the folder that contains the MIB files you want to install. Place the file in a location that is accessible by the account under which the `NetIQmc` service is running on the agent. This script supports UNC shares if the agent's parent account has authority to access the share.
List of MIB files	Provide a comma-separated list of the MIB files you want to install. The MIB files should not be compiled MIB files. The MIB files you specify must be located in the folder you identified in the *Full path to MIB files* parameter.
Reload MIB tree?	Select **Yes** to update the MIB tree. The default is Yes.
Raise event if reloading of MIB tree succeeds?	Select **Yes** to raise an event if the reloading of the MIB tree succeeds. The default is Yes.
Event severity when reloading of MIB tree succeeds	Set the severity level, from 1 to 40, to indicate the importance of an event in which the reloading of the MIB tree succeeds. The default is 25.
Raise event if reloading of MIB tree fails?	Select **Yes** to raise an event if AppManager fails to reload the specified MIB files. The default is Yes. Failure scenarios include: ◆ MIB reload timeout period expired. ◆ Not all specified MIB files were installed.
Event severity when reloading of MIB tree fails	Set the severity level, from 1 to 40, to indicate the importance of an event in which the reloading of the MIB tree fails. The default is 10.
Raise event if reload MIB parser warnings received?	Select **Yes** to raise an event if warning messages are received during the reload process. The default is Yes. A potential warning scenario could be if not all the specified MIB files were loaded to the MIB tree.
Event severity when reload MIB parser warnings received	Set the severity level, from 1 to 40, to indicate the importance of an event in which warning messages are received during the reload process. The default is 15.
MIB reload timeout	Specify the length of time AppManager should attempt to update the MIB tree before timing out and raising a failure event. The default is 10 seconds.

3.2 TrapMonitor

Use this Knowledge Script to monitor v1, v2, and v3 traps sent by remote devices. You can configure the script to generate events for each SNMP trap received. You can also configure this script to raise AppManager events based on the different alarm types used by the monitored SNMP traps.

After you run the Knowledge Script, the SNMPTraps_TrapMonitor job waits for notification of a trap from the NetIQ Trap Receiver server or servers. When the server receives a trap, the TrapMonitor job determines whether the IP address of the source device matches a device that the job is currently monitoring.

You can also use this script to create a new object in the Navigation pane or TreeView, with custom display name format, when a trap is received from a device that is not currently in the Navigation pane or TreeView.

This script also lets you filter the list of devices monitored, with filters based on OID (object identifier) values, ODE (object descriptive name) values, and varbind values, and exclusion filters based on MIB subtrees and trap source devices.

In addition, this script allows you to customize the AppManager event messages that correspond to SNMP traps listed in the `SNMPTraps_AlarmMappings.csv` file that comes with this module. For more information, see Section 3.3, “Customizing AppManager Events for Trap Source Devices,” on page 40.

The SNMPTraps_TrapMonitor script also includes vendor-specific formatting for Avaya G3 and Avaya Communication Manager traps to make the AppManager event messages for those traps easier to read. For more information, see the following topics:

- “Formatting Event Message Text for Avaya G3 Traps” on page 44
- “Formatting Event Message Text for Avaya CM Traps” on page 45

Prerequisite

Before running the SNMPTraps_TrapMonitor script, configure AppManager Security Manager with the community string and version information for each device you want to monitor. Security Manager entries for SNMP v1 and v2 are optional, but SNMP v3 traps require a Security Manager entry.

If you already use other modules that monitor SNMP traps, such as AppManager for Avaya Communication Manager or AppManager for Network Devices, you can continue to use any existing SNMPTrap Security Manager entries.

The type of Security Manager information you configure varies according to the version of SNMP implemented on the device. AppManager for SNMP Traps supports SNMP versions 1, 2, and 3.

Configuration for SNMP Versions 1 and 2

To set up Security Manager for SNMP v1 or SNMP v2 traps, complete the following fields on the **Custom** tab in Security Manager:

Field	Description
Label	`SNMPTraps` This script also supports Security Manager entries labeled `SNMPTrap`, which is a label used by other modules that you might have already installed, such as AppManager for Avaya Communication Manager or AppManager for Network Device.
Sub-label	Specify whether the community string is used for a single device or for all devices: ◆ For a single device, list the IP address for the community string. ◆ For all devices, enter `default`.
Value 1	Specify the community string for the device or devices.
Value 2	Leave this field blank.
Value 3	Leave this field blank.

Configuration for SNMP Version 3

AppManager for SNMP supports the following modes for SNMP version 3 (SNMP v3):

- No authentication; no privacy
- Authentication; no privacy
- Authentication and privacy

In addition, the module supports the following protocols for SNMP v3:

- MD5 (Message-Digest algorithm 5, an authentication protocol)
- SHA (Secure Hash Algorithm, an authentication protocol)
- DES (Data Encryption Standard, an encryption protocol)
- AES (Advanced Encryption Standard, an encryption protocol, 128-bit keys only)

Configure SNMP v3 information for each device monitored by each proxy computer.

If you plan to monitor SNMP v3 traps, install the NetIQ Trap Receiver and the AppManager agent on the *same* computer to prevent malicious users from gaining secure access to the information in these traps. The SNMPTraps_TrapMonitor script notifies you if an SNMP v3 trap source device's corresponding NetIQ Trap Receiver IP address does not match the IP address of the AppManager agent monitoring it.

The SNMPTraps_TrapMonitor script does not fully validate SNMP v3 credentials retrieved from Security Manager for a particular device or set of devices, and the script does not notify you if these credentials do not match. As a result, the SNMPTraps_TrapMonitor script might miss some SNMP v3 traps if you do not enter the Security Manager credentials properly.

For SNMP v3 configuration, complete the following fields in the **Custom** tab of Security Manager for the proxy agent computer.

Field	Description
Label	`SNMPTraps` This script also supports Security Manager entries labeled `SNMPTrap`, which is a label used by other modules that you might have already installed, such as AppManager for Avaya Communication Manager or AppManager for Network Devices.
Sub-label	Specify the IP address, or enter `default` for all devices that do not have a specific IP address entry.
Value 1	Specify the SNMP user name, or *entity*, configured for the device. All SNMP v3 modes require an entry in this field.
Value 2	Specify the name of the context associated with the user name or entity entered in **Value 1**. A *context* is a collection of SNMP information that is accessible by an entity. If possible, enter a context that provides access to all MIBS for a device. If the device does not support context, type an asterisk (*). All SNMP v3 modes require an entry in this field.
Value 3	Specify the combination of protocol and password appropriate for the SNMP v3 mode you have implemented. ◆ For *no authentication/no privacy mode*, leave this field blank. ◆ For *authentication/no privacy mode*, enter `md5` or `sha` and the password for the protocol, separating each entry with a comma. For example, enter `md5,abcdef` ◆ For *authentication/privacy mode*, enter `md5` or `sha` and the associated password, and then enter `des` and the associated password, separating each entry with a comma. For example, enter `sha,hijklm,des,nopqrs`

Resource Objects

- NT_MachineFolder
- TRAP_SOURCE_DEVICE

Default Schedule

The default interval for this script is **Asynchronous**.

Setting Parameter Values

Set the **Values** tab parameters as needed:

Parameter	How to Set It
General Settings	
Job Failure Notification	
Event severity if TrapMonitor job fails unexpectedly	Set the event severity level, from 1 to 40, to reflect the importance when this script fails unexpectedly. The default is 5.
Event Details	
Event detail format	Select whether to view event details in an HTML table or in plain text. The default is HTML Table.

Trap source address format	Select the elements of the trap source address you want to include in AppManager event messages. The default is Both.
	
	If you select **Host ID**, the event message lists the host ID in brackets before the trap details. For example:
	
	`[RALDVAP655]: Trunk Layer 1 state changed to up`
	
	If you select **Source IP**, the event message lists the IP address for the source in brackets before the trap details. For example:
	
	`[10.22.124.33]: Trunk Layer 1 state changed to up`
	
	If you select **Both**, the event message lists the name of the host and the IP address in brackets, followed by the trap details. For example:
	
	`[RALDVAP655 (10.22.124.33)]: Trunk Layer 1 state changed to up`

Format trap data according to SNMP version?	Select the version of SNMP to determine the type of formatting that will be used for trap event messages. The data provided by each format is the same, and only the layout is different. The default is SNMP v2.
Include prefix information to format event messages for Netcool adapter?	Select **Yes** if you are using the NetIQ AppManager Connector for IBM Tivoli Netcool/OMNIbus, and want to format trap events for the connector. If you select Yes, at the start of the resulting AppManager event short message, the script will add four values that are each preceded by tilde characters (~) that get used by the Netcool connector. The default is unselected.
Varbind display options	**Note** Enabling any of the following varbind display parameters might negatively impact the speed at which traps are processed.

Parameter	How to Set It
Display ‘friendly’ ODEs in event messages?	Select **Yes** if you want to include spaces in the varbind ODE name in the event detail message. This parameter will add spaces in the varbind ODE name in between characters that differ in case, and it will add spaces between characters and numbers. The default is Yes. For example, the varbind ODE name `v1clogHistFacility` would display as `v1 clog Hist Facility`.
Include varbind OID in event messages?	Select **Yes** to add the OID (object identifier) of the varbind in a separate column in the Varbind table. The default is unselected.
Include varbind MIB name in event messages?	Select **Yes** to add the name of the MIB in front of the varbind ODE in the details of the event message varbinds. For example, the varbind ODE name `v1clogHistFacility` would display as `CISCO-SYSLOG-MIB::v1clogHistFacility`. The default is unselected.
Trap Filters	**Note** The SNMPTraps_TrapMonitor script processes the include filters first, and then it processes the exclude filters applied against those results. Also, each filter parameter is processed in the order listed below, so the *List of OIDs and ODEs* parameters are processed before the *List of MIB subtrees* parameters.
Include Filters	
List of OIDs and ODEs to include	Specify the object identifiers (OIDs) of the traps you want to monitor, ignoring all other traps. You can type one OID or a list of OIDs. If you use a list, separate the OIDs with a comma, without any spaces. This parameter also supports the use of ODEs (descriptive names) if the relevant MIBs were loaded into the MIB subtree. If the relevant MIBs are not installed by this module, load them with the SNMPTraps_AddMIB Knowledge Script. The case of the ODEs in your list must match the case of the ODEs as they are defined in the MIBs. This parameter does not support wildcard characters or regular expressions. List the OID or ODE information in the following format: *`MIBName`*`::`*`TrapName`* or `NumericalTrapOID` Separate multiple trap OIDs or ODEs with commas, without any spaces. For example: `EXTREME-DOS-MIB::extremeDosThresholdCleared, 1.3.6.1.4.1.1916.4.14.0.2`

Parameter	How to Set It
File with list of OIDs and ODEs to include	If you have many OID values to monitor, you can specify the full path to a file that contains a list of the OID values you want to include. This parameter also supports the use of ODEs if the relevant MIBs were loaded into the MIB subtree. If the relevant MIBs are not installed by this module, load them with the SNMPTraps_AddMIB Knowledge Script. The case of the ODEs in your list must match the case of the ODEs as they are defined in the MIBs. List each OID or ODE value on a separate line in the file, and format them in the manner described in the previous parameter. Place the file in a location that is accessible by the account under which the `NetIQmc` service is running on the agent. If you place the file in the `NetIQ\AppManager\bin\SNMPTraps` folder on the local agent, you do not need to specify a full path to the file. This script supports UNC shares if the agent's parent account has authority to access the share. If you edit the contents of this file after running this job, restart the job to include the updates.
List of MIB subtrees to include	Specify a set of MIB subtrees for which you want to monitor all child traps. The script ignores any traps that are not part of the listed MIB subtrees. You can type one MIB subtree or a list of MIB subtrees. If you type a list, separate the subtrees with a comma, without any spaces. If you add multiple MIB subtrees in this parameter, the script ignores any higher-level subtrees if you also included a lower-level subtree in the list. For example, if you list both `1.3.6.1.4.1.9148` and `1.3.6.1.4.1.9148.1`, the script ignores the first, higher-level entry to focus on the second, lower-level entry in the MIB subtree.
File with list of MIB subtrees to include	If you have many MIB subtrees to monitor, you can specify the full path to a file that contains a list of the subtrees you want to include. Each MIB subtree in the file should be on a separate line. Place the file in a location that is accessible by the account under which the `NetIQmc` service is running on the agent. If you place the file in the `NetIQ\AppManager\bin\SNMPTraps` folder on the local agent, you do not need to specify a full path to the file.This script supports UNC shares if the `netiqmc` service account has permission to access the share.
Exclude Filters	

Parameter	How to Set It
List of OIDs, ODEs, and varbind values to exclude	Specify the OIDs, ODEs, and varbind values of the traps you do not want to monitor. You can specify one OID or ODE, or a list of OIDs and ODEs. If you use a list, separate the OIDs and ODEs with a comma, without any spaces. The case of the ODEs in your list must match the ODEs as they are defined in the MIBs. List the ODE information in the following format: *MIB Name*::*Trap Name* For example: CXC-MIB::callHeld,CXC-MIB::callRetrieved List the varbind value information in the following format: *MIBName*::*TrapName*+*MIB Name*::*Varbind Name*=*Value* For example: CXC-MIB::callHeld+CXC-MIB::Varbind1=1 If you need to use a comma for the *Value*, above, use a tilde (~) character in place of the comma every location where a comma should appear.
File with list of OIDs, ODEs, and varbind values to exclude	If you have many OIDs, ODEs, and varbind values to exclude, you can specify the full path to a file that contains a list of the OIDs, ODEs, and varbind values that you want to exclude. List each value on a separate line in the file, and format them in the manner specified in the previous parameter. The case of the ODEs in your list must match the ODEs as they are defined in the MIBs. Place the file in a location that is accessible by the account under which the NetIQmc service is running on the agent. If you place the file in the NetIQ\AppManager\bin\SNMPTraps folder on the local agent, you do not need to specify a full path to the file.This script supports UNC shares if the netiqmc service account has permission to access the share. If you edit the contents of this file after running this job, restart the job to include the updates.
List of MIB subtrees to exclude	Specify the MIB subtrees of the traps you want to exclude from monitoring so you can focus on a smaller set of traps. You can type one MIB subtree or a list of MIB subtrees. If you use a list, separate the subtrees with a comma, without any spaces.
File with list of MIB subtrees to exclude	If you have many MIB subtrees you want to exclude from monitoring, you can specify the full path to a file that contains a list of the subtrees you want to exclude. Each MIB subtree in the file should be on a separate line. Place the file in a location that is accessible by the account under which the NetIQmc service is running on the agent. If you place the file in the NetIQ\AppManager\bin\SNMPTraps folder on the local agent, you do not need to specify a full path to the file.This script supports UNC shares if the netiqmc service account has permission to access the share.
Additional Settings	

Parameter	How to Set It
Monitor devices not yet discovered?	Select **Yes** to create AppManager events for traps forwarded by devices that are not currently displayed in the Navigation pane or TreeView. The default is unselected.
Discover new devices when traps received?	Select **Yes** to enable the script to discover a new device and create a new object for that device in the Navigation pane or TreeView if a device that has not yet been discovered receives a trap. The default is unselected.
Reverse lookup DNS hostname from an unknown trap source IP address?	Select **Yes** to perform a reverse lookup of the IP address to determine the DNS hostname of the discovered device. The IP address for the device displays as part of the name of the object created for the discovered device in the Navigation pane or TreeView. The default is Yes. This parameter only applies to devices that are not listed in the following parameter, *File containing additional device name/IP address pairs*. Enabling this parameter might negatively impact the performance of this script. If you select Yes for this parameter, you must also select Yes for the *Monitor devices not yet discovered?* parameter to enable the discovery of new devices.

Parameter	How to Set It

File containing additional device name/IP address pairs	Specify the path to a list of mappings that pairs device names to IP addresses.
	
	When a trap is received from an undiscovered device, this parameter determines the object display name in the Navigation pane or TreeView if a match is found. The *Monitor devices not yet discovered?* and the *Discover new devices when traps received?* parameters must both be set to Yes to enable the discovery of new devices.
	
	If you selected No for the *Discover new devices when traps received?* parameter, this parameter formats the short event message of the relevant trap so a device name is specified.
	
	In the file, list just one mapping pair per line, and separate the mappings with a comma, no spaces. Use the following format for the mappings in this file:
	
	`DeviceName,IPAddress`
	
	For example:
	
	`DeviceA,10.41.5.100`
	
	`DeviceB,10.41.5.102`
	
	If the received trap's source IP address does not match the source IP address contained in any monitored Navigation pane or TreeView object, but the IP address *does* match a source IP address provided in the file for this parameter, the script displays the new device in the Navigation pane or TreeView in one of the following three formats:
	
	`Trap Source: DNSHostname [IP Address]`
	
	`Trap Source: CustomDeviceName [IP Address]`
	
	`Trap Source: [IP Address]`
	
	For example:
	
	`Trap Source: DeviceA [10.41.5.101]`
	
	Note You can use IPv6 addresses in your file, and the script will format the alarm event message properly to use the correct custom display name for the object. However, the script will *not* discover a device that uses an IPv6 address if you selected Yes for the *Discover new devices when traps received?* parameter. Traps containing IPv6-formatted source addresses will not have a corresponding object created in the Navigation pane or TreeView.
	
	Place the file in a location that is accessible by the account under which the `NetIQmc` service is running on the agent. This script supports UNC shares if the `netiqmc` service account has permission to access the share. If you place the file in the `NetIQ\AppManager\bin\SNMPTraps` folder on the local agent, you do not need to specify a full path to the file.

Parameter	How to Set It
File with list of IP addresses not yet discovered to exclude	Specify the full path to a file that contains an exclusion list of the IP addresses for devices that have not yet been discovered. This parameter lets you exclude a set of devices that are not relevant and, as a result, are never included as part of the unknown device support in the module. Do not use this parameter to specify a set of already-discovered devices to exclude. Also, this parameter does not exclude already-discovered devices. In the file, list one IPv4 or IPv6 address per line. The script ignores any lines that start with a hash (#) character, and the script also ignores any blank lines. Place the file in a location that is accessible by the account under which the `NetIQmc` service is running on the agent. If you place the file in the `NetIQ\AppManager\bin\SNMPTraps` folder on the local agent, you do not need to specify a full path to the file.This script supports UNC shares if the `netiqmc` service account has permission to access the share. If you edit the contents of this file after running this job, restart the job to include the updates.
List of Trap Receiver IP address/TCP port pairs	Specify a list of mappings that pair IP addresses with the TCP port numbers for any Trap Receiver servers that can receive traps from a device that is not currently discovered in the Navigation pane or TreeView, or Trap Receiver servers that can receive any traps from IPv6 devices. The *IP address* is for the NetIQ Trap Receiver (NTR) server that received the forwarded trap, and the *port* is the TCP port where the SNMPTraps_TrapMonitor job connects to the relevant Trap Receiver server. Use only the IP address, not the host name for a Trap Receiver server. Traps containing IPv6-formatted source addresses will *not* have a corresponding object created in the Navigation pane or TreeView. Format the pairs in the following manner: `10.22.50.100:2735`.
Custom message mapping file	Specify the path to the file containing the custom event short message and alarm severity information for individual SNMP trap ODEs. This file also allows you to customize the alarm severity for individual varbind values and substitute text strings for individual varbind values. The default is `SNMPTraps_AlarmMappings.csv`, located in the `NetIQ\AppManager\bin\SNMPTraps` folder on the AppManager agent. This file is pre-populated with objMapping, severityMapping, and varbindMapping entries for each trap defined in each MIB installed by the module. If the `SNMPTraps_AlarmMappings.csv` file does not exist at the target location when you install this module, the installation process will create a new file in the target location. All fields except for the `<trap text>` are *not* case-sensitive. For more information about the `SNMPTraps_AlarmMappings.csv` file, see Section 3.3, “Customizing AppManager Events for Trap Source Devices,” on page 40.
Tracing (for advanced users only)	

Parameter	How to Set It
Logging level	Select the logging level you want to monitor. The options are Off, Fatal, Error, Warn, Info, Debug, or All. The default is Warn. Use these tracing settings only with the help of NetIQ Technical Support.
Monitor SNMP Traps	
Event Notification	
Raise critical alarm event?	Select **Yes** to raise an AppManager event when the script receives a trap with a trap ODE that matches an objMapping or a severityMapping entry with an AlarmSeverity of *critical* in the file specified in the *Custom message mapping file* parameter. The default is Yes. A *critical* alarm indicates that a condition that impacts service has occurred and an immediate corrective action is required. An example of a critical event is when a total loss of service occurs, and that service must be restored.
Event severity when critical alarm received	Set the severity level, from 1 to 40, to indicate the importance of an event in which a monitored device receives a trap that maps to a critical alarm. The default is 5.
Raise major alarm event?	Select **Yes** to raise an AppManager event when the script receives a trap with a trap ODE that matches an objMapping entry or a severityMapping entry with an AlarmSeverity of *major* in the file specified in the *Custom message mapping file* parameter. The default is Yes. A *major* alarm indicates that a service-affecting condition has developed and requires an urgent corrective action. An example of a major event is when a severe degradation of service occurs, and the full capability of that service must be restored.
Event severity when major alarm received	Set the severity level, from 1 to 40, to indicate the importance of an event in which the script receives a trap that maps to a major alarm. The default is 10.
Raise minor alarm event?	Select **Yes** to raise an AppManager event when the script receives a trap with a trap ODE that matches an objMapping entry or a severityMapping entry with an AlarmSeverity of *minor* in the file specified in the *Custom message mapping file* parameter. The default is Yes. A *minor* alarm indicates the existence of a fault condition that is not service-affecting, but you should take corrective action to prevent a more serious fault.
Event severity when minor alarm received	Set the severity level, from 1 to 40, to indicate the importance of an event in which the script receives a trap that maps to a minor alarm. The default is 15.
Raise warning alarm event?	Select **Yes** to raise an AppManager event when the script receives a trap with a trap ODE that matches an objMapping or a severityMapping entry with an AlarmSeverity of *warning* in the file specified in the *Custom message mapping file* parameter. The default is Yes. A *warning* alarm indicates the detection of a potential or impending service-affecting fault before any significant effects have occurred. You should take action to further diagnose the problem, if necessary, and then correct the problem to prevent it from becoming a more serious service-affecting fault.
Event severity when warning alarm received	Set the severity level, from 1 to 40, to indicate the importance of an event in which the script receives a trap that maps to a warning alarm. The default is 20.

Parameter	How to Set It
Raise unmapped alarm event?	Select **Yes** to raise an AppManager event when the script receives a trap with a trap ODE that matches an objMapping or a severityMapping entry with an AlarmSeverity of *unmapped* in the file specified in the *Custom message mapping file* parameter. The default is Yes. An *unmapped* alarm indicates that no mapping exists for a trap that the script does not recognize.
Event severity when unmapped alarm received	Set the severity level, from 1 to 40, to indicate the importance of an event in which the script receives a trap that maps to an unmapped alarm. The default is 15.
Raise indeterminate alarm event?	Select **Yes** to raise an AppManager event when the script receives a trap with a trap ODE that matches an objMapping or a severityMapping entry with an AlarmSeverity of *indeterminate* in the file specified in the *Custom message mapping file* parameter. The default is Yes. An *indeterminate* alarm indicates that an entry exists in the `SNMPTraps_AlarmMappings.csv` file, but the severity level cannot be determined due to a missing varbind value, or the entry contains a dynamic value that cannot be specified.
Event severity when indeterminate alarm received	Set the severity level, from 1 to 40, to indicate the importance of an event in which the script receives a trap that maps to an indeterminate alarm. The default is 20.
Raise cleared or resolved alarm event?	Select **Yes** to raise an AppManager event when the script receives a trap with a trap ODE that matches an objMapping or a severityMapping entry with an AlarmSeverity of *cleared* or *resolved* in the file specified in the *Custom message mapping file* parameter. The default is Yes. A *cleared* or *resolved* alarm indicates that one or more previously reported alarms have been cleared.
Event severity when cleared or resolved alarm received	Set the severity level, from 1 to 40, to indicate the importance of an event in which the script receives a trap that maps to a cleared or resolved alarm. The default is 25.
Raise event if Trap Receiver is unavailable?	Select **Yes** to raise an event if a monitored Trap Receiver is not available. The default is Yes.
Event severity when Trap Receiver is unavailable	Set the severity level, from 1 to 40, to indicate the importance of an event in which a monitored Trap Receiver is unavailable. The default is 5.
Raise event if Trap Receiver becomes available?	Select **Yes** to raise an event if a monitored Trap Receiver becomes available. The default is No.
Event severity when Trap Receiver becomes available	Set the severity level, from 1 to 40, to indicate the importance of an event in which a monitored Trap Receiver becomes available. The default is 25.

3.3 Customizing AppManager Events for Trap Source Devices

This module installs a file named `SNMPTraps_AlarmMappings.csv` in the `NetIQ\AppManager\bin\SNMPTraps` folder on the AppManager agent. You can use the contents of this `.csv` file to customize the text of the AppManager events for the trap source devices you are monitoring.

View a brief video demonstration of this feature on the NetIQ YouTube channel:

 http://www.youtube.com/watch?v=09jT2CnbjIA

Customizing the AppManager events in this way allows you and other AppManager users to easily identify problems related to SNMP traps in AppManager and quickly address those problems instead of spending time trying to decipher the meaning of the default trap messaging.

This file contains a list of NetIQ-specific mapping entries for the values in the MIBs installed by the module. The mapping entries in the `.csv` file use one of the following formats:

- *objMapping* entries map a trap ODE to an AppManager event short message and an AppManager event severity. For more information, see “Customizing Event Messages and Severities Based on Trap ODE” on page 41.
- *severityMapping* entries map varbind ODE and varbind value to an AppManager event severity. For more information, see “Customizing Event Severities Based on Varbind Values” on page 42.
- *varbindMapping* entries map varbind ODE and varbind value to a human-readable string used by the AppManager event, replacing the default values generated by the trap. For more information, see “Customizing Event Message Text Based on Varbind Values” on page 43.

The SNMPTraps_TrapMonitor Knowledge Script uses these mappings to generate an AppManager event with an event severity or event message based on the parameters you selected in the SNMPTraps_TrapMonitor script.

For example, if you selected **Yes** for the *Raise critical alarm parameter* in the SNMPTraps_TrapMonitor script, and the following events occur:

1. The SNMPTraps_TrapMonitor script receives a trap,
2. The trap’s ODE matches an objMapping entry or a varbind ODE in the trap matches a severityMapping entry in the `.csv` file,
3. The entry in the `.csv` file has an AlarmSeverity of *critical*,

then the SNMPTraps_TrapMonitor script generates an AppManager event for that critical trap. You can specify the event severity level of the trap by using the *Event severity when critical alarm received* parameter, or you can use the default AppManager severity level for that parameter, which is 5.

The objMapping, severityMapping, and varbindMapping entry types in the `SNMPTraps_AlarmMappings.csv` file also supports the use of *derived fields*, which are varbind values that the module formats into values that are easier to understand.

If the `SNMPTraps_AlarmMappings.csv` file does not exist at the target location when you install this module, the installation process installs a new file in the target location. If the `SNMPTraps_AlarmMappings.csv` file already exists in the target location when you install this module, the installation process renames the existing file `SNMPTraps_AlarmMappings_OLD.csv` and installs the new `SNMPTraps_AlarmMappings.csv` file in the target location. If you already have a list of alarm mappings, you can specify this file using the *Custom event mapping file* parameter in the SNMPTraps_TrapMonitor script.

Each SNMPTraps_TrapMonitor job uses a unique version of the `SNMPTraps_AlarmMappings.csv` file from the `NetIQ\AppManager\bin\SNMPTraps` folder. Also, the TrapMonitor job reloads the `.csv` file every 24 hours.

Customizing Event Messages and Severities Based on Trap ODE

When the SNMPTraps_TrapMonitor job receives a trap, and the trap ODE matches an objMapping entry in the `SNMPTraps_AlarmMappings.csv` file (or the file you specified in the *Custom message mapping file* parameter), the job creates an AppManager event for that trap. You can customize the AppManager event message and event severity for that trap by editing the entry in the `.csv` file.

The AppManager event message uses the event short message that comes after the fourth tilde (~) character in the relevant entry in the .csv file. The severity for the event corresponds with the event severity parameter for that type of alarm in the trap. Use the parameters in the *Event Notification* section of the SNMPTraps_TrapMonitor script to specify the AppManager alarm settings.

The ODE entries in the .csv file are not case-sensitive, and they use the following format:

`objMapping,`*`MIBName`*`::`*`TrapName`*`,`*`AlarmSeverity`*`,~`*`NetcoolPrefix1`*`~`*`NetcoolPrefix2`*`~`*`NetcoolPrefix3`*`~`*`TrapText`*

- `objMapping` states that this line contains a mapping of an SNMP trap ODE to an AppManager event short message and alarm severity category.
- *`MIBName`*`::`*`TrapName`* specifies the trap ODE that you are mapping. The ODE contains both the MIB name and the trap name.
- *`AlarmSeverity`* specifies the alarm severity category for this ODE. The following severity category values are supported: *critical, major, minor, warning, indeterminate*, *unmapped*, and *cleared*.
- The final section of the entry is used to format the actual AppManager event short message. This portion is split into four sections, with each section prefixed with a tilde (~). Each of these four sections can contain normal text or substitution variables. *Substitution variables* represent different varbind values or derived fields (also known as derived varbind values) that you can substitute into an AppManager event message created for a trap. Substitution variables are listed with braces, such as *`{DerivedHostID}`*, and these variables should contain a substituted value at runtime.
  - The three *`NetcoolPrefix`* labels are only for Netcool connector support. The first label signifies an alert group, the second label signifies an alert key, and the third label signifies the source host and address. If you are not using the Netcool connector, leave these entries blank except for the three tildes (~~~).
  - *`TrapText`* specifies the event message text that will display for the AppManager event. You can customize this text, and the text is required.

The following is an example of an objMapping entry from the .csv file:

```
objMapping,LOAD-BAL-SYSTEM-MIB::loadBalTrapNoMem,major,~~~~ MAWS BOOT service
cannot access SES database
```

The SNMPTraps_TrapMonitor script supports the following substitution variables in ODE entries:

- *`{DerivedHostID}`* is the name of the trap-forwarding device, which can be a DNS host name or a custom name provided as input into the SNMPTraps_TrapMonitor script or input as part of a discovered Navigation pane or TreeView object.

- *{DerivedSourceIP}* is the IP address of the forwarding device.
- *{DerivedTrapName}* is the ODE of the SNMP trap received.

The SNMPTraps_TrapMonitor script also supports varbindMapping substitution variables.

Customizing Event Severities Based on Varbind Values

When the SNMPTraps_TrapMonitor job receives a trap, and the trap varbind value matches a severityMapping entry in the `SNMPTraps_AlarmMappings.csv` file (or the file you specified in the *Custom message mapping file* parameter), the job creates an AppManager event that corresponds to the type of alarm in the SNMP trap. You can customize the AppManager event severity for that trap by editing the corresponding entry in the `SNMPTraps_AlarmMappings.csv` file.

Mapping Varbind Values to Alarm Severities

Entries in the `.csv` file can specify a one-to-one mapping of varbind values to a alarm severities.

The severityMapping entries in the file are not case-sensitive, and they use the following format:

`severityMapping,`*`MIBName`*`::`*`VarbindName`*`,`*`VarbindValue`*`,`*`AlarmSeverity`*

- `severityMapping` states that this line contains a mapping of a varbind value to an AppManager event severity category.
- *`MIBName`*`::`*`VarbindName`* specifies the varbind ODE that you are mapping. The varbind ODE contains both the MIB name and the varbind name.
- *`VarbindValue`* specifies an alphanumeric string for the varbind being represented.
- *`AlarmSeverity`* specifies the alarm severity category for this varbind ODE. The following severity category values are supported: *critical, major, minor, warning, indeterminate*, and *cleared*.

The following is an example of a severityMapping entry from the `.csv` file:

```
severityMapping,G700-MG-MIB::cmgTrapSeverity,1,cleared
```

Mapping Derived Fields to Alarm Severities

An entry in the `.csv` file that maps a derived field to an alarm severity uses the following format:

`severityMapping,`*`DerivedFieldName`*`,`*`DerivedFieldValue`*`,`*`AlarmSeverity`*

- `severityMapping` states that this line contains a mapping of a derived field to an AppManager event severity category.
- *`DerivedFieldName`* specifies the derived field that you are mapping. This name should be prefixed with the word `Derived`, though it is not required. Also, this name cannot contain any double colon characters (`::`).
- *`DerivedFieldValue`* specifies an alphanumeric string that could be a possible value for the derived field being represented.
- *`AlarmSeverity`* specifies the alarm severity category for this derived field. The following severity category values are supported: *critical, major, minor, warning, indeterminate*, and *cleared*.

The following is an example of how a severityMapping entry mapped with a derived field value might look:

```
severityMapping,DerivedDefAudFaultMessage,A:1,cleared
```

Customizing Event Message Text Based on Varbind Values

An AppManager event for an SNMP trap can contain a number of values for the various varbinds for that trap, and many times the varbinds do not clearly describe the conditions of the trap. You can replace a varbind value with a string of text that is more relevant and “human-readable” than the original varbind values.

Mapping Varbind Values to Event Text

Entries in the `SNMPTraps_AlarmMappings.csv` file (or the file you specified in the *Custom message mapping file* parameter) can specify a one-to-one mapping of varbind values to more readable strings of text.

The varbindMapping entries in the file are not case-sensitive, and they use the following format:

`varbindMapping,`*`MIBName`*`::`*`VarbindName`*`,`*`VarbindValue`*`,`*`HumanReadableString`*

- `varbindMapping` states that this line contains a mapping of a varbind alphanumeric value to a “human-readable” string.
- *`MIBName`*`::`*`VarbindName`* specifies the varbind ODE that you are mapping. The varbind ODE contains both the MIB name and the varbind name.
- *`VarbindValue`* specifies an alphanumeric string for the varbind being represented.
- *`HumanReadableString`* specifies any relevant identifying text you want to use to replace the varbind value.

The following is an example of a varbindMapping entry from the `.csv` file:

```
varbindMapping,AVAYA-LOAD-MIB::avGenOpLastFailureIndex,222,ftpResumeNotSupported
```

Mapping objMapping Entries to Event Text

In addition to varbindMapping entries, you can apply substitutions to objMapping entries in the `.csv` file. If an objMapping entry contains a substitution variable that matches a varbind ODE defined in the relevant MIB, the resulting AppManager event short messages are updated so that the substitution variable is replaced with the alphanumeric value for that varbind ODE.

If the varbind ODE has a matching varbindMapping entry in the file specified in the *Custom message mapping file* parameter, the corresponding “human-readable” string replaces that alphanumeric value in the event short message.

For example, an objMapping entry includes the following event short message:

```
~~~~ Trunk Layer 2 state changed to {applianXAlarmStatus}
```

This message displays like this if no matching varbindMapping entry exists:

```
Trunk Layer 2 state changed to 1
```

In this instance, the value of the varbind is substituted directly, but the “1” might not mean anything to you. If the file contains a matching varbindMapping entry, the following displays in the event short message:

```
Trunk Layer 2 state changed to up
```

Mapping Derived Values to Event Text

Entries in the `SNMPTraps_AlarmMappings.csv` file (or the file you specified in the *Custom message mapping file* parameter) can specify a one-to-one mapping of derived fields to more readable strings of text.

The varbindMapping entries in the `.csv` file are not case-sensitive, and they use the following format:

`varbindMapping,`*`DerivedFieldName`*`,`*`DerivedFieldValue`*`,`*`HumanReadableString`*

- `varbindMapping` states that this line contains a mapping of a derived field value to a "human-readable" string.
- *`DerivedFieldName`* specifies the derived field that you are mapping. This name should be prefixed with the word `Derived`, though it is not required. Also, this name cannot contain any double colon characters (`::`).
- *`DerivedFieldValue`* specifies an alphanumeric string that could be a possible value for the derived field being represented.
- *`HumanReadableString`* specifies any relevant identifying text you want to use to replace the derived field value.

The following is an example of a varbindMapping entry with a derived field value from the `.csv` file:

```
varbindMapping,DerivedDefAudFaultMessage,0:LINK_PORTS,Check error log
```

Formatting Event Message Text for Avaya G3 Traps

AppManager for SNMP Traps includes vendor-specific formatting for Avaya G3 traps to make the AppManager event messages for those traps easier to read.

If you run an SNMPTraps_TrapMonitor job, and an Avaya G3 trap successfully passes all relevant filters to create an AppManager event for that trap, the SNMPTraps_TrapMonitor script provides vendor-specific formatting for all Avaya G3 traps (which are defined under the 1.3.6.1.4.1.6889.1.8.1.0 MIB subtree).

The detail portion of the event message for Avaya G3 traps includes the following information in the **Trap details** table:

- **CM Hostname**: the script populates this value with the g3clientExternalName varbind, which defines the external name of the G3 client. If this varbind is not populated, the source IP address is set as the value.
- **Maintenance Object**: the script populates this value with the g3alarmsMaintName varbind, which defines the Maintenance Object Name. Known values are populated in the `SNMPTraps_AlarmMappings.csv` file with varbindMapping entries so that a human-readable string is used. If the job does not find a corresponding varbindMapping entry in the `SNMPTraps_AlarmMappings.csv` file, the relevant cell will display just the varbind value.
- **Generation Time**: the script populates this value with the g3alarmsAlarmNumber varbind, and it states the time the alarm was generated.
- **Resolution Time**: the script populates this value with the g3alarmsAlarmNumber varbind, and it states the time that the condition causing the alarm was fixed.
- **New/Modified Alarm**: this value can be *New* for a new alarm condition, or *Modified* for an existing alarm condition that was updated.
- **Derived G3 Alarm Port**: the script populates this value with the g3alarmsPort varbind, which defines the location port in that particular system, such as `cabinet(01-44):carrier(A-E):slot(01-20):port(01-32)`.

The following formatting changes occur in the **Derived G3 Alarm Port** column:

- *cmgTrapSubsystem* will be replaced with *SS*.
- *cmgTrapOnBoard* will be replaced with *OB*.
- *cmgTrapLocation* will be replaced with *LOC*.
- *cmgActiveControllerAddress* will be replaced with *ACA*.
- *cmgTrapTypes* will be replaced with *cmgTT*.

If a varbind is empty and cannot be used to display a value in the new **Trap details** table, a value of *N/A* appears in the corresponding cell of the table.

Formatting Event Message Text for Avaya CM Traps

The SNMPTraps_TrapMonitor script includes vendor-specific formatting for Avaya Communication Manager (Avaya CM) traps to make the AppManager event messages for those traps easier to read.

If you run an SNMPTraps_TrapMonitor job, and an Avaya CM trap successfully passes all relevant filters to create an AppManager event for that trap, the SNMPTraps_TrapMonitor script provides vendor-specific formatting for two of the three traps defined in the INADS-MIB definition:

- INADS-MIB::inadssnmpAlarm
- INADS-MIB::inadssnmpAlarmSet

Both of those traps expose the inadssnmpAlarmMessage varbind. The following is an example of the inadssnmpAlarmMessage varbind:

```
inadssmnpAlarmMessage: 1001119999 10/12:24,ACT|<27>May 10 12:23:28 CDOM
snmpd[1425]: +01:00 2013 426 1 com.avaya.vsp | 0 cannot open /pro/net/snmp6
```

The detail portion of the event message for these two Avaya CM traps includes the following information in the **Trap details** table:

- **DerivedInadsProdID**: the script populates this value with the first 10 characters of the inadssnmpAlarmMessage varbind. In the example above, this value is represented by `1001119999`.
- **DerivedInadsAlarmTime**: the script populates this value with the date and time from the inadssnmpAlarmMessage varbind. In the example above, the script uses the `10/12:24` data and adds the current month in front of it, resulting in the following value: `June 10 12:24`.
- **DerivedInadsAlarmType**: the script populates this value with the three characters following the product ID and the timestamp in the inadssnmpAlarmMessage varbind. In the example above, this value is represented by `ACT`.
- **DerivedInadsAlarmMessage**: the script populates this value with the remaining content in the inadssnmpAlarmMessage varbind. In the example above, this value is represented by `<27>May 10 12:23:28 CDOM snmpd[1425]: +01:00 2013 426 1 com.avaya.vsp | 0 cannot open /pro/net/snmp6`.