Administrator Guide

NetIQ® AppManager®

January 2012

Legal Notice

NetIQ AppManager is covered by United States Patent No(s): 05829001, 05986653, 05999178, 06078324, 06397359, 06408335.

This product claims FIPS compliance by use of one or more of the Microsoft cryptographic components listed below. These components were certified by Microsoft and obtained FIPS certificates via the CMVP.

893 Windows Vista Enhanced Cryptographic Provider (RSAENH)

894 Windows Vista Enhanced DSS and Diffie-Hellman Cryptographic Provider (DSSENH)

989 Windows XP Enhanced Cryptographic Provider (RSAENH)

990 Windows XP Enhanced DSS and Diffie-Hellman Cryptographic Provider (DSSENH)

997 Microsoft Windows XP Kernel Mode Cryptographic Module (FIPS.SYS)

1000 Microsoft Windows Vista Kernel Mode Security Support Provider Interface (ksecdd.sys)

1001 Microsoft Windows Vista Cryptographic Primitives Library (bcrypt.dll)

1002 Windows Vista Enhanced Cryptographic Provider (RSAENH)

1003 Windows Vista Enhanced DSS and Diffie-Hellman Cryptographic Provider (DSSENH)

1006 Windows Server 2008 Code Integrity (ci.dll)

1007 Microsoft Windows Server 2008 Kernel Mode Security Support Provider Interface (ksecdd.sys)

1008 Microsoft Windows Server 2008

1009 Windows Server 2008 Enhanced DSS and Diffie-Hellman Cryptographic Provider (DSSENH)

1010 Windows Server 2008 Enhanced Cryptographic Provider

1012 Windows Server 2003 Enhanced Cryptographic Provider (RSAENH)

This product may also claim FIPS compliance by use of one or more of the Open SSL cryptographic components listed below. These components were certified by the Open Source Software Institute and obtained the FIPS certificates as indicated.

918 - OpenSSL FIPS Object Module v1.1.2 - 02/29/2008 140-2 L1

1051 - OpenSSL FIPS Object Module v 1.2 - 11/17/2008 140-2 L1

1111 - OpenSSL FIPS Runtime Module v 1.2 - 4/03/2009 140-2 L1

Note: Windows FIPS algorithms used in this product may have only been tested when the FIPS mode bit was set. While the modules have valid certificates at the time of this product release, it is the user's responsibility to validate the current module status.

EXCEPT AS MAY BE EXPLICITLY SET FORTH IN THE APPLICABLE END USER LICENSE AGREEMENT, NOTHING HEREIN SHALL CONSTITUTE A WARRANTY AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OR CONDITION OF FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW AND ARE EXPRESSLY DISCLAIMED BY NETIQ, ITS SUPPLIERS AND LICENSORS.

Contents

About this Book and the Library

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

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

Intended Audience

This guide is intended for senior-level system and network administrators who are responsible for managing one or more AppManager sites. Managing an AppManager site involves tasks such as configuring and maintaining site communication, setting up and maintaining security roles and user rights, establishing event and data handling policies, and maintaining and managing the repository database and data archiving.

This guide assumes that you are already familiar with your operating system, network configuration, basic database management, and the servers and applications you intend to monitor.

This guide also assumes you have a working knowledge of or access to other documentation for performing basic system management and AppManager activities. For example, you should be familiar with AppManager components and terminology and managing user accounts and permissions for your operating environment.

Other Information in the Library

The library provides the following information resources:

Installation Guide

Provides detailed planning and installation information.

Control Center User Guide

Provides information about managing groups of computers, including running jobs, responding to events, creating reports, and working with the Control Center console.

Operator Console User Guide

Provides information for system and network administrators working with the AppManager Operator Console.

Upgrade and Migration Guide

Provides information about upgrading from a previous version of AppManager.

Module management guides

Provide information about installing and monitoring specific applications with AppManager.

NetIQ UNIX Agent documentation

Provides information about installing, upgrading, and configuring the NetIQ UNIX Agent and UNIX Agent Manager.

Conventions

The library uses consistent conventions to help you identify items throughout the documentation. The following table summarizes these conventions.

Convention	Use
Bold	• Window and menu items • Technical terms, when introduced
Italics	• Book and CD-ROM titles • Variable names and values • Emphasized words
`Fixed Font`	• File and folder names • Commands and code examples • Text you must type • Text (output) displayed in the command-line interface
Brackets, such as [*value*]	Optional parameters of a command
Braces, such as {*value*}	Required parameters of a command
Logical OR, such as *value1*|*value2*	Exclusive parameters. Choose one parameter.

About NetIQ Corporation

NetIQ, an Attachmate business, is a global leader in systems and security management. With more than 12,000 customers in over 60 countries, NetIQ solutions maximize technology investments and enable IT process improvements to achieve measurable cost savings. The company's portfolio includes award-winning management products for IT Process Automation, Systems Management, Security Management, Configuration Audit and Control, Enterprise Administration, and Unified Communications Management. For more information, please visit www.netiq.com.

Contacting Sales Support

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

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

Contacting Technical Support

For specific product issues, please 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, please email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

Qmunity, 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, Qmunity helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, please visit http://community.netiq.com.

Chapter 1
Introduction to AppManager Site Administration

This chapter provides an overview of the components that make up an AppManager management site, the communication between AppManager components, the role of the site administrator, and examples of ways you can configure the site to suit your organization's needs.

The AppManager Management Site

In AppManager, a **management site** always consists of one AppManager repository, at least one AppManager management server and Operator Console or Control Center console, and some number of AppManager agents on agent computers that report events and data through the management server to the repository. A single management site may have multiple management servers to distribute processing and communication for agent computers, but each management server communicates with only one repository. Therefore, the repository and the management servers that communicate with it define the management site.

In planning the configuration of your site, decide whether to use a single management server or multiple management servers. If you install multiple management servers within a given management site (that is, for a single repository), you should explicitly designate a primary and secondary management server for each agent computer to communicate with. Explicitly designating a primary and a secondary, or backup, management server enables you to distribute processing load, provide failover support for agent computers, and control which management servers communicate with which agents based on the constraints of your network, department requirements, or other factors.

Using a single management server may simplify site administration and troubleshooting of your AppManager environment, but it can overload the management server, inhibiting system performance.

Note
Although you can install multiple management servers in your environment without explicitly specifying a primary and secondary management server for each agent computer, NetIQ Corporation does not recommend this practice. Choosing not to designate management servers in the recommended way can limit the amount of control you have over which management servers communicate with which agent computers. Always install a single management server and explicitly designate its agent computers before installing a second management server.

For a review of issues to consider in planning the number of management servers and management sites, see "Implementation Guidelines" in the *Installation Guide for AppManager* and "Developing a Management Site and Site Policies" on page 4 in this guide.

Understanding Site Communication

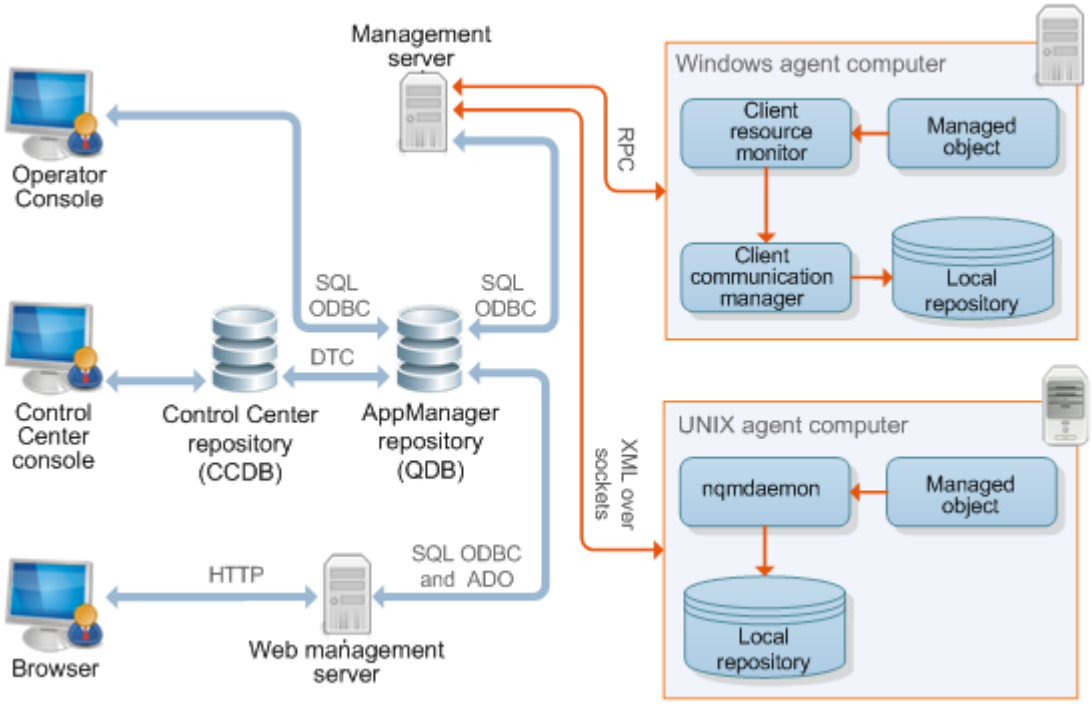
AppManager manages and monitors the availability, performance, and server health of operating system services, hardware, and applications through **jobs**. When you run a Knowledge Script on an agent computer and create a job:

- The Operator Console or Control Center console delivers information about the properties you have set for the Knowledge Script to the AppManager repository (QDB).
- When the management server next checks the QDB, it identifies the new job and collects all of the information about the job to send it to the appropriate agent computers.

Note
In addition to submitting new and changed job information, the management server periodically polls all of the agent computers it is allowed to communicate with to check the health of the agents on each agent computer.

- The NetIQ Corporation AppManager Client Resource Monitor (`NetIQmc`) service on the agent computer receives the job information from the management server, stores the information in a local repository, then executes the job to begin monitoring.
- When the job runs, the Knowledge Script invokes program objects that collect performance data or perform tasks by accessing raw system statistics, through APIs, or using other methods.

  The following diagram illustrates the basic flow of job information between the Operator Console or Control Center console and the agent (managed) computer (in this diagram the agent computer and the managed computer are the same).



- If the job detects an event condition or collects data, the NetIQ Corporation AppManager Client Communication Manager (`NetIQccm`) service or `nqmdaemon` daemon sends the event information or data point to the management server. If the `NetIQccm` service or `nqmdaemon` daemon cannot communicate with the management server, it saves the event information or data point in the agent computer's local repository until the management server is available.
- When the management server receives event information or data points from the agent computer, it forwards this information to the repository server to update the AppManager repository.
- Once the repository is updated, any new events or data points are sent to the Operator Console or Control Center at its next update interval (for example, in the next 30 seconds for an active view or in five minutes for a background view).

Understanding the Site Administrator's Role

The site administrator, working with an implementation team, senior-level system administrators, and application experts, is typically responsible for determining site-level policies for managing jobs, events, and data, and for configuring and maintaining AppManager components.

Ideally, most, if not all, of these decisions would be made before installing AppManager in a production environment, but often these policies and the implications of the decisions you make are not fully understood until the site is up and running. In most cases, changes can be made and policies refined after you have installed but it is always best to perform the planning and pre-installation steps described in the *Installation Guide for AppManager* before you install any AppManager components.

The site administrator should actively participate in the following actions during the planning and pilot deployment phases:

- Develop a core list of management goals.
- Develop a deployment and plan that addresses your network and site configuration. For example, which components should be installed together and which should be installed separately and whether your management site requires a single management server or multiple management servers.
- Develop a security plan to determine the level of security to use, the user authentication mode you are using for SQL Server, and the number of users and administrators to be given access.
- Verify pre-installation tasks are complete for the production environment, including a check of network connections, account requirements, and account permissions.

For large and widely distributed organizations, many of the decisions you need to make are relatively complex and require a thorough understanding of your own network requirements and constraints of the AppManager architecture. For these organizations, in particular, NetIQ Corporation recommends that you review all of this guide and the *Installation Guide for AppManager*.

After you have successfully installed all of your AppManager components, the site administrator typically performs a variety of post-installation tasks to properly configure the environment. These tasks include:

- Defining or identifying the SQL Server logins and users who should have access to AppManager
- Changing the account information for the agent or the management server
- Setting event-handling policies and preferences for the site
- Setting data-handling policies and preferences for the site
- Creating core Knowledge Script groups and monitoring policies for the site
- Updating application-specific information

Once AppManager is installed and configured, the site administrator typically monitors the health of AppManager components, performs periodic database maintenance, optimizes communication flow and console performance, maintains user accounts and security profiles, and troubleshoots problems within the environment.

Some of the site administrator's common activities are performed using the Operator Console and Control Center console, but many of the tasks require the administrator to use other tools or programs. For example, setting up AppManager users may require the administrator to use standard Windows administrative tools, the SQL Server Management Studio, and the AppManager Security Manager. Therefore, the site administrator should be familiar with these administrative tools as well as the basic operation of AppManager.

Developing a Management Site and Site Policies

As discussed in the *Installation Guide for AppManager,* a key element in the successful deployment of AppManager is understanding the characteristics of your environment and the network you are going to monitor. Although agents, jobs, and event and data handling policies are typically deployed and refined over time, one of the first issues the site administrator must confront is how to configure the core AppManager components to best suit the current environment and anticipated changes.

To determine whether you need one management site with a single management server, one management site with multiple management servers, or multiple management sites with multiple repositories and multiple management servers, consider several factors:

- The network bandwidth, latency, and topology for the subnets you will be monitoring
- The departmental or functional structure of the organization and the expectations and level of autonomy associated with different groups in the organization
- The granularity of management that will best suit the organization

To help you determine how to configure one or more management sites to suit your environment, consider the most common deployment scenarios and evaluate how the specific characteristics of your organization might fit these scenarios. Keep in mind that these scenarios are only a few common examples of the many possible ways you can deploy and organize your site.

Managing a Small, Internal Network

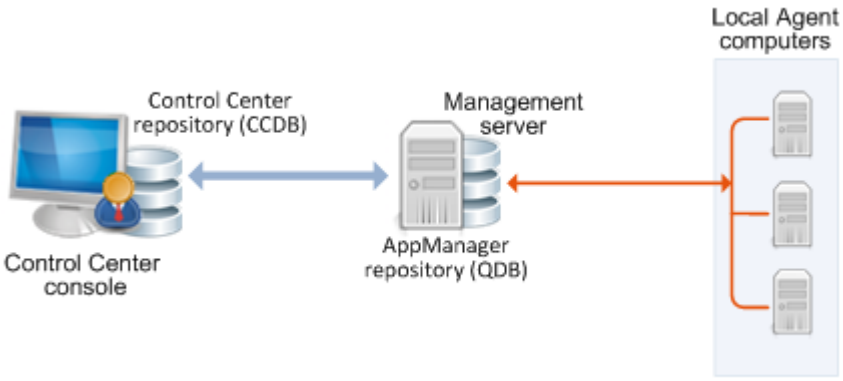
In smaller organizations, it is common to monitor a networked domain or a simple trusted set of domains. An organization like this may be primarily interested in monitoring key application and file and print servers for internal users and a few key business critical services. For example, this organization might focus on basic monitoring of CPU, memory, and disk space for all servers and workstations and specific performance statistics, such as response time and system availability, for important services such as the messaging system and a customer database.

In a small environment like this, with relatively simple monitoring needs for 150 or so servers, a typical site configuration involves one repository and one management server, installed together on the same computer. An environment like this may have a very small administrative staff and need to deploy the core AppManager components on a single computer to simplify maintenance and console viewing. Also, as the focus is on basic performance and availability, the organization can expect fewer events and likely only requires a few charts or reports to show status or trends, and therefore will only collect a few key data streams.

Having the repository and management server installed together on the same computer has the following advantages:

- Eliminates network communication problems between the repository, management server, and Control Center or Operator Console
- Eliminates the need for any special account permissions associated with accessing another computer over the network or through a firewall
- Simplifies maintenance and troubleshooting because components are centrally located
- Reduces the importance of developing carefully considered security, data, event, and job handling policies, database management and backup plans, and monitoring strategies

The diagram below illustrates this type of site configuration:



For small organizations or pilot deployments, these benefits can outweigh the disadvantage of burdening a single computer with the management server's communication load and the repository's storage and communication requirements.

Managing a Mid-size Network with Local and Remote Facilities

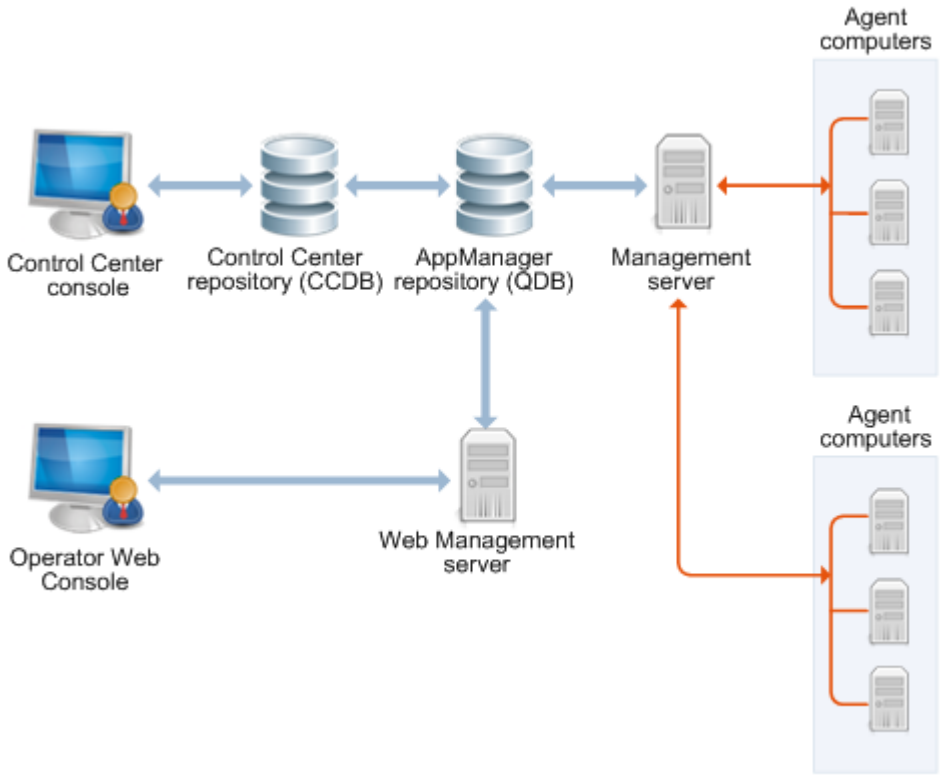
A small- to mid-size organization may contain from 151 to 600 servers and involve monitoring a local network and remote facilities. As the number of servers increases, this organization may put strain on the management server, generate more frequent events, need more sophisticated reports, and need to collect and save more data, all of which require more database resources and more database management. In addition, an organization of this type may have a larger administrative staff and need multiple Control Center consoles or may require a Web management server and Operator Web Console access for multiple local and remote users.

For this environment, the repository and management server should probably be installed on separate computers. In addition, because the organization is monitoring some computers remotely, the site administrator needs to evaluate the reliability and bandwidth of the WAN connection and possibly plan for scheduled communication links between the remote computers and the management server.

Having the repository and management server installed on different computers offers the following advantages:

- Eases the workload on the repository server and management server computers by distributing the workload to two separate computers
- Reduces system requirements for the computers where the components are installed and provides additional space for database growth
- Provides more flexibility in configuring how and when agent computers communicate with the management server

This configuration is still relatively straightforward and easy to maintain. The following diagram illustrates this type of site configuration:



No inherent drawbacks are associated with this configuration for a small or mid-size organization, or even for larger organizations with basic monitoring needs. However, over time you may place stress on this environment if you:

- Add a large number of servers
- Dramatically increase the monitoring you do or the data you collect
- Add widely distributed facilities to the management site
- Start to experience network bandwidth, latency, topology, or reliability issues

If your organization is considering this type of configuration, consider the following potential site administration improvements:

- More planning and testing to optimize communication channels, particularly for monitoring remote computers
- Evaluating security and port requirements more carefully, particularly if the management server is inside of a firewall and some number of computers being monitored are outside of a firewall (which is a common configuration if you are monitoring more than the organization's internal network)
- If your organization has a larger administrative staff or more console computers, more planning for AppManager security roles and profiles to restrict access to some views and activities
- Setting up special domain accounts or trust relationships for certain types of monitoring and to allow remote installation of the AppManager agent

Monitoring Large Environments with Multiple Management Servers

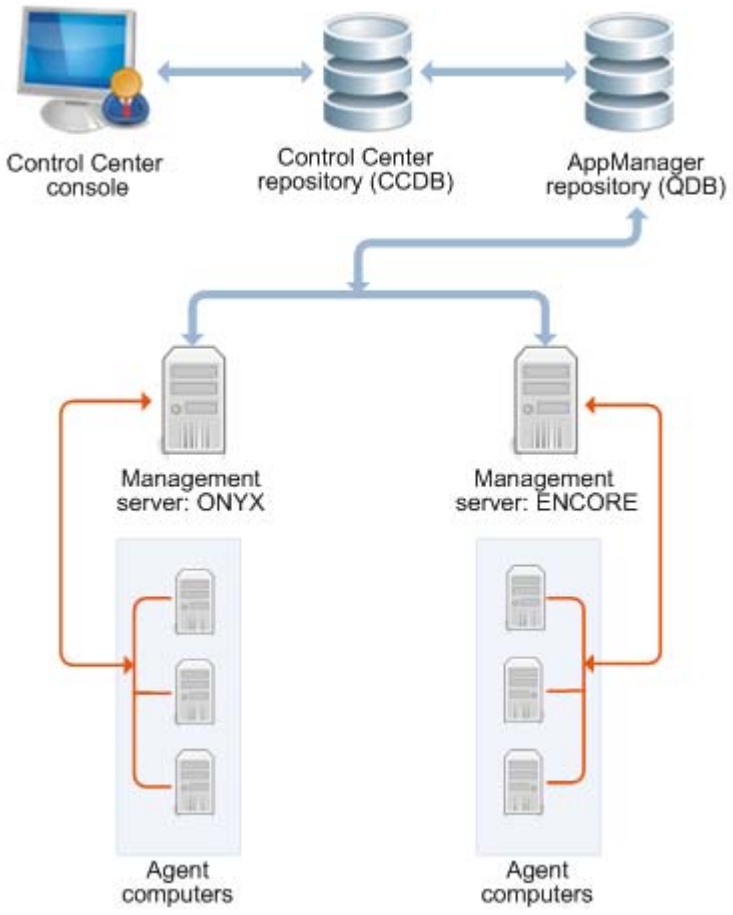
For organizations that need to monitor more than 600 servers, including remote facilities, it is often helpful to install additional management servers to distribute the communication workload. Using two or more management servers in a management site provides the following advantages:

- Reduces the chances of the management server becoming a bottleneck for event and data handling
- Provides you with a way to balance the communication load between the management servers to ensure optimum efficiency for your specific network configuration
- Allows you to establish a primary management server and a backup management server for each of your agent computers to provide failover support
- Provides better scalability for organizations that need to grow quickly or are expanding their monitoring requirements

Using more than one management server allows more control and flexibility in managing site communication but it requires far more planning to implement effectively. For example, you must decide if a second management server is strictly a backup for a primary management server or if each management server is responsible for a specific set of agent computers.

You must also decide how many management servers you can reasonably manage within a single site. Although there is no specific restriction on the number of management servers you can use, most organizations can handle four management servers before the complexity of the site becomes too difficult to manage.

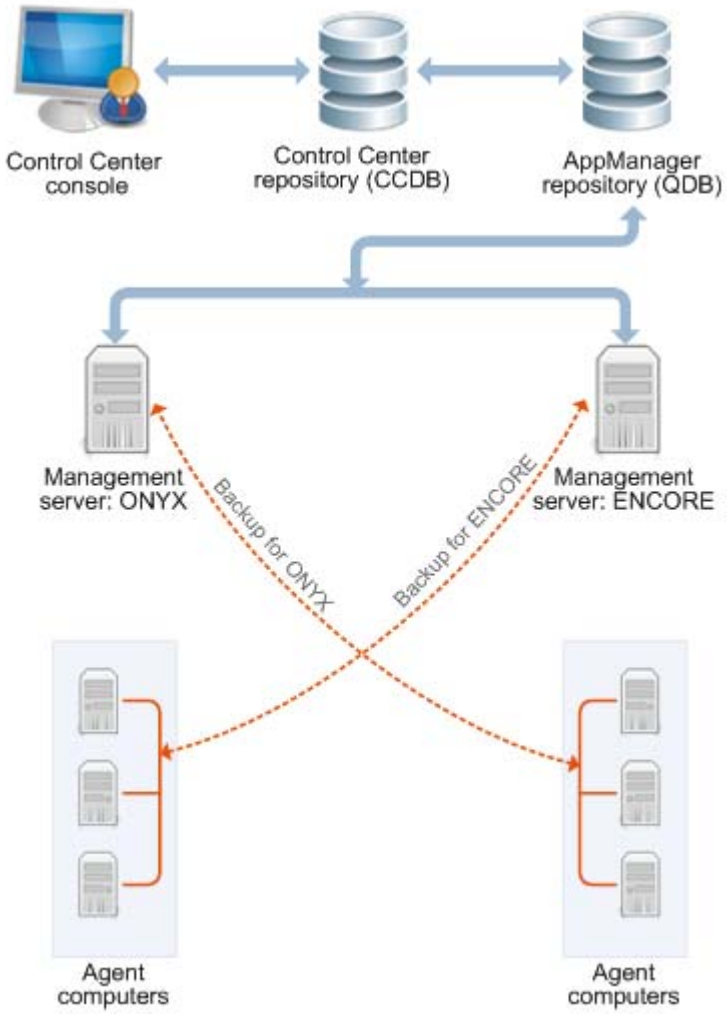
The following diagram represents a simple two-management server site configuration:



Each management server has been designated as a primary management server for several agent computers.

All of the agent computers that send information to the management server ONYX are also configured to use the management server ENCORE as their secondary, or backup, management server. In this scenario, the computer ENCORE is a powerful, high-speed computer that can handle the extra load from the agent computers that are typically managed by the computer ONYX. If communication with ONYX is interrupted, its agent computers automatically begin communicating with their secondary management server, ENCORE.

The site administrator could also designate ONYX as a secondary management server for the computers managed by the computer ENCORE, so that both computers have a backup management server, or could designate a separate management server computer that does not have any agent computers to be a secondary management server. In this second scenario, the backup management server is normally "idle" and only takes over the communication with agent computers when either of the primary management servers fails. If you decide to configure a primary management server as a secondary for another management server, it is good practice to configure this management server so that it can accommodate its own agents plus the full complement of agents from the management server it is backing up.



As this example illustrates, installing multiple management servers provides flexibility but can increase the complexity of site management and requires planning.

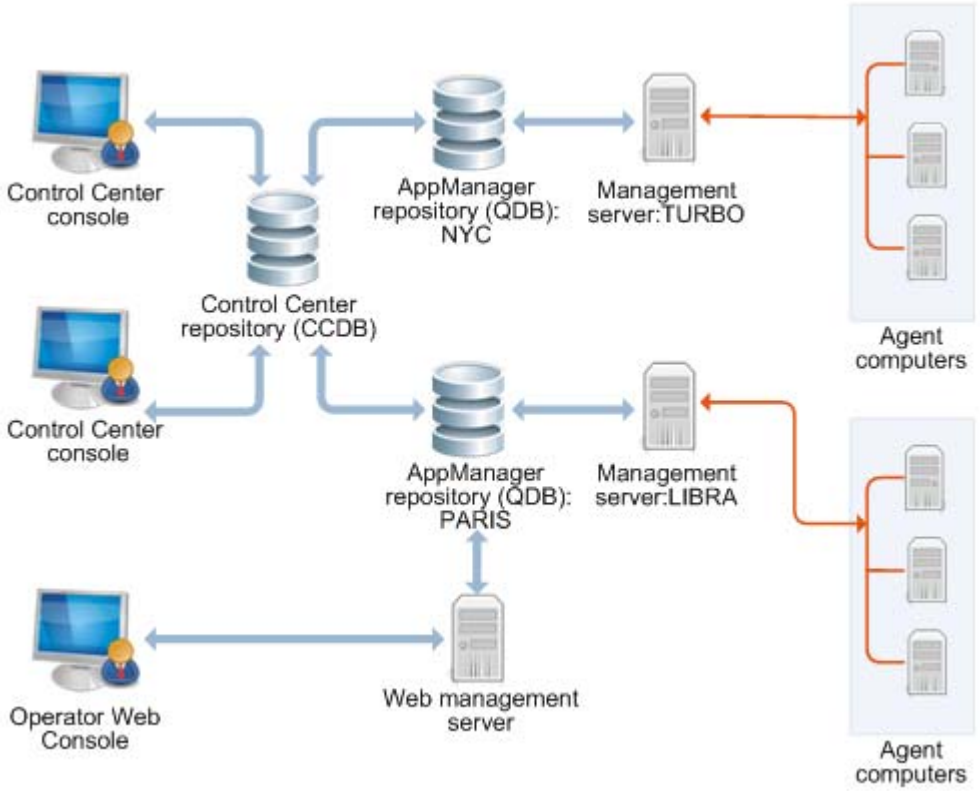
Monitoring a Widely Distributed Enterprise

A single management site with one repository and a small number of management servers is sufficient to meet the monitoring needs of most organizations. For large-scale, distributed networks, however, a single site may not be possible, manageable, or desirable. For an enterprise with computer resources widely distributed across a national or international network, multiple management sites may be the most practical solution.

Because so many key elements of a monitoring strategy and communication control are defined at a site level, keeping multiple repositories synchronized can be very difficult. For example, you can establish consistent monitoring policies for all of the computers in your network. With AppManager Control Center, you define these policies once, and they are automatically replicated to each management site. In addition, Control Center lets you assign permissions so that each administrative team manages its own site.

In a decentralized environment with multiple administrative teams that operate independently, consistency across multiple repositories may not present an issue. If your organization is decentralized or requires only minimal standardization (for example, by establishing a common set of reports for all sites to use but letting each site define its own event handling policies), multiple management sites may best suit your needs.

The following diagram illustrates this type of site configuration:



Defining a Management Site

You define the configuration of an AppManager management site when you install AppManager components. You start by creating the repository, then install the management server and identify the repository to which the management server connects.

Once the core components are installed, you install AppManager agents on the computers you want to monitor and specify the management server with which each agent computer can communicate. Once agents are installed and discovered, you can use the UNIX Agent Manager or you can use the AMAdmin_SetPrimaryMS and AMAdminUNIX_SetPrimaryMS Knowledge Scripts to set or change the primary and secondary management server for each agent computer.

Planning and Staging a Deployment

In general, the more planning you do before deploying AppManager in a production environment, the more successful your implementation will be, particularly if your organization is very large with complex network and security issues. In practice, most of the characteristics of your installation can be modified after installation, so many organizations deploy a basic monitoring environment and refine policies and procedures over time. For suggestions about how to stage a deployment of AppManager components over time, see the "Staging the Deployment" chapter in the *Installation Guide for AppManager*.

Defining Site-Level Policies

In addition to decisions about the basic configuration of your AppManager management site, other policy decisions will also need to be made, from defining security roles for AppManager users to establishing a database management and backup strategy. The remainder of this guide focuses on the key issues that typically need to be addressed.

Some topics, such as setting up security roles and identifying AppManager users, apply for all environments, regardless of size. Other topics, such as adjusting the flow of data from agent computers to the management server, only apply to organizations that need to control and customize the communication to suit their network topology or bandwidth constraints.

Managing Multiple Sites

Large and widely distributed organizations often require more than one repository because of the number of computers being managed, the distribution of computers on the network, or the amount of data they collect. Using multiple repositories, however, increases the complexity of managing your environment and the burden on the administrative staff in performing routine tasks. If you need to use multiple repositories, you can:

- Manage each site independently, with each site responsible for deciding its own monitoring, event handling, and data handling policies and managing its own repository
- Manage computers, jobs, events, and data collection for some or all of the sites from a single, central location using Control Center

Depending on the structure and policies of your organization, there are benefits and drawbacks to either approach for managing multiple sites. In general, however, the first option works best for decentralized organizations with distinct functional or departmental units. For organizations that require centralized and standardized IT management across sites, the second approach provides the administrator with a single console for managing computer groups, monitoring policies, events, and data across multiple repositories. For more information about Control Center, see the *Control Center User Guide for AppManager*.

Chapter 2
Site Communication and Security

This chapter describes ways you can customize the communication between your agent computers and the AppManager management server. It includes a brief overview of the communication protocols for AppManager components and describes the security and configuration options available.

Customizing site communication is optional. You can tailor the methods and frequency of agent computer communications with the management server to suit your network requirements, bandwidth, latency, and operational policies.

AppManager Communication Protocols

AppManager relies on specific types of network connectivity between the computers where AppManager components are installed as well as specific port bindings. For more information about the communication protocols and port bindings that AppManager requires between components, see the *Installation Guide for AppManager*.

Understanding Communication Security Levels

Within any single management site, choose the level of security for communication between the management server and all of the agent computers. The options available are:

- **Unencrypted messages (no security)**: no extra measures taken to secure agent-to-management server communications. All data sent between the management server and the agent is transmitted without encryption, and the agent does not authenticate the identity of the management server.

  The lowest security setting for agent communications is entirely appropriate in many environments. Cleartext communications facilitate troubleshooting and are suitable for a closed network environment. However, many organizations require greater security to ensure data privacy and integrity and to help prevent potential attacks from unauthorized, external sources.

- **Encrypted communication only (Security Level 1)**: a basic level of security. All data transmitted between the management server and the agent is encrypted and decrypted using a session key generated dynamically when the management server is started.
- **Authentication and encrypted communications (Security Level 2)**: a high level of security. The agent uses a predefined key to authenticate the identity of the management server before sending encrypted data. The key information is stored in the repository and a portion of the key is made available for the agent computers to use.

Although a single management server can use one security level to communicate with Windows agents and a different security level to communicate with UNIX agents within a single site, you must select one security level for all of your Windows-based agents, and one security level for all of your UNIX-based agents. All management servers that connect to the same repository must use the same security level for communications with Windows-based agents. And if a key file is required, all Windows agents within the same site must use the same key file. The same restrictions apply to UNIX agents within a management site.

Note
For simplicity and ease of maintenance, NetIQ Corporation recommends that you select one security level to use for all agent computers in a site (Windows and UNIX agents).

Selecting a Security Level for the Agent

In deciding the security level to use, consider the following:

- As you increase the level of security you enforce, you increase the system resources and processing time required to send and receive data and events. In most cases, the increase is nominal, but you should consider this additional load if you are monitoring heavily-burdened computers.
- Changing the security level after installation may interrupt or prevent communication between the management server and agent computers, at least temporarily. Therefore, avoid changing the security level, if possible, or plan carefully for any changes to reduce disruption to your environment.
- To use the highest security level (management server authentication and encryption), generate a key and store this information in the repository, extract a portion of the key into a file, and make the key file available to each agent. In addition, you can periodically repeat these steps to update and replace keys for enhanced security.
- Implementing security level 1 or 2 enables FIPS-compliant algorithms for communication between repositories, management servers, and agents. AppManager retains proprietary encryption algorithms for backward compatibility and supports a mix of FIPS-compliant and non-FIPS-compliant components. However, you also have the option to use only FIPS-compliant algorithms, which makes non-FIPS-compliant AppManager components unreachable. For more information about AppManager FIPS compliance, see "Understanding FIPS Compliance" on page 12.

Although the security levels and issues to consider are similar for Windows-based agents and UNIX-based agents, the steps for managing security are different for Windows and UNIX systems and are discussed separately.

Understanding FIPS Compliance

There are two components to AppManager FIPS compliance:

- The FIPS-compliant algorithms that AppManager uses for security levels 1 and 2
- The Control Center console FIPS-only compliance flag

AppManager implements FIPS-compliant algorithms for security levels 1 and 2. These algorithms secure communication between repositories, management servers, and agents. AppManager retains non-compliant encryption algorithms for backward compatibility with earlier versions of AppManager and supports a mix of FIPS-compliant and non-FIPS-compliant components. For security levels 1 and 2, FIPS-compliant components communicate with each other using FIPS-compliant algorithms and communicate with non-FIPS-compliant components using non-FIPS-compliant encryption algorithms.

Note
AppManager FIPS compliance is independent of operating system FIPS compliance.

The Control Center console offers an option to use only FIPS-compliant security algorithms for security levels 1 and 2. If you choose to implement this option, AppManager no longer supports a mixed security environment and any non-FIPS-compliant AppManager components are no longer available. For more information about implementing FIPS-compliant algorithms, see the *Control Center User Guide for AppManager*.

Using Secure Communication for Windows Agents

AppManager offers the following options for securing the data traffic between a management server and Windows-based agent computers:

- Encryption (Security Level 1)
- Authentication and encryption (Security Level 2)

For more information, see "Understanding Communication Security Levels" on page 11.

For either security level, all communication between the management server and the agent is encrypted using 40-bit RPC encryption.

The option to use encryption and authentication requires the 128-bit Windows High Encryption Pack, which you may need to install on the agent computer. The High Encryption Pack can be exported from the U.S. to worldwide destinations, except where expressly restricted.

Although you normally set the security level for a site during installation, you can change the security level after installation using the `NQKeyGenWindows.exe` utility. You can also use the `NQKeyGenWindows.exe` utility any time you need to create and manage key file information for one of the agent security options.

When running the `NQKeyGenWindows.exe` program on Windows Vista, Windows 7, Windows Server 2008, or Windows Server 2008 R2 with UAC enabled, run the program as an Administrator. To open the Command Prompt window as an Administrator, right-click the Command Prompt program, `cmd.exe`, and click **Run As Administrator**.

Changing the Security Level for Management Servers

After installation, you can use the `NQKeyGenWindows.exe` program to change the security level for communication between the management server and Windows-based agents.

On Windows Server 2008 or Windows Server 2008 R2 with UAC enabled, run the program as an Administrator. To open the Command Prompt window as an Administrator, right-click the Command Prompt program, `cmd.exe`, and click **Run As Administrator.**

If you change the security setting for the management server, update the security setting for all Windows agents in the site. Also, if you are changing from no security to Security Level 1 or 2, generate or identify a repository key to use before changing the security level. For more information, see "Generating a Repository Key for Windows" on page 15.

To change the security level for the management server:

1. On a management server computer, open a Command Prompt window and change to the `NetIQ\AppManager\bin` directory.
2. Run `NQKeyGenWindows.exe` to set the security level for the management server using the following format:

   `NQKeyGenWindows -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`*`\`*`instance`* `-seclev` *`level`*

   For example, to use your current Windows account name and password and set the security level to use encryption only (Security Level `1`) with a QDB installed on the server `NYC2006`, you would type a command-line similar to this:

   ```
   NQKeyGenWindows -db qdb::nyc2006 -seclev 1
   ```

   Notes

   - All management servers that connect to the same repository must use the same security level for all Windows agents.
   - For encryption or management server authentication and encryption, use the same key file.

3. Change the security level for all of your Windows agents to match the new security level setting.
4. Stop and restart the NetIQ AppManager Management Service (`NetIQms`). This allows the management server to receive the new security level information.

Changing the Security Level for Windows Agents

If you change the security level for the management server, you must also update the security setting for every Windows agent.

To change the security level for an agent:

1. Start the Control Center console.
2. In the Enterprise Layout pane, select the Knowledge Scripts view of a management group that includes the agent computer where you want to change the agent security level.
3. In the View pane, select the **AgentConfigSecurityLevel** Knowledge Script.
4. In the Tasks pane, click **Create New Job**.
5. Click the **AMAdmin** tab in the Knowledge Script pane.
6. In the Select Servers dialog box, select the agent computer where you want to run the Knowledge Script and then click **OK**.

7. Click the **Values** tab, and:

   - Select the new security level from the **Security level** list.
   - Set the event notification and event severity parameters as desired.

   Note
   If you change the security level from Security Level 1 or 2 to Unencrypted communications, the management server ignores the event raised because it is not encrypted. Therefore, no event indicator is displayed in the Control Center console if you change the security level to Unencrypted communications. If you are changing from Unencrypted communications to Security Level 1 or 2, you must generate or identify the agent key to use before changing the security level. For more information, see "Generating a Repository Key for Windows" on page 15.

8. Click **OK** to start the job.
9. After updating all of your Windows agents, manually stop and restart each management server in the management site by stopping and then restarting the NetIQ AppManager Management Service (`NetIQms`).

As an alternative, you can run `NQKeyGenWindows.exe` directly on an agent to set the security level for the agent or to set the security level for a remote agent. For example, to change the security level on an agent to use encryption without authentication, type a command similar to this:

```
NQKeyGenWindows -agentseclev 1
```

For more information about using `NQKeyGenWindows` options, see "Key File Utility for Windows Agents" on page 211.

Generating a Repository Key for Windows

If you are using Security Level 1 or 2 (encryption or authentication and encryption) to secure communications between the management server(s) and Windows agent computers, generate a new encryption key and store it in the repository.

When running the `NQKeyGenWindows.exe` program on Windows Vista, Windows 7, Windows 2008, or Windows 2008 R2 with UAC enabled, you must run the program as an Administrator. To open the Command Prompt window as an Administrator, right-click the Command Prompt program, `cmd.exe`, and click **Run As Administrator**.

To generate a new repository key for Windows agents:

1. On a management server computer, open a Command Prompt window and change to the `NetIQ\AppManager\bin` directory.

2. Run the `NQKeyGenWindows.exe` program to generate a new key and store the key information in the repository:

   `NQKeyGenWindows -db database_name:user_name:sql_server -new`

Variable	Description
`database_name`	The name of the AppManager repository.
`user_name`	A valid SQL Server login with permission to access the AppManager repository. **Note** If you are using Windows authentication to connect to the repository, leave the username blank. If you are using SQL Server authentication, type a SQL Server username for connecting to the repository. The program prompts for the password to use for the SQL Server account.
`sql_server\instance`	The name of the SQL Server computer and SQL Server instance name (if applicable) where the AppManager repository is installed.

   For example, to run `NQKeyGenWindows.exe` on a computer named `NYC2003` with Windows authentication, type a command similar to this:

   `NQKeyGenWindows -db qdb::nyc2003 -new`

3. Type a password for the repository key. If you want to extract the key into a file or use this key in another repository, you need to know this password. For information about sharing a key across multiple repositories, see "Extracting and Sharing Key Information from the Repository" on page 17.

4. Run `NQKeyGenWindows.exe` to extract the portion of the key for the agents to use with the following command-line format:

   `NQKeyGenWindows -db database_name:user_name:sql_server\instance -ckey filelocation`

   Note
   If you are using Windows authentication to connect to the repository, leave the username blank. For SQL Server authentication, type a SQL Server username for connecting to the repository. The NQKeyGenWindows.exe program prompts for the password to use for the SQL Server account.

   In specifying a path for the file, use a directory that you can access from the computers to be managed.

5. Stop and restart the NetIQ AppManager Management Service (`NetIQms`) to register the new key with the management server.

Extracting and Sharing Key Information from the Repository

The `NQKeyGenWindows.exe` program can extract repository encryption key information and save it in a password-protected file. Saving this information in a file allows you to share a single key across multiple repositories.

When running the `NQKeyGenWindows.exe` program on Windows 2008 or Windows 2008 R2 with UAC enabled, you must run the program as an Administrator. To open the Command Prompt window as an Administrator, right-click the Command Prompt program, `cmd.exe`, and click **Run As Administrator**

If you want to create this password-protected file, run the `NQKeyGenWindows.exe` program using the following command:

`NQKeyGenWindows -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`* `-skey` *`filelocation`*

Note
If you are using Windows authentication to connect to the repository, leave the username blank. To use SQL Server authentication, type a SQL Server username for connecting to the repository. The NQKeyGenWindows.exe program prompts for the password to use for the SQL Server account.

To check the key into another repository from the file location specified:

1. On a management server computer, open a Command Prompt window and change to the `NetIQ\AppManager\bin` directory.
2. Run the `NQKeyGenWindows.exe` program to import the key pair into the repository.

   For example, if you created the key using the password `^myPass` and extracted the encrypted key to the file location `C:\Security\AMkey.txt`, type the following command to import the key pair into the repository `QDB01` on the computer `SFO2003`:

   ```
   NQKeyGenWindows -db QDB01:smithj:SFO2003 -change C:\Security\AMkey.txt
   ```
3. Use the password you used to create the key in the repository. Provide the key file password `^myPass`.
4. After you import the key, stop and restart the AppManager Management Service (`NetIQms`) to register the new key with the management server.

Extracting the Key File for Windows Agents

The `NQKeyGenWindows.exe` program can extract a portion of the key information stored in the repository and save it in a file. You can then make this agent key file available to all of your Windows agents.

When running the `NQKeyGenWindows.exe` program on Windows 2008 or Windows 2008 R2 with UAC enabled, run the program as an Administrator. To open the Command Prompt window as an Administrator, right-click the Command Prompt program, `cmd.exe`, and click **Run As Administrator**.

To extract the portion of the key for the agents to use:

1. On a management server, run the `NQKeyGenWindows.exe` program with the following command-line format:

   `NQKeyGenWindows -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`* `-ckey` *`filelocation`*

   Note
   If you are using Windows authentication to connect to the repository, leave the username blank. For SQL Server authentication, type a SQL Server username for connecting to the repository. The NQKeyGenWindows.exe program prompts for the password to use for the SQL Server account

2. Specify a path for the file that you can access from the agent computers. You can then use the AMAdmin_AgentConfigSecurityKey Knowledge Script to distribute the agent key file to all of your Windows agents. For more information, see "Updating the Key File for Windows Agents" on page 18.

Updating the Key File for Windows Agents

For maximum security and to prevent keys from being compromised over time, periodically create new keys and distribute new key files to all Windows agent computers. This process, called "re-keying," applies when you are using Security Level 1 or 2 (encryption or management server authentication and encryption).

When changing the agent key file, update all of the agent computers before updating the management servers. And keep in mind that all management servers and Windows-based agent computers within a management site must use the same security level and the same key file.

Because re-keying requires you to restart all of your management servers, you should plan for re-keying carefully. If you cannot update the key file for some agents, you will experience communication failures between the management server and those agents. In addition, anytime you update the key file, you may experience a temporary loss of communication between the management server and the agents. Therefore, consider disabling communication with some agents before updating key files or security.

To replace the agent key file:

1. Generate a new key and store the key information in the repository using the `NQKeyGenWindows.exe` utility.
2. Extract the agent portion of the key and save it to a file location using the `NQKeyGenWindows.exe` utility.
3. Use the Control Center console to run the **AgentConfigSecurityKey** Knowledge Script on the agent computers you want to update. When creating the job, click the **Values** tab and:
   - Type the path to the new agent key file in the **Location of key file** field.
   - Type the password for the new agent key file in the **Encryption password** field.
   - Set the event notification and severity parameters.

4. Click **OK** to start the job.
5. Verify that all jobs complete successfully.
6. After updating all of your Windows agents, manually stop and restart each management server in the management site by stopping and then restarting the NetIQ Corporation AppManager Management Service (`NetIQms`).

Using Secure Communication for UNIX Agents

AppManager has three security levels for controlling the communication between a management server and UNIX-based agent computers:

- Clear Text (Security Level 0)
- Encryption (Security Level 1)
- Authentication and encryption (Security Level 2)

For more information, see "Understanding Communication Security Levels" on page 11

Security Level 0 provides no security at all. For Security Levels 1 and 2, the management server and UNIX agent use a Secure Socket Layer (SSL) protocol to secure their communications. The SSL protocol is a widely used standard for implementing encrypted network communication and authentication.

If you use Security Level 1 (encryption only), AppManager does not authenticate the management server. For both security levels, AppManager uses a symmetric encryption algorithm to encrypt and decrypt the data being transferred. The cipher suite used is `SSL_RSA_WITH_RC4_128_SHA`. The cryptographic library used is the Open Secure Sockets Layer Library (OpenSSL). This library can be exported outside of the United States, except where expressly restricted.

Although you normally set the security level for a management site during installation, you can change the security level after installation by using the `NQKeyGenUnix.exe` utility. You can also use the `NQKeyGenUnix.exe` utility any time you need to create and manage public/private key pairs and key files if you are using authentication and encryption.

For more information, see "Changing the Security Level for Management Servers" on page 14.

Changing the Security Level for Management Servers

After installation, you can use the `NQKeyGenUNIX.exe` program to change the security level for communication between the management server and UNIX agents.

To change the security level for the management server:

1. On a management server computer, open a Command Prompt window and change to the `NetIQ\AppManager\bin` directory.
2. Run `NQKeyGenUnix.exe` to set the security level for the management server using the following command:

   `NQKeyGenUnix -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`* `-seclev` *`level`*

   For example, to log into the QDB on the computer named `NYC2003` using your current Windows account name and password and set the security level to use encryption only (`1`), type a command similar to this:

   ```
   NQKeyGenUnix -db qdb::nyc2003 -seclev 1
   ```

   Note
   All management servers that connect to the same repository must use the same security level, and, if using management server authentication, the same key pair.

3. Stop and restart the NetIQ AppManager Management Service (`NetIQms`) to have the management server receive the new security level information.

Changing the Security Level for UNIX Agents

If you change the security level for the management server, update the security setting for every UNIX agent. You can change the security setting for an agent from the UNIX Agent Manager console. For information about UNIX Agent Manager, see the NetIQ UNIX Agent documentation, which is included in t he AppManager UNIX download package.

Generating a Public/Private Key Pair for UNIX Agents

If you are using Security Level 2 (management server authentication and encryption) to secure communications between the management server and UNIX agent computers, generate a new public/private encryption key pair.

To generate a new public-private encryption key pair:

1. On a management server computer, open a Command Prompt window and change to the `NetIQ\AppManager\bin` directory.
2. Run the `NQKeyGenUnix.exe` program to generate a new public/private key pair and store the key information in the repository using the following command:

   `NQKeyGenUnix -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`* `-new`

Variable	Description
`database_name`	The name of the AppManager repository.
`user_name`	A valid SQL Server login with permission to access the AppManager repository. **Note** If you are using Windows authentication to connect to the repository, leave the username blank. If you are using SQL Server authentication, type a SQL Server username for connecting to the repository. The program prompts for the password to use for the SQL Server account.
`sql_server`	The name of the SQL Server computer where the AppManager repository is installed.

   For example, to use Windows authentication with a QDB installed on the server `NYC2003`, you would type a command similar to this:

   ```
   NQKeyGenUnix -db qdb::nyc2003 -new
   ```

   Note
   If you attempt to generate a new key pair when a key pair already exists in the repository, the `NQKeyGenUnix.exe` program issues a warning. If you continue, the existing key pair becomes inactive and is added to a historical listing of keys, and the new key pair is activated. You can remove these older keys, as needed. For more information, see "Key File Utility for UNIX Agents" on page 214.

3. Type a password for the public/private key pair.
4. Run `NQKeyGenUnix.exe` to extract the public portion of the key for the UNIX agents to use with the following command:

   `NQKeyGenUnix -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`* `-ckey` *`filelocation`*

   In specifying a path for the file, use a directory that you can access from the UNIX computers to be managed.
5. Copy the public key file to a network location accessible from the UNIX computers to be managed.
6. Stop and restart the NetIQ AppManager Management Service (`NetIQms`) to register the new key pair with the management server.

Extracting and Sharing Key Information from the Repository

The `NQKeyGenUnix.exe` program can extract the public/private key information and save it in a password-protected file. Saving the information in a file allows you to share a single key pair across multiple repositories.

If you want to create this password-protected file, run the `NQKeyGenWindows.exe` program using the following command:

`NQKeyGenWindows -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`* `-skey` *`filelocation`*

Note
If you are using Windows authentication to connect to the repository, leave the username blank. To use SQL Server authentication, type a SQL Server username for connecting to the repository. The `NQKeyGenWindows.exe` program prompts for the password to use for the SQL Server account.

To check the key into another repository from the file location specified:

1. On a management server computer, open a Command Prompt window and change to the `NetIQ\AppManager\bin` directory.
2. Run the `NQKeyGenUnix.exe` program to import the key pair into the repository.

   For example, if you created the key using the password `^myPass` and extracted the encrypted key to the file location `C:\Security\AMkey.txt`, type the following command to import the key pair into the repository `QDB01` on the computer `SFO2003`:

   ```
   NQKeyGenUnix -db QDB01:smithj:SFO2003 -change C:\Security\AMkey.txt
   ```

3. Use the password you used to create the key in the repository. Provide the key file password `^myPass`.
4. After you import the key, stop and restart the NetIQ AppManager Management Service (`NetIQms`) to register the new key pair with the management server.

Extracting the Public Key for UNIX Agents

The `NQKeyGenUnix.exe` program can extract the public encryption key information and save it in a file, which you can distribute to all of your UNIX agents. To extract the public portion of the key for the agents to use, run the `NQKeyGenUnix.exe` program using the following command:

`NQKeyGenUnix -db` *`database_name`*`:`*`user_name`*`:`*`sql_server`* `-ckey` *`filelocation`*

In specifying a path for the file, use a directory that you can access from the UNIX computers to be managed.

If you change the public key file, you must run the AMAdminUNIX_AgentUpdateSecurityLevel Knowledge Script, replace the old public key file in the UNIX agent `data` directory with the new public key file, and restart the UNIX agent. For more information about changing the security settings for the UNIX agent, see the *AppManager for UNIX Servers Management Guide*.

Updating the Public/Private Key Pair for UNIX Agents

For maximum security and to prevent keys from being compromised over time, periodically you should create new public/private key pairs and distribute the new public portion of the key to all of your UNIX agent computers. This process, called "re-keying," only applies when you are using server-side authentication security. You do not need to re-key if you are only using Security Level 1 (encryption).

Notes

- If you have more than one management server in your management site, the management server acting as the current primary management server for the agent must complete the re-keying process. If communication with the acting primary management server is interrupted before re-keying is complete, failover to the inactive management server will not take place and communication with the UNIX agent will be lost.
- To prevent this problem, check the status of all management servers and ensure the agent can communicate with the management server before you start re-key operations. And never stop the UNIX agent management server during re-key operations.

To replace the public/private key pair:

1. Create a new public/private key pair and store it in the repository using the `NQKeyGenUnix` utility.
2. Stop and restart the management server so it picks up the newly created key pair.

After you restart the management server, the next communication from the UNIX agent fails when the agent attempts to authenticate the management server using the old public key. The UNIX agent then uses an encrypted message to request the new public key from the management server by sending a message that includes a checksum for its current key.

The management server uses the checksum to retrieve the key pair from the repository and to encrypt the new public key with the previous private key, and then it sends the signature and the new encrypted public key back to the UNIX agent. The UNIX agent decrypts the new public key using its old public key, which verifies the new key has come from the appropriate management server and begins using the new public key.

You can remove historical keys from the repository using the `NQKeyGenUnix` utility at any time. If you remove the historical keys, however, you will need to manually replace the public key file on each UNIX agent when you change the public/private key pair. In this case the automated re-keying process described above fails.

Setting up Primary and Backup Management Servers

Within each management site, you can explicitly designate a **primary management server** and a backup, or **secondary management server** for each agent computer. Establishing a primary and secondary management server for each agent computer provides the following benefits:

- Predictable **failover support**
- Static **load distribution**

You can designate the primary and secondary management server when you install the AppManager agent. You can also run the AMAdmin_SetPrimaryMS Knowledge Script for Windows computers and the AMAdminUNIX_SetPrimaryMS Knowledge Script for UNIX computers after installation.

After you explicitly designate a primary management server, only the primary management server sends job requests to the agent and receives corresponding events and data. If communication with the primary management server is interrupted and you have identified a secondary or backup management server, communication is automatically transferred to the secondary management server. If you have not specified a secondary management server, data and events are stored locally on the agent computer. When communication with the primary management server is restored, the agent then resumes communication with the management server.

Note

For UNIX-based agents, the management server you identify during installation becomes your default primary management server. The installation steps also prompt you to specify whether the UNIX agent can also communicate with other management servers. You can then run the AMAdminUNIX_SetPrimaryMS Knowledge Script to designate a secondary management server.

If you have multiple management servers in your environment, NetIQ Corporation recommends the following:

- Designate the local management server as the primary management server. For more information, see "Designating the Local Management Server as the Primary Management Server" on page 24.
- Configure a single management server for remote installation tasks. This configuration avoids excessive network traffic associated with remote agent installation or upgrade tasks.
- Designate primary and secondary management servers for all agent computers. For more information, see "Configuring a Primary and Secondary Management Server for Windows Agent Computers" on page 25 and "Configuring a Primary and Secondary Management Server for UNIX Agent Computers" on page 25.

Designating the Local Management Server as the Primary Management Server

When you install the management server component, an AppManager agent is automatically installed locally on the management server. Configure the local agent to have the local management server designated as its primary management server.

To designate the primary management server for the agent on the management server:

1. Log on to the Control Center console as a user who has permissions to run Knowledge Scripts.
2. If you have not already done so, run the Discovery_NT Knowledge Script and any additional Discovery Knowledge Scripts on each management server computer. For example, if you have installed the management server component on the computer `JUNO`, drag and drop the Discovery_NT Knowledge Script onto `JUNO` in the Enterprise Layout pane of the console.
3. After a successful discovery, run the SetPrimaryMS Knowledge Script on the management server computer.
4. On the **Values** tab, specify the local management server as the primary management server for this computer. You do not need to configure a secondary management server for this computer.

Configuring a Primary and Secondary Management Server for Windows Agent Computers

Each agent computer can have only one designated primary management server and one designated secondary management server. Unless you designate the primary and secondary management server during agent installation, you need to perform this task manually after you:

- Configure each management server to be its own primary management server
- Configure a single management server to perform installation-related tasks

To designate a primary and secondary management server for each agent computer:

1. Log on to the Control Center console with an account that is a member of the Administrator user group.
2. Make sure all agent computers have been discovered.
3. Run the SetPrimaryMS Knowledge Script on the agent computer.
4. On the **Values** tab, specify the primary and secondary management servers for this computer.
5. Set the **management server operation to perform** to designate both the primary and secondary management server for this agent computer, then click **OK**.

It can take up to 15 minutes for the QDB to designate the primary and secondary management servers. To start new jobs after you change the designation for an agent computer, wait until the repository updates the management server designation.

Once the management server designation is complete, you can run jobs on an agent computer. The designated primary management server sends job requests.

Configuring a Primary and Secondary Management Server for UNIX Agent Computers

When you install the UNIX agent, you specify the management server you want the UNIX agent to communicate with. Because you explicitly designate the management server during installation, that management server becomes the default primary management server for the agent computer. After installation, you can use the AMAdminUNIX_SetPrimaryMS Knowledge Script to designate a secondary management server or to change the primary management server.

To designate the secondary management server or change the primary management server for a UNIX agent computer after installation:

1. Log on to the Control Center console as an account that is a member of the Administrator user group.
2. Make sure all agent computers have been discovered.
3. In the Enterprise Layout pane, select the Knowledge Scripts view for the management group containing the UNIX agent computers you want to configure.
4. In the View pane, expand the AMAdminUNIX category.
5. Drag and drop the SetPrimaryMS Knowledge Script onto the agent computer.
6. On the **Values** tab, type the name or IP address for the management server you want to designate as a primary or secondary management server.

7. Select either **Set primary management server** or **Set secondary management server** to designate a new primary or secondary management server for this agent computer.

   Note
   You can only perform one management server operation at a time with this Knowledge Script. Therefore, to change both the primary management server and the secondary management server, you need to run this Knowledge Script twice. By default, if you use this Knowledge Script to specify a new primary management server, the management server you specified when you installed the UNIX agent becomes the secondary management server. You can use the Unset specified management server option to remove a specific management server you no longer want to use as either a primary or secondary management server.

8. If you want to change the port number the UNIX agents use to communicate with the management server, you can type a new port number for the **Port number for the management server** parameter. For information about setting a new port on the management server, see "Changing the Listening Ports" on page 142.

   Note
   If you change the port number of the management server, you can use this Knowledge Script to update this setting for your UNIX agents. After running the job, restart the management server to restore communication using the new port number.

9. Click **OK** to run the job.

It can take up to 15 minutes for the QDB to designate the primary and secondary management servers. To start new jobs after you change the designation for an agent computer, wait until the repository updates the management server designation.

When the management server designation is complete, you can run jobs on the agent computers. Job requests will be sent from the designated primary management server.

How Failover Works

After you have designated a primary and secondary management server for an agent computer, the AppManager repository and the local repository on the AppManager agent store the server information. The agent accepts job requests from and sends events and data to its primary management server.

If an attempt to communicate with the primary management server fails, the agent waits for one minute before attempting to reconnect to the primary management server. By default, the agent attempts to connect three times every minute before failing over to the secondary management server.

After the third attempt to connect to the primary management server fails, the agent sends events and data to the secondary management server to store in the AppManager repository. However the secondary management server does not immediately send new or changed job requests to the agent computer.

Every 15 minutes, the management server updates its list of the agent computers. If a secondary management server picks up communication with new agent computers because communication with a primary management server fails, the secondary management server updates itself with that information after the interval. If there are any new jobs or changes to job properties, the secondary management server can then communicate these changes to the agent computers that have failed over. Because of this delay before the secondary management server recognizes the failed-over agent computer, it can take up to 15 minutes to communicate any job information to the agent computer.

While the agent computer is managed by the secondary management server, the agent continues trying to contact its primary management server every minute. After the agent is able to re-establish communication with the primary management server, the agent sends events and data to its primary management server. Each management server updates its list of agent computers and the communication of job information is transferred back to the primary management server a minute later.

For information about modifying the interval and the number of times the agent attempts to contact the primary management server, see "Changing Agent Failover Configuration" on page 156.

For information about changing the interval at which a management server checks its list of agent computers, see "Changing the Polling Interval for Agent Computers" on page 141.

Distributing Processing Load

You can distribute processing load by assigning agent computers to different management servers. For example, you can assign agent computers in New Zealand to a management server in New Zealand and agent computers in Sweden to a management server in Sweden. Or you can assign agent computers to management servers according to functional groups or departments.

If you plan to use multiple management servers to balance processing load, limit the number of computers each management server is responsible for monitoring. For example, if you have two management servers, `LONDON` and `PARIS`, you should configure half of your agent computers to use the `LONDON` management server as the primary management server and `PARIS` as the secondary management server, and half of your agent computers to use `PARIS` as the primary management server and `LONDON` as the secondary management server. The primary/secondary pair should not manage more than approximately 600 agent computers between them which means each management server can manage up to 300 agent computers.

This configuration ensures that no single management server is responsible for more agent computers than it can handle in the event of a failover.

Verifying the Management Server Assigned to an Agent Computer

You can verify if a management server that is assigned to an agent computer is the primary management server or the secondary management server.

To verify the management server assigned to an agent computer:

1. Click the agent computer in the Enterprise Layout pane of the Control Center console.
2. The View pane displays both the primary as well as any secondary management server for the agent computer.

Changing a Management Server Assigned to an Agent Computer

There are a variety of reasons you may want to change the management server assigned to an agent computer. The management server may not be performing as expected, or if an agent computer is moved to another network or geographical location you may want to change its primary or secondary management server. For more information about changing the management server, see "Configuring a Primary and Secondary Management Server for Windows Agent Computers" on page 25 and "Configuring a Primary and Secondary Management Server for UNIX Agent Computers" on page 25.

Adding a New Management Server

You may want to add new management servers to your AppManager management site. Before adding a new management server, determine whether to use the new management server as a passive, secondary management server for all agent computers or as a primary management server for some of your agent computers.

If you plan to use the new management server as a primary management server for some agent computers to balance processing load, you should plan to configure 50% of the agent computers to use the first management server as the primary and to use the new management server as the secondary management server and 50% of the agent computers to use the new management server as the primary and to use the first management server as the secondary management server.

To add a management server to your management site:

1. ***If you have not already explicitly designated the management server*** for every agent computer, run the AMAdmin_SetPrimaryMS Knowledge Script on all agent computers to configure the current management server as the primary management server.
2. Install the new management server and configure it to communicate with the QDB.
3. Run AMAdmin_SetPrimaryMS on the new management server computer and designate the local management server as the primary management server for the local agent. For more information, see "Designating the Local Management Server as the Primary Management Server" on page 24.
4. Configure each agent computer to recognize the new management server as follows:
   - To configure a Windows agent computer, run the AMAdmin_SetPrimaryMS Knowledge Script.
   - To configure a UNIX agent computer, run the AMAdminUNIX_SetPrimaryMS Knowledge Script.

It can take up to 15 minutes for the management server to identify any changes to its list of agent computers. Therefore, the agent computer can begin sending information to a newly designated management server immediately, but there may be a delay of approximately 15 minutes before a newly designated management server can begin sending new job information to the agent computer.

Chapter 3
Managing Security for AppManager and Control Center

Configuring who has access to the AppManager Operator Console and the Control Center console and defining the tasks each user can perform is the primary way you ensure security and enforce job-specific access to AppManager. You configure security settings for the Operator Console using Security Manager. For the Control Center console, you configure security settings using the Manage Security option on the Global Tasks tab of the Ribbon.

This chapter explains how AppManager works with SQL Server and either Windows Authentication or Mixed Mode security to define basic security for AppManager. This chapter also describes:

- How to use Security Manager to define role-based security for AppManager Operator Console users, add AppManager user accounts based on SQL Server login accounts, assign AppManager users to the roles you create, and manage user rights.
- How to use the Control Center console to define group-based security for Control Center users, add Control Center user accounts based on SQL Server login accounts, assign Control Center users to the groups you create, and manage user rights

Understanding User Security

Since both the AppManager repository and Control Center repository are SQL Server databases, AppManager security relies on SQL Server security. Every user who needs access to the Operator Console or Control Center console must have a valid SQL Server login name and password for the SQL Server where the AppManager repository or the Control Center repository database is running. The creation and authentication of the SQ Server login accounts at connection time depends on the SQL Server security mode you use.

Before you create any Operator Console or Control Center console users, determine the SQL Server security mode you are using. You can configure SQL Server to use one of the following security modes:

- **Windows Authentication** security, which links SQL Server login accounts with Windows user accounts and uses Windows account authentication to validate SQL Server logins for all connections.
- **Mixed Mode** security, which allows SQL Server login accounts to be independent of any Windows user or group account. SQL Server login requests can be validated using either Windows authentication or SQL Server internal authentication.

Control Center console users also need access to the QDBs that connect to the Control Center repository. Regardless of the authentication method, Control Center users cannot access a QDB if they are not added as a user in the QDB.

By default, Control Center console users are granted Read Only permissions for every QDB you register them with in the Control Center console. However, depending on the tasks a user needs to perform in the Control Center console, you may need to modify the permissions the user has on the primary QDB. You need to make these modifications using Security Manager. For more information about defining security for the Control Center console, see "Managing Control Center Security" on page 47. For more information about Operator Console security, see "Managing Operator Console Security" on page 31.

Using Windows Authentication Security

If you use Windows-only authentication, use Windows administrative tools to create and manage user and group accounts and then use the Control Center console to map those groups and users to SQL Server logins. An SQL account can be a Windows group or user.

With this security mode, users log on to the AppManager or Control Center repository using their Windows domain, user account name, and password and have only the permissions associated with that account as defined by the Operator Console or Control Center console.

For more information see "Adding a Control Center User" on page 48.

Using Mixed Mode Security

If you are using Mixed Mode security, you can create and maintain SQL Server login accounts independently of any Windows accounts or groups. With this mode, you can manage login accounts through SQL Server and authorize which accounts should have access to the AppManager Operator Console or Control Center console. You can also create new SQL Server logins with access to an AppManager repository or Control Center repository using the AppManager Security Manager or the Control Center console.

Note
You must be an SQL Server administrator to create SQL Server logins using Security Manager or the Control Center console.

With mixed mode security, users can log on to an AppManager or Control Center repository using Windows authentication or SQL Server authentication. If you are using mixed mode security, you need to inform users whether they should use Windows Authentication or SQL Server Authentication to log on to an AppManager or Control Center repository, depending on how you have configured the account.

Managing Users with Windows Groups

In addition to understanding SQL Server security modes, you should also consider using Windows groups to manage user accounts most effectively. You can create groups using standard Windows administrative tools, then map an entire group to a single SQL Server login. When you have created the SQL Server login for the group, all privileges assigned to that login through SQL Server and AppManager apply to all of the member user accounts within that Windows group.

Once you grant the SQL Server login account permission to access the AppManager repository, you can use Security Manager or the Control Center console to add the group account as a new AppManager user.

Although it is common for a user to belong to more than one Windows group, you should avoid this when using Windows groups for AppManager users. If a user belongs to more than one Windows group that is mapped to a SQL Server login account and added to AppManager, maintaining security can become difficult. For example, if the user `SPeters` belongs to two Windows groups, `ExchAdmins` and `JrAdmins`, that have been given different privileges or assigned different AppManager roles, the user may have unexpected or conflicting rights or restrictions.

The best way to ensure consistency and manageability is to create new Windows groups specifically for each Security Manager role or Control Center user group you plan to define. Using Security Manager, you can specify the individual functional rights for viewing information and performing tasks you want available for each role. For example, if there are two AppManager roles available, `Read-Only User` and `SrAdmin`, you can create two corresponding Windows groups called `AppManager ReadOnly` and `AppManager Senior Admins` and assign the corresponding role to each group of users. Using the Control Center console, you can define user groups and then add the Windows groups to the Control Center user groups. You then assign user groups and permission sets to management groups to define security for the Control Center console. For more information about managing security using the Control Center console, see "Managing Control Center Security" on page 47 and "Adding a Control Center User" on page 48.

Note
In creating Windows user accounts and groups to access AppManager, you need to consider that specific privileges may be required to perform certain tasks. For example, any Windows user account or group that is used to log on to the Operator Console must be granted Write permission for the `NetIQ\AppManager\bin\cache` folder.

Managing Operator Console Security

Use the Security Manager to configure security for a QDB. This section describes how to use the Security Manager.

Security Manager enables AppManager administrators to control access to views and tasks in the Operator Console. Depending on your access rights and your SQL Server security setting, you can use Security Manager to identify SQL Server logins to use the Operator Console, add new SQL Server logins, assign roles to Operator Console users, and manage user rights.

You can use SQL Server Management Studio to identify at least one Windows or SQL Server account for logging in to a QDB the first time, and then log onto Security Manager with this SQL Server login grant other SQL Server login accounts access to the Operator Console.

Starting Security Manager

To start Security Manager from the Operator Console, click **Extensions > Security Manager**. When you start Security Manager from the Operator Console, you are automatically connected to the QDB you specified when you started the Operator Console.

To use the Security Manager for the first time, you must have at least one SQL Server login with permission to access SQL Server and the AppManager repository. Use SQL Server Management Studio to create this SQL Server login.

To start Security Manager for the first time:

1. Start Security Manager from the **NetIQ AppManager > Tools & Utilities** program group.
2. In the Security Manager Logon dialog box, enter the following information:

Field	Description
Server	The name of the Windows server where the AppManager repository is installed. After you enter the name, the repositories available on that server are displayed in the Repository list.
Repository	The database name for the AppManager repository you want to work with. The default repository name for AppManager is `QDB`. **Note** Once you have logged into the Security Manager, you can switch to another repository.
Connection Information	**Use Windows authentication** — Log on using your current Windows user name and password. You must use this type of connection if SQL Server uses Windows Authentication security. **Use SQL Server authentication** — Log on to SQL Server by typing the **Login name** and **Password**. **Notes** • If you are running SQL Server in Windows Authentication mode, you will not see this option. • When using a SQL user account, make sure the password for the user account is less than 32 characters. If your password exceeds 32 characters, Security Manager displays an error message. • If FIPS is enabled on the operating system, you cannot log on to SQL Server using SQL Server authentication. You must use Windows authentication.

3. Click **Logon** to open Security Manager.

Connecting to Another Repository

You can use Security Manager to administer only one AppManager repository at a time. However, you can easily connect to another AppManager repository from Security Manager.

To connect to another QDB:

1. On the Security menu, click **Connect Repository > New**.

   You can also choose recently used repositories from the bottom of the **Connect Repository** submenu.

2. Select the AppManager repository server and database you want to connect to as well as a security mode.

   - If you are connecting using **Windows authentication**, click **Logon**.
   - If you are connecting using SQL Server authentication, type your SQL Server login and password, then click **Logon**.

3. If you are able to successfully log on to the repository, Security Manager disconnects from the current repository and connects to the new one.

Adding Operator Console Users

Operator Console users are SQL Server logins with permissions to access a QDB. You must configure each QDB to recognize the users and allow them to log into the Operator Console.

How you add Operator Console users and configure QDBs depends on the SQL Server security mode you are using:

- ***If you are using Windows Authentication security,*** use the SQL Server Management Studio to add Windows groups or users as valid SQL Server logins on a QDB. Then use the Security Manager to identify the new SQL Server logins as Operator Console users.
- ***If you are using Mixed mode security***, you can create the SQL Server logins with Security Manager at the same time you are creating the Operator Console user account. You can also add SQL Server logins using SQL Server Management Studio and then identify these logins as Operator Console users with the Security Manager.

Granting a User or Group Permissions to a QDB

If you are using Windows Authentication, you first need to create new SQL Server logins based on Windows users and groups using SQL Server Management Studio. You must then identify the new SQL Server logins as Operator Console users using the Security Manager.

To add new SQL Server logins in SQL Server Management Studio:

1. Start the SQL Server Management Studio on the SQL Server computer.
2. Expand the server where the database resides, and then expand the **Databases** folder.
3. Expand the **Security** folder.

4. Right-click the **Logins** folder and select **New Login**.

   Configure the following properties:

Node	Properties
General	• Type the name of the Windows user or group or click **Search** to locate the user or group in the domain.
Server Roles	• Select the `public` option.
User Mapping	• Select the QDB to grant access to the repository database. • Select the `public` and `AM_public` database roles for the repository.
Status	• Select the **Grant** option under **Status** to grant the user permission to access the database engine. • Select the **Enabled** option under **Login** to enable the user account.

   For more information about using the SQL Server Management Studio to create SQL Server login accounts and using Windows Authentication, see your SQL Server documentation.

5. Click **OK** to add the login account.

Identifying SQL Server Users as Operator Console Users

After you add a SQL login to a QDB, you can give the login permission to log on to the QDB with an assigned Operator Console role. You identify SQL Server logins as Operator Console users in Security Manager.

To identify SQL Server logins as Operator Console users:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Log on with an account that belongs to the AppManager Administrator role.
3. From the Security menu, select **User Setup**.
   - The **SQL users** list displays the existing SQL Server logins that have access to the AppManager repository database using the SQL Server Management Studio.
   - The **AppManager users** list shows the SQL Server logins that you have authorized to log on to AppManager.
4. Select a SQL Server login from the SQL users list, then click **Add** to move the login to the **AppManager users** list. AppManager assigns the user to the default role. For information about setting the default role, see "Selecting the Default Role" on page 40.

   If you want to assign a different role for any user, click in the **Security Role** column and select an appropriate role from the list.

   Note
   Server Roles defined in SQL Server Management Studio take precedence over Operator Console roles. Therefore, if you add SQL Server users who have System Administrator or Server Administrator server roles, those users' rights are not restricted by their Operator Console role and those users will be able to perform virtually any Operator Console task. If you are assigning SQL Server users to a Standard User, Read-Only User, or custom role, verify that the user account has not been assigned a server role with broad permissions. For more information about server roles, see the Microsoft SQL Server documentation.

5. Click **Close** when you are finished adding users and groups and setting roles.

Creating a SQL Server User in Security Manager

If you create a new SQL Server login from Security Manager, AppManager automatically assigns the new login as an Operator Console user and configures the account to use SQL Server authentication.

Notes

- You cannot create a SQL Server login that uses Windows Authentication through Security Manager. If you are using Mixed mode security and want to create SQL Server logins that use Windows Authentication, you must create the accounts with SQL Server Management Studio.
- If you have enabled FIPs compliant security in the Control Center Console, you cannot create SQL Server logins in Security Manager. You can create new logins using SQL Server Management Studio or the Control Center Console.

To add new SQL Server logins with Security Manager:

1. Start Security Manager and log on with an AppManager administrator account.
2. From the Security menu, select **User Setup**.
3. Click **New SQL User**.
4. Specify a SQL Server username, SQL Server group, and login password for the user you want to create, then click **OK**.

Field	Description
SQL user name	A username for the account. The maximum length you can specify for a user name is 29 characters. If the user name is too long, it is truncated and you cannot log into the Operator Console. **Note**: • You cannot specify login names with certain special characters. These characters include: \ / * ? : < > | “ • You can specify a case-sensitive user name in a case-sensitive SQL Server environment.
SQL group	A valid SQL group for the user. The default SQL Server group is `public`, but your organization may have additional groups.
SQL login password	The SQL login password for the new SQL Server login. The maximum length you can specify for a password is 32 characters. If the password is too long, it is truncated and you cannot log into the Operator Console.

Managing Operator Console User Accounts

Most organizations start with a few administrative users and add specialized role-based users over time. In general, once you have created the roles and security profiles appropriate to your organization, there is very little account maintenance required for managing user accounts. There are a few common tasks, however, that you may need to perform.

Operator Console Roles

Operator Console roles enable you to define what different groups of users can see and do within the Operator Console. For example, you may want to prevent some users from starting and stopping jobs, closing events, or changing job properties.

For each AppManager role, you define the specific rights you want the users in that role to have. This collection of rights associated with an Operator Console role is called a **security profile**. Each time you add a new user, you select the appropriate Operator Console role for that user to establish what information that user can view and what tasks that user can perform.

Note
An Operator Console user or group can belong to only one Operator Console role.

The rights you can set for Operator Console roles include:

- Access to basic AppManager functions, such as whether users can run jobs, acknowledge and close events, or modify the TreeView
- Permission to start AppManager console programs such as the Chart Console and the Repository Browser
- Access to the different views in the Operator Console and Operator Web Console
- Access to advanced AppManager operations, such as Knowledge Script property propagation, the ability to modify monitoring policies, or permission to put a computer in maintenance mode

Security Manager includes three **predefined roles** that you can modify to suit your needs. You can also create your own custom roles.

In general, you should use roles to restrict access to Operator Console features and capabilities. Initially, you should allow only site administrators or expert-level administrators to perform most tasks and you should limit access to the Operator Console to a small number of people until you have firmly established site policies and role definitions that suit your organization. Once your production environment is stable and your threshold settings, job properties, event-handling, and data-handling policies have been refined to meet your organization's needs, you may want to grant more operators and administrators access to the Operator Console.

Understanding Predefined Operator Console Roles

AppManager roles control what users can do when they work with the Operator Console. Every Operator Console user must be assigned a role. To help you get started, the Operator Console provides the following predefined roles:

Role	Default Rights
Administrator	All functional rights to perform all Operator Console activities and see all views. This role can be copied but not modified, deleted, or renamed. Because you cannot modify this role, only the Users tab is available in the Properties pane when you select the Administrator role.
Read-Only User	Functional rights to start the Operator Console or Control Center console and see all views but not perform any AppManager activities. This role can be copied, modified, deleted, or renamed.
Standard User	Functional rights to perform all basic Operator Console and Chart Console activities and to see the Master view. This role can be copied, modified, deleted, or renamed.

If you plan to use the predefined roles with the default functional rights and views, you can begin adding Operator Console users and assigning those users to these predefined roles. For information about adding new Operator Console users, see "Adding Operator Console Users" on page 33.

Most organizations, however, find it useful to modify the predefined roles or create custom roles before adding any Operator Console users.

Modifying a Predefined Role

If you select the Standard User or Read-Only User role, the Functional Rights, Views, Exceptions, Users, and Knowledge Scripts tabs are displayed in the Properties pane. You can edit the functional rights, limit the views available, or define computer-based exceptions for these predefined roles as needed.

To change rights for the predefined role:

1. Start Security Manager and select the role that you want to modify in the Security pane.
2. Click the **Functional Rights** tab.
3. Define the functional rights for the role. For more information, see "Defining Functional Rights for a Role" on page 43.
4. Click **Apply**.

You can also copy any of the predefined roles to create a new custom role and modify that role to suit your needs. For more information about defining roles and setting functional rights, see "Understanding Custom Roles" on page 38 and "Modifying the Security Profile for a Role" on page 40.

Regenerating Predefined Roles

If you delete any of the predefined roles, you can regenerate the role with its default rights by clicking **Generate Predefined Roles** on the Security menu. You can then modify the rights for the regenerated roles. If you did not delete a predefined role, any changes you made to a predefined role are preserved.

Understanding Custom Roles

You can create custom roles to use along with the predefined roles, or you can create custom roles as an alternative to the predefined roles. In most cases, the custom roles reflect the types of activities that a specific group of AppManager users perform. For example, you can set up a custom role for your Exchange administrators that gives them access only to the Exchange view or create a custom role for managers who only want to create and view charts in the Chart Console.

The users assigned to a role have all of the rights and restrictions defined in the security profile for that role. It is important to keep this in mind when defining the security profile because each user can only be assigned to one role. Therefore, many organizations find it useful to create several specialized custom roles.

Note
Server Roles defined in SQL Server Management Studio take precedence over Operator Console roles. Therefore, if you add SQL Server users who have System Administrator or Server Administrator server roles, those users' rights are not restricted by their Operator Console role.

Adding a New Custom Role

You can create custom roles by adding a new role or by copying an existing role. By default, adding a new role creates a role with complete access to all AppManager views and with all functional rights enabled for all AppManager components.

When you create custom AppManager roles, you need to:

- Specify a role name.
- Specify the security profile for the role by setting the functional rights and exceptions for the role.
- Assign AppManager users to the role.

To add custom roles:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. On the Security menu, click **Add Role**.
3. Type a **Role name**, then click **OK**. The new role is added to the list of AppManager Roles in the Security pane.

In most cases, after creating the role, you need to modify its security profile to restrict some operations or to disable access to some views or components before assigning any users to that role.

- For information about setting functional rights, see "Defining Functional Rights for a Role" on page 43.
- For information about assigning roles to new AppManager users, see "Changing User Roles" on page 40.
- For information about copying, renaming, and deleting roles, see the online help.

Tip
If you need to create a role that is similar to an existing role, copy the existing role, and then change its properties. For more information, see "Copying an Existing Role to Create a New Role" on page 39.

Copying an Existing Role to Create a New Role

To create new Operator Console roles, you may want to use existing roles as templates. You can do this by first copying a role, and then creating a new role based on the copy.

To copy a role and create a new role based on the copy:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Expand the list of **AppManager Roles**.
3. Select the role you want to copy.
4. On the Security menu, click **Copy Role**.
5. Type a **Role name** for the new role, then click **OK**. The new role is added to the list of AppManager Roles in the Security pane.

Renaming a Role

Operator Console roles, both predefined and custom, can be renamed. However, the Administrator role cannot be renamed.

To rename a predefined or custom Operator Console role:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Expand the list of **AppManager Roles**.
3. Select the role you want to rename.
4. On the Security menu, click **Rename Role**.
5. Type a **New name** for the role, then click **OK**. The renamed role is added to the list of AppManager Roles in the Security pane.

Deleting a Role

Operator Console roles, both predefined and custom, can be deleted. However, the Administrator role cannot be deleted.

To delete any predefined or custom AppManager role:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Expand the list of **AppManager Roles**.
3. Click the role you want to delete.
4. On the Security menu, click **Delete Role**, then click **Yes**.

Selecting the Default Role

Initially, new Operator Console users are assigned the Administrator role by default because administrators are typically the first users added. After you have created or modified the roles you want to use and defined appropriate security profiles for each role, you can modify the default role that is automatically assigned to each new Operator Console user, which simplifies the process of adding new users.

To define the default role:

1. Click **AppManager Roles** in the left pane of Security Manager.
2. In the **AppManager Roles** tab in the Properties pane, choose the desired role from the **Default role** list.

Changing User Roles

You can modify the role for an individual user to change the permissions the user currently has on the Operator Console.

To change the role for a single user:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Log in with an account that belongs to the AppManager Administrator role.
3. Expand the list of **AppManager Users**.
4. Select the user whose role you want to change. Information for the selected user is displayed in the Properties pane.
5. Select a new role from the **Role** list. The views and rights information changes to reflect the new role.
6. Once you have selected the role you want, click **Apply**.

Modifying the Security Profile for a Role

The **security profile** determines the AppManager activities the users assigned to a role can perform. Whether you use the predefined roles or create custom roles, you should review and modify the security profile for each role.

In general, you should prevent most users from performing advanced AppManager activities such as creating monitoring policies, browsing the repository, or performing activities that have site-level implications, such as setting repository preferences.You set restrictions for users by modifying the security profile for a role.

To view or modify the security profile for a role:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Expand the list of **AppManager Roles**.
3. Select the role for which you want to view or enter information.

4. Click the four tabs in the Properties pane to set rights and restrictions for the role.

Tab	Tasks
Functional Rights	Select the specific components that users assigned this role can access, and the specific activities the user can perform using each component. For more information, see "Defining Functional Rights for a Role" on page 43.
Views	Select the specific AppManager views and custom views that users assigned this role can access. For more information, see "Restricting Access to AppManager Views" on page 44.
Exceptions	Select the rights that users assigned this role are not allowed to exercise on specific computers. For more information, see "Setting Computer-Based Exceptions for a Role" on page 44.
Users	See a list of the AppManager users currently assigned this role. You cannot modify the list of users and logins from this tab. You must assign users to a role when you add them as AppManager users. For information about changing a user's role, see "Changing User Roles" on page 40.
Knowledge Scripts	Select the specific Knowledge Script categories that users assigned this role can access. For more information, see "Restricting Access to Knowledge Scripts by Category" on page 45.

When you configure the security profile, users who do not have the functional right to perform an operation cannot access the related menu commands or toolbar buttons. For example, if a user is not permitted to modify Operator Console preferences, the **File > Preferences** command and Preferences toolbar button are both inactive and a **X** icon appears next to the command.

Understanding Access Restrictions

In the Operator Console, **views** organize and filter the information displayed to reflect a specific application or custom properties you define. The default view in the Operator Console is the Master view, which includes information about all of the computers you are managing in your environment and all Knowledge Script categories for discovered computers. All other views, including custom views, limit the information displayed.

By default, the rights you set on the Functional Rights tab are inherited in each view, so that if you have prevented users assigned to a role from starting jobs, they are prevented from starting jobs in each view they are allowed to access. In some cases, however, you may want change the functional rights for a group of users when they are using a specific view. For example, you may want to allow users in the Exchange Admins role to start jobs in the Exchange view but not in any other view.

Functional Rights for the Operator Console

You can set the following rights to control what different users can do in the Operator Console and the Operator Web Console.

Functional Group	Available Rights
Events	Acknowledging, closing and deleting events and adding or changing event comments.
Extensions	Customizing the Extensions menu and launching Extension menu programs.
Graphs	Creating, deleting, exporting, importing and modifying Operator Console graphs and graph properties. These rights only apply to the graphs created in the Operator Console. Rights for the Chart Console are set separately.
Jobs - Existing	Starting, stopping, closing, deleting, and modifying the properties for existing jobs.
Jobs - New	Starting new jobs and setting initial job properties.
Knowledge Scripts	Checking Knowledge Scripts in and out the repository, copying and deleting Knowledge Scripts, modifying and propagating Knowledge Script properties.
Launch AppManager	Starting the Operator Console. This setting controls access to the Operator Console and the Operator Web Console. **Note** You cannot restrict which users can view reports when users access the Operator Web Console. From the Operator Web Console, users view reports by clicking an HTML link rather than through Report Viewer or a custom view.
Modify preferences	Changing preference settings for the AppManager database and console applications. **Note** To configure the **Time interval to purge old points** option under the Repository tab in the Preferences dialog box, the Operator Console user must be logged in as a user who has privileges associated with the System Administrator role.
TreeView	Adding, deleting, or setting maintenance mode for computers in the TreeView. Adding, deleting, or modifying custom properties for objects in the TreeView. Attaching and detaching monitoring policies Adding, deleting, or modifying computer groups in the TreeView. Using Troubleshooter.
Views	Creating, deleting, and renaming custom views.

Rights in Other Consoles

You can set the following rights to control what different users can do in each of the other console programs.

Functional Group	Available Rights
Chart Console	Launching the Chart Console. Viewing and editing shared charts. This right allows the user to view and edit charts that are organized into the Public group of the Chart Console.
Icon Manager	Launching the Icon Manager program.
Repository Browser	Launching the Repository Browser program. **Note** There are security issues in letting users browse through database records. Therefore, you should restrict access to the Repository Browser to the Administrator role or similar roles.
Web Recorder	Launching the Web Recorder.

Defining Functional Rights for a Role

By default, all newly created roles grant all functional rights to the Operator Console and allow users to perform all Operator Console tasks, excluding tasks that are specific to the Control Center console.

To enable or disable the functional rights for any role:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Expand the list of **AppManager Roles**.
3. Select the role whose functional rights you want to define.
4. Click the **Functional Rights** tab.
5. Select **Role has rights only to selected functions** to activate the functional rights list for AppManager console programs.
6. Expand any item to see a list of the rights you can set for that role.
7. Set the functional rights as follows:
   - To allow users in this role to have a functional right, select the functional right.
   - To prevent users in this role from having a functional right, clear the functional right.

   Tip
   NetIQ Corporation recommends that you assign most users to a role with minimal functional rights and that you strictly control which users can:
   - Start and stop jobs
   - Acknowledge, close, and delete events
   - Modify job properties
   - Create and delete custom views
   - Perform other important activities
8. Click **Apply** when you finish modifying rights for the role.

Restricting Access to AppManager Views

Using Security Manager, you can restrict the views a user can access. If a user does not have access to a particular view, the view does not display in the console.

To limit the views users assigned to a role can access:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Expand the list of **AppManager Roles**.
3. Select the role for which you want to define View permissions, and click the **Views** tab.
4. Click **Role has access to only selected views** to activate the list of the currently discovered views and custom views.
5. Select the views that users in this role can access. Clear any view you do not want users to access. If you want to restrict the access to some computers based on a view, be sure to clear the **Master** view check box.
6. ***If you want to use the inherited functional rights*** for all views for this role, click **Apply**. You do not need to perform any additional steps.
7. ***If you want to set view-specific functional rights***, select the view for which you want to set view-based functional rights, then click the **Functional Rights** browse [...] button to define the specific activities a user can perform in the selected view.
8. Click **Override functional rights for this view** to activate the list of functional rights available.
9. Expand any item to see the list of rights you can set.
10. Select the functional rights you want users in this role to have when using the selected view. For information about functional rights for Operator Console views, see "Functional Rights for the Operator Console" on page 42.
11. Click **OK** when you are finished modifying the functional rights for the view.
12. When you have finished modifying the views for the role, click **Apply.**

Setting Computer-Based Exceptions for a Role

By default, the functional rights you set for a role apply to all computers. However, with Security Manager you can set up exceptions on a computer-by-computer basis to fine-tune which rights users in a role can exercise on selected computers.

To define exceptions for a role:

1. Start Security Manager, and expand the list of **AppManager Roles**.
2. Select the role for which you want to define exceptions, and click the **Exceptions** tab.
3. Click **Select Computers**.
4. Select one or more computers from the list of computers and click **OK**. The computers are added to the role's exception list.

5. Select the tasks you want to prevent users from doing on the selected computer. For example, if you do not want users in the current role to put a specific computer into maintenance mode, select **Cannot put this computer into maintenance mode**. If no boxes are checked for a computer, then users in this role have all the rights you have specified as functional rights on the computer.

   Note
   Once you have added a computer to the list of exceptions for a role, you cannot delete the computer from the list. However, you can clear the exceptions you have set to remove any restrictions for that computer.

6. Click **Apply** when you are finished modifying exceptions for computers.

Restricting Access to Knowledge Scripts by Category

In the Operator Console, views organize and filter which Knowledge Script categories are available for running jobs on a discovered resource.

Using the Operator Console's role-based security, you can further restrict the Knowledge Script categories displayed. If a user lacks permission to access a particular Knowledge Script category, the Knowledge Script category does not appear in the Operator Console and the user cannot create new parent jobs from that category.

This role-based permission restricts the user's ability to create new parent jobs, but the restriction does not extend to the user's ability to add new child jobs to an existing parent job or to modify existing jobs.

To limit the Knowledge Script categories users assigned to a role can access:

1. Start Security Manager, and expand the list of **AppManager Roles**.
2. Select the role for which you want to define View permissions, and click the **Knowledge Scripts** tab.
3. Select the **Role has access only to selected Knowledge Scripts** option.
4. Clear each Knowledge Script category that you want to restrict.
5. Click **Apply**.

Viewing a Security Profile

If you are working in the Operator Console, you can click **Help > Security Profile** to see view the security profile of the user.

You can view the security profile for a user, including the following information:

- Assigned role
- User rights
- Views the user can access
- SQL Server login names mapped to the user

To view a user's security profile:

1. Start Security Manager. For more information, see "Starting Security Manager" on page 32.
2. Log in with an account that belongs to the AppManager Administrator role.
3. Expand the list of **AppManager Users**.
4. Select the user whose security profile you want to view.

Disabling an Operator Console User Account

Disabling a user account provides a way to temporarily prevent a user from accessing the Operator Console. When you disable a user account, the account remains displayed in Security Manager and you can re-enable the user account at any time, but the user is not able to log on to the Operator Console.

To disable an Operator Console user account:

1. Start Security Manager.
2. Log in with an account that belongs to the AppManager Administrator role.
3. Expand the **AppManager Users** list.
4. Select the username to disable.
5. Check **Account disabled** to disable the user account.

   To re-enable a user account you have previously disabled, select the username and clear **Account disabled**.

Removing a User Account from Security Manager

Removing a user account prevents a user from having access to the Operator Console and also removes the user from Security Manager. When you remove an AppManager user account with Security Manager, it does not delete the corresponding SQL Server login and username from SQL Server, but you lose any functional rights you have set for the user. If you remove an AppManager user account and need to restore it, you can reset the user's functional rights, as necessary. For more information, see "Defining Functional Rights for a Role" on page 43.

To remove a user account from Security Manager:

1. Start Security Manager.
2. Log in with an account that belongs to the AppManager Administrator role.
3. On the **Security** menu, click **User Setup**.
4. Select users from the AppManager users list, then click **Remove**.
5. Click **Close** when you have finished removing user accounts.

Managing Control Center Security

You use the Control Center console to manage security for the Control Center repository. This section describes how you use the Control Center console to configure security.

The Control Center administrator controls user access to the Control Center console and the operations that users can perform. The administrator configures Control Center security in conjunction with standard Windows and SQL Server login account management. For more information about Windows and Mixed Mode authentication, see "Using Windows Authentication Security" on page 30 and "Using Mixed Mode Security" on page 30.

Note

Members of the Control Center console Administrator group are granted the sysadmin server role on the SQL server where you installed the Control Center repository. If the primary QDB is also installed on the same SQL server, members of the Administrator group will also have the sysadmin server role on the QDB.

Configuring Control Center Permissions

To configure security permissions in Control Center, you must add users and user groups, define permission sets, and then assign the user groups and permission sets to management groups. You must be a member of the Control Center Administrator group to modify the members of a user group or modify permission sets, but you do not have to be an administrator to assign user groups and permission sets to management groups.

Most organizations start with a few administrative users and add specialized user groups and permission sets over time. Initially, you should allow only expert-level administrators to perform most tasks and you should limit access to the Control Center console to a small number of people until you have firmly established user groups and permissions sets that suit your organization. Once your production environment is stable and your threshold settings, job properties, event-handling, and data-handling policies have been refined to meet your organization's needs, you may want to grant more operators and administrators access to the Control Center console.

In general, once you have created the user groups and permissions sets appropriate to your organization, there is very little account maintenance required for managing user accounts.

To configure Control Center permissions:

1. Add Control Center users. For more information, see "Adding a Control Center User" on page 48.
2. Create or choose an existing user group and add the users to a group. For more information, see "Creating a User Group" on page 50.
3. Create or use an existing permission set. For more information, see "Creating a Permission Set" on page 55.
4. Associate a user group with a permission set and a management group. For more information, see "Granting and Removing Access to Management Groups" on page 59.

Adding a Control Center User

To add a Control Center user, you must be a member of the default **Administrator** user group in Control Center. You can import Windows user or group accounts, create new SQL Server logins, or add existing SQL Server logins.

Importing Windows Users into Control Center

You can import Windows users into the Control Center repository from one of the following domains:

- **Local System Domain.** You can import all the users and groups that are added in your local system domain.
- **Local Domain.** You can import users and groups from network domains that are available within your local area network.
- **One-way Trust Domain.** You can import users and groups from another domain with which your domain has a one-way trust relationship.

The Import process adds the user to the SQL Server and gives the user or group the required permissions on the Control Center repository. You do not need to grant permissions on the SQL Server manually.

To import users to Control Center:

1. On the Global Tasks tab of the Ribbon, click **Manage Security**.
2. In the **Manage Security** dialog box, click the **Users** tab.
3. Click **Import.**
4. In the **Import Users** dialog box, click **Import.**
5. To select the domain from which you want to import users, click **Locations** and select the domain you want.
6. Click **Advanced**.
7. Click **Find Now**.
8. Select the users you want to add to Control Center. You can select multiple users.
9. Click **OK**.
10. Select the QDBs where you want to register the user. This permits the user to manage the registered repositories.
11. Click **OK**.

The Manage Security dialog box displays the user names along with their respective domains, and the user type displays **Windows User.** For example, if you import User1 from domain A and User2 from domain B, the Manage Security dialog box displays the user names as A\User1 and B\User2.

Note

Windows users who have already logged in to Control Center with a particular set of permissions can continue to perform all their assigned activities even if you delete the user account from the domain. Control Center denies access to such users only when they log out and try to log on to the Control Center console again.

Adding or Creating SQL Users in Control Center

You can create new SQL Server logins or add existing SQL Server logins.

To create or add a SQL Server login:

1. On the Global Tasks tab, click **Manage Security**.
2. In the Manage Security dialog box, click the **Users** icon.
3. Click **Create New**.
4. In the Create New User dialog box, specify:

Field	Action
User Name	Specify the login name of the SQL user account you want to add. This can be a new SQL login name or the name of an existing SQL login. **Note**: • You cannot specify login names with certain special characters. These characters include: \ / * ? : < > | “ • You can specify a case-sensitive user name in a case-sensitive SQL Server environment.
Password	Specify the password of the SQL user account. If the account does not exist, it is created in the Control Center repository and is given **public** and **CC_public** permission in the Control Center repository. When you use the Control Center console to create a new SQL user account: • Ensure the login name for the user account is less than 29 characters and the password is less than 32 characters. If the user name or password is too long, it is truncated and you cannot log in to the Control Center console. • If the SQL Server is case-sensitive, do not create the same user name with a different capitalization. • If your database has a strong password policy, make sure the password meets your policy. • If you add an existing SQL user, specify the same password that the SQL user uses to log in to the SQL Server.
Register users with the selected repositories	Select the repositories you want the user to be able to manage. If you do not register a SQL login with a QDB when you first create the account, you cannot register the account with a QDB later using the Control Center console. You must use SQL Server Management Studio to give the SQL login proper permissions for a QDB.

5. Click **OK**.

   The Manage Security dialog box displays the user and the user description. For example, if you create a new user, the user type displays **AppManager.**

Note
If you have enabled FIPs compliant security in the Control Center Console you cannot create SQL Server logins using Security Manager.

Understanding the Administrator Group

Control Center includes a predefined **Administrator** user group. Only members of the Administrator user group can:

- Manage Control Center security, including adding and removing Control Center users and configuring user groups and permission sets
- Configure the QDBs that are managed by Control Center, including adding and removing a repository
- Configure Control Center preferences under **Options** on the Main tab of the ribbon
- View Control Center commands in the Queue Manager
- View AppManager license information under **View Licenses** on the Global Tasks tab of the ribbon

Control Center users who belong to the Administrator user group have full access to Control Center, including all management groups. The Administrator user group does not need to be associated with any management groups or permission sets in order to grant its members access and privileges in the Control Center console.

By default, the command queue service account that you entered during installation and the netiq account belong to the Administrator user group.

When you add a user to the Control Center Administrator user group, Control Center automatically adds the user to the Microsoft SQL Server System Administrators (sysadmin) server role. Therefore, you should restrict the members of the Administrator group to users who you want to belong to the Microsoft SQL Server System Administrators server role. After you remove a user from the Control Center Administrator group, Control Center automatically removes the user from the Microsoft SQL Server System Administrators server role.

Creating a User Group

You can create a Control Center user group that contains local or domain Windows user accounts or SQL Server logins. You can create a user group and add SQL Server logins to the group and you can import Windows users and groups. A user can belong to more than one user group.

To create a user group:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **User Groups** and then click **Create New**.
3. Specify a name and description for the Control Center user group.
4. Click **Add**.
5. Select the users you want to add to the group, and click **OK**.

Importing a User Group

You can import user groups into the Control Center repository from the following domains:

- **Local System Domain.** You can import all the user groups that are added in your local system domain.
- **Local Domain.** You can import user groups from network domains that are available within your local area network.
- **One-way Trust Domain.** You can import user groups from another domain with which your domain has a one-way trust relationship.

Notes

- You need to log in as the Administrator user of the trusted domain to import user groups from the trusted domain.
- You can import user groups from trusted domains within the same forest. However, ensure that the user groups are either Global groups or Universal groups.

Ensure that all the members of the Windows user group you import into Control Center have access to SQL Server.

To import Windows user groups:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **User Groups**.
3. Click **Import**.
4. In the Import User Groups dialog box, click **Import**.
5. In the Select Groups dialog box, click **Locations**.
6. Select the domain from which you want to import user groups.
7. Click **Advanced**.
8. Click **Find Now**.
9. Select the user groups you want to add to Control Center, and then click **OK**.

   You can select multiple user groups.
10. Select the QDBs where you want to register the group, and then click **OK**.

    This permits the group members to manage the registered repositories.

The Manage Security dialog box displays the user group names along with their respective domains. For example, if you import User Group1 from domain A and User Group2 from domain B, the Manage Security dialog box displays the user names as A\User Group1 and B\User Group2.

Modifying a User Group

You can modify an existing user group to add or remove users from the group or change the name or description of the group. You can only change user groups you created in the Control Center console. To modify user groups you imported from Windows, you must make those modifications using administrative tools for Active Directory.

To modify a user group:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **User Groups**.
3. Select the user group you want to modify, and then click **Modify**.
4. In the Modify User Group dialog box:

To change...	Do this...
The name of the user group	Specify a new name.
The description of the user group	Specify a new description.
The users who belong to the user group	Click **Add** to add a user. To remove a user, select the user and click **Remove.**

5. Click **OK**.

Removing a User Group

Removing a user group from the Control Center console prevents group members from logging in to the Control Center console. However, the group still has Operator Console access on each QDB if you have configured this access. You cannot remove a Windows user group from the Active Directory using the Control Center console. If you are removing all members of a user group from the Control Center console, NetIQ Corporation recommends deleting all members of the group before you delete the user group.

To remove a user group:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **User Groups**.
3. Select the user group you want to remove, and then click **Delete.**

Copying a User Group

You can create new user groups by copying an existing user group and modifying it. When you copy a user group, all the members of the original user group are added to the duplicate user group. If you copy a Windows user group, Control Center creates a new user group in Control Center but not in the Active Directory. You cannot copy the Administrator user group.

To copy a user group:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **User Groups**.
3. Select the user group you want to copy, and then click **Make Copy.**

4. In the **Copy User Group** dialog box:

To change...	Do this...
The name of the user group	Specify a new name.
The description of the user group	Specify a new description.
The users who belong to the user group	Click **Add** to add a user. To remove a user, select the user and click **Remove.**

5. Click **OK**.

Default User Groups

The Control Center console includes a set of default user groups you can use, modify, copy, or delete to help implement security for the console. The default user groups include the following:

- Administrator
- Executives and Stakeholders
- NOC Tier 1
- NOC Tier 2
- Trusted Application Admins
- Trusted Application Owners

You cannot copy or delete the Administrator group.

Understanding Permission Sets

A permission set is a collection of operational and Knowledge Script permissions that defines a group of activities that can be performed and Knowledge Scripts that can be used in the Control Center console. To apply a permission set, you associate the permission set with a user group and a management group. Users belonging to that particular user group can perform the activities that you define in the permission set for the associated management group. You can associate the same user group with different permission sets for different management groups. For more information about applying permission sets, see "Granting and Removing Access to Management Groups" on page 59.

You should define permission sets with the specific rights needed by a user to perform a particular job or function using the Control Center console. You can further refine the application of permissions by associating permission sets with specific user groups and management groups.

To perform any tasks related to the management of discovered objects in the Control Center console, a user must be a member of a user group and that user group must be associated with at least one management group and a permission set. This is true of all Control Center console users except members of the Administrator user group, who have access to all management groups, permissions, and Knowledge Scripts by default.

AppManager users do not need access to any management groups to perform remote deployment tasks. You can define user groups and permission sets with specific remote deployment permissions and assign the permission sets and user groups as global permissions. This will grant members of these user groups the remote deployment permissions you defined in the user group and permission set pairings. Members of these user groups will not have any access to management groups unless you also assign the user groups to one or more management groups.

Understanding Granted, Not Granted, and Denied

The same user can belong to more than one user group. Should this be the case, the most restrictive set of permissions is applied by combining the permissions with a logical OR. For example, if the same user is a member of two user groups associated with the same management group but with different permission sets, and is granted rights in one permission set but denied the same rights in the other permission set, then the rights are denied. If a permission is undefined (neither granted nor denied) for the same user in two different user groups, then the permission is denied. If a permission is granted for one user group and either undefined or granted in another group for the same user, then the permission is granted.

Understanding Global Permissions

A global permission set is a permission set associated with a specific user group that applies to all management groups managed by the Control Center console. Since global permissions apply to all management groups, they do not depend on association with a specific management group to take effect.

You can use global permissions to set a common or base set of permissions for a user group for your entire environment. You can then refine these permissions by applying specific permission sets for the same user group to individual management groups.

The Control Center console applies any global permissions and any permissions specific to a management group to determine the security context for any objects in a management group. Control Center applies any global permissions and any permissions assigned to the management group in the following order:

- If an operational or Knowledge Script permission is denied either globally or for a management group, the permission is denied.
- If an operational or Knowledge Script permission is granted either globally or for a management group, the permission is granted.
- If an operational or Knowledge Script permission is neither granted nor denied, the permission is denied.

The Control Center console provides a default set of global permissions:

User Group Name	Permission Set Name
Executives & Stakeholders	Read Only
NOC Tier 1	Event Operation
NOC Tier 2	Monitoring Operation
Trusted Application Admins	Monitoring Administration
Trusted Application Owners	Management Group Administration

For more information, see "Setting Global Permissions" on page 56 and "Default User Groups and Permission Sets" on page 60.

Understanding Permission Inheritance

Permissions assigned to a management group are inherited by any children of that management group, so any user group assigned to the management group will have the same permissions on any children of the management group. You do not need to assign user groups or permissions sets individually on each child management group.

If a user group has a global permission set assigned to it and the permission set includes management group permissions, members of the user group will have those permissions on all management groups and child management groups associated with the user group.

Creating a Permission Set

You can associate user groups with permission sets when you create a management group and define the security of the management group. You can also directly associate user groups with permission sets as global permissions. For more information, see "Setting Global Permissions" on page 56.

To create a permission set:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **Permission Sets**, and then click **Create New**.
3. In the New Permission Set dialog box, specify the following details:

Field	Action
Name	Specify a name for the permission set. **Note**: You can specify permission set names with blank spaces and special characters.
Description	Specify a short description for the permission set.
Operational and Knowledge Script permissions	• Click the check boxes to grant permissions. • Double-click the check boxes to deny permissions. • Do not click the check box if you do not want to grant or deny a permission.

4. Click **OK**.

Modifying a Permission Set

You can choose to modify the permissions in a permission set.

To modify the permission set:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **Permission Sets**.
3. Select the permission set you want to modify, and then click **Modify**.

   In the Permission Set Properties dialog box, you can:

   - Change the name of the permission set.
   - Change the description of the permission set.
   - Change the operational and Knowledge Script permissions by granting or denying permission to a particular activity or Knowledge Script category.
4. Click **OK**.

Removing a Permission Set

You can delete unused permission sets. If the permission set is associated with a management group or with a user group as a set of global permissions, you cannot delete the permission set.

To delete a permission set:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **Permission Sets**.
3. Select the permission set you want to delete, and then click **Delete.**
4. Click **Yes**.

Copying a Permission Set

You can create new permission sets based on a copy of an existing permission set. You need to first copy the permission set and then modify the permissions.

To copy an existing permission set:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click **Permission Sets**.
3. Select the permission set you want to copy, and then click **Make Copy**.
4. Specify the name and an optional description of the new permission set.
5. Modify the operational and Knowledge Script permissions as needed to define the new permission set.
6. Click **OK**.

Setting Global Permissions

Global permissions are permission sets that apply to specific user groups for all management groups in the Control Center console. For more information about global permissions, see "Understanding Global Permissions" on page 54.

To create, modify, or remove a global permission set:

1. On the Global Tasks tab of the ribbon, click **Manage Security**.
2. In the Manage Security dialog box, click the **Global permissions** tab.
3. ***If you want to create or modify a global permission set***:
   a. Select the user group for which you want to assign a global permission set, and then click **Assign**.
   b. In the Assign Permissions dialog box, select a permission set from the **Permission Set** list, and then click **OK**.
4. ***If you want to remove a global permission set***, select the user group you want, and then click **Remove**.
5. Click **Close**.

Permissions for Control Center

You can set four types of permissions in Control Center. Three of these pertain to operational permissions and the fourth pertains to Knowledge Scripts:

- Operational permissions.
  - Deployment permissions. These permissions allow you to perform tasks specific to remote deployment.
  - General permissions. These permissions allow you to add computers to the Control Center repository.
  - Management group and view permissions. These permissions allow you to perform tasks specific to management groups.
- Knowledge Script permissions. These permissions determine which Knowledge Scripts can be used according to Knowledge Script category.

You can set the following operational permissions to control what a user group can do in the Control Center console.

Permissions	Functional Group	Available Rights
Deployment Permissions	Packages	• Check in packages • Delete packages
	Rules	• Copy rules • Create rules • Delete rules • Enable and disable rules • Import rules • Modify rules
	Deployment Tasks	• Approve tasks • Change credentials for tasks • Change schedules for tasks • Configure tasks • Delete tasks • Reject tasks
General Permissions	Computer	• Add a computer to a repository.
Management Group and View Permissions	Chart	• Access views • Delete data streams • Modify views
	Custom Property	• Create custom properties • Delete custom properties • Update custom properties
	Event	• Access views • Acknowledge and close events • Create and modify views • Delete views • Delete events • Modify event comments

Permissions	Functional Group	Available Rights
	Job	• Access views • Add child jobs • Close jobs • Create and modify views • Create jobs • Delete views • Delete jobs • Modify job properties • Start jobs • Stop jobs
	Knowledge Script	• Access views • Check Knowledge Scripts and Knowledge Script Groups into repositories • Check Knowledge Scripts and Knowledge Script Groups out of repositories • Copy Knowledge Scripts and Knowledge Script Groups • Create and modify views • Create Knowledge Script groups • Delete views • Delete Knowledge Scripts and Knowledge Script Groups • Modify Knowledge Script and Knowledge Script Group properties • Propagate Knowledge Script properties to jobs and Knowledge Script Group members
	Management Group and Folder	• Access groups • Change group and folder general properties • Change group members • Change group policies • Change group security permissions • Create and modify groups and folders • Move groups and folders
	Monitoring Policy	• Close monitoring policy jobs • Create monitoring policies • Delete monitoring policies • Start monitoring policy jobs • Stop monitoring policy jobs • Update monitoring policies

Permissions	Functional Group	Available Rights
	Server	• Access views • Create and modify views • Delete views • Delete servers • Put servers into and remove servers from maintenance mode
	Service Map	• Access maps • Create and modify maps • Delete maps

Granting and Removing Access to Management Groups

Control Center users must be given permission to access a management group. You must be a member of the Control Center Administrator group to modify the members of a user group or modify permission sets, but you do not have to be an administrator to assign user groups and permission sets to management groups.

You can configure each management group to give one or more user groups permissions to objects in the management group. The permission set that you associate with each user group determines what the members of the user groups can do with objects in the assigned management group.

AppManager users may be members in more than one user group assigned to a management group. If this is the case, the resulting set of permissions is based on combining all the applicable permission sets with a logical OR to produce the most restrictive permissions. That is, if a permission is denied in any permission set it is denied even if it is granted in another permission set. If a permission is neither granted nor denied in all the permission sets, the permission is denied.

If you assign a user group and a permission set to a management group and that user group also has a global permission set defined, the resultant set of permissions is also determined by combining the permissions with a logical OR to produce the most restrictive permissions. For more information about global permission sets, see "Understanding Global Permissions" on page 54.

You can only assign one permission set at a time to a user group for a management group. You can assign the same user group to a management group more than once with different permission sets. However, if you do this the resultant set of permissions for the members of the user group is the result of a logical OR of all the permissions defined across all associated permission sets to produce the most restrictive set of permissions.

You can organize management groups into a hierarchy, and permissions you assign to the top-level management group in the hierarchy are inherited by the children of that top-level management group. For more information about permission inheritance, see "Understanding Permission Inheritance" on page 55.

To grant or remove access to a management group:

1. Right-click the management group in the Enterprise Layout pane and choose **Management Group Properties > Security**.
2. ***If you want to grant access:***
   a. Click **Add**.
   b. In the Assign Permissions dialog box, select a user group from the **User Group** list.
   c. Select a permission set from the **Permission Set** list.

   Note
   You also have the option to modify an existing permission set by clicking **Modify** or creating a new permission set by clicking **Create New**.

   d. Click **OK**.
3. ***If you want to remove access***, select the user group you want to remove, and then click **Remove**.

If you want to change the permission set associated with a user group, you must first remove the user group and then add it back with the permission set you want.

Default User Groups and Permission Sets

The Control Center Console has a default set of user groups and permission sets. You can use these groups and permission sets, copy or modify them to develop your own groups and permission sets, or delete them if they do not meet your requirements. You cannot delete the Administrator group.

The default user groups are:

- Administrator
- Executives & Stakeholders
- NOC Tier 1
- NOC Tier 2
- Trusted Application Admins
- Trusted Application Owners

The default permission sets are:

- AppManager Administrator
- Deny Management Group Access
- Event Operation
- Management Group Administration
- Monitoring Administration
- Monitoring Operation
- Read Only

The following tables provide details about the specific operational permissions granted for each of the default permission sets. The tables show only those permissions that have been expressly granted. All other available permissions in the default permission sets are neither granted nor denied. All default permission sets grant access to all Knowledge Script categories. For more information about all of the available operational permissions, see "Permissions for Control Center" on page 57.

The Deny Management Group Access default permission set does not grant any operational permissions.

AppManager Administrator Operational Permissions

This table describes the operational permissions defined in the AppManager Administrator default permission set.

Permissions	Functional Group	Available Rights
Deployment Permissions	Packages	• Check in packages • Delete packages
	Rules	• Copy rules • Create rules • Delete rules • Enable and disable rules • Import rules • Modify rules
	Deployment Tasks	• Approve tasks • Change credentials for tasks • Change schedules for tasks • Configure tasks • Delete tasks • Reject tasks
General Permissions	Computer	• Add a computer to a repository
Management Group and View Permissions	Chart	• Access views • Delete data streams • Modify views
	Custom Property	• Create custom properties • Delete custom properties • Update custom properties
	Event	• Access views • Acknowledge and close events • Create and modify views • Delete views • Delete events • Modify event comments

Permissions	Functional Group	Available Rights
	Job	• Access views • Add child jobs • Close jobs • Create and modify views • Create jobs • Delete views • Delete jobs • Modify job properties • Start jobs • Stop jobs
	Knowledge Script	• Access views • Check Knowledge Scripts and Knowledge Script Groups into repositories • Check Knowledge Scripts and Knowledge Script Groups out of repositories • Copy Knowledge Scripts and Knowledge Script Groups • Create and modify views • Create Knowledge Script Groups • Delete views • Delete Knowledge Scripts and Knowledge Script Groups • Modify Knowledge Script and Knowledge Script Group properties • Propagate Knowledge Script properties to jobs and Knowledge Script Group members
	Management Group and Folder	• Access groups. • Change group and folder general properties • Change group members • Change group policies • Change group security permissions • Create and modify groups and folders • Move groups and folders
	Monitoring Policy	• Close monitoring policy jobs • Create monitoring policies • Delete monitoring policies • Start monitoring policy jobs • Stop monitoring policy jobs • Update monitoring policies

Permissions	Functional Group	Available Rights
	Server	• Access views • Create and modify views • Delete views • Delete servers • Put servers into and remove servers from maintenance mode
	Service Map	• Access maps • Create and modify maps • Delete maps

Event Operation Operational Permissions

This table describes the operational permissions defined in the Event Operation default permission set.

Permissions	Functional Group	Available Rights
Management Group and View Permissions	Event	• Access views • Acknowledge and close events • Delete events • Modify event comments
	Management Group and Folder	• Access groups
	Server	• Access views • Put servers into and remove servers from maintenance mode
	Service Map	• Access maps

Management Group Administration Operational Permissions

This table describes the operational permissions defined in the Management Group Administration default permission set.

Permissions	Functional Group	Available Rights
Deployment Permissions	Rules	• Enable and disable rules • Modify rules
	Deployment Tasks	• Approve tasks • Change credentials for tasks • Change schedules for tasks • Delete tasks • Reject tasks
General Permissions	Computer	• Add a computer to a repository
Management Group and View Permissions	Custom Property	• Create custom properties • Delete custom properties • Update custom properties

Permissions	Functional Group	Available Rights
	Event	• Access views • Acknowledge and close events • Create and modify views • Delete views • Delete events • Modify event comments
	Job	• Access views • Add child jobs • Close jobs • Create and modify views • Create jobs • Delete views • Delete jobs • Modify job properties • Start jobs • Stop jobs
	Knowledge Script	• Access views • Check Knowledge Scripts and Knowledge Script Groups into repositories • Check Knowledge Scripts and Knowledge Script Groups out of repositories • Copy Knowledge Scripts and Knowledge Script Groups • Create and modify views • Delete views • Delete Knowledge Scripts and Knowledge Script Groups • Propagate Knowledge Script properties to jobs and Knowledge Script Group members
	Management Group and Folder	• Access management groups • Change management group and folder general properties • Change management group members • Change management group policies • Change management group security permissions • Create and modify management groups and folders • Move management groups and folders
	Monitoring Policy	• Close monitoring policy jobs • Create monitoring policies • Delete monitoring policies • Start monitoring policy jobs • Stop monitoring policy jobs

Permissions	Functional Group	Available Rights
	Server	• Access views • Create and modify views • Delete views • Delete servers
	Service Map	• Access maps • Create and modify maps • Delete maps

Monitoring Administration Operational Permissions

This table describes the operational permissions defined in the Monitoring Administration default permission set.

Permissions	Functional Group	Available Rights
General Permissions	Computer	• Add a computer to a repository
Management Group and View Permissions	Custom Property	• Create custom properties • Delete custom properties • Update custom properties
	Custom Property	• Create custom properties • Delete custom properties • Update custom properties
	Event	• Access views • Acknowledge and close events • Delete events • Modify event comments
	Job	• Access views • Add child jobs • Close jobs • Create jobs • Delete jobs • Modify job properties • Start jobs • Stop jobs
	Knowledge Script	• Access views • Copy Knowledge Scripts and Knowledge Script Groups • Create Knowledge Script Groups • Delete Knowledge Scripts and Knowledge Script Groups • Modify Knowledge Script and Knowledge Script Group properties • Propagate Knowledge Script properties to jobs and Knowledge Script Group members
	Management Group and Folder	• Access groups • Change group policies

Permissions	Functional Group	Available Rights
	Monitoring Policy	• Close monitoring policy jobs • Start monitoring policy jobs • Stop monitoring policy jobs
	Server	• Access views • Delete servers • Put servers into and remove servers from maintenance mode
	Service Map	• Access maps

Monitoring Operation Operational Permissions

This table describes the operational permissions defined in the Monitoring Operation default permission set.

Permissions	Functional Group	Available Rights
Management Group and View Permissions	Custom Property	• Update custom properties
	Event	• Access views • Acknowledge and close events • Delete events • Modify event comments
	Job	• Access views • Start jobs • Stop jobs
	Management Group and Folder	• Access groups
	Server	• Access views • Put servers into and remove servers from maintenance mode
	Service Map	• Access maps

Read Only Operational Permissions

This table describes the operational permissions defined in the Read Only default permission set.

Permissions	Functional Group	Available Rights
Management Group and View Permissions	Event	Access views
	Job	Access views
	Management Group and Folder	Access groups
	Server	Access views
	Service Map	Access maps

Understanding the Interaction Between Control Center Console and Operator Console Security

Permissions in the Control Center console depend on user group, permission set, and management group assignments in the Control Center console as well as role assignments in Security Manager. In some instances, the permissions a Control Center console user has for a specific AppManager repository are limited by the permissions granted to that same user by Security Manager, regardless of the permissions they are granted in the Control Center console. The permissions granted a Control Center console user apply on a repository by repository basis. If a Control Center console is managing more than one AppManager repository, user permissions may need to be set for each repository in Security Manager.

By default, when you add a user in the Control Center console and register the user with one or more AppManager repositories, that user is granted Read Only permissions in Security Manager for each repository. This is true for both Windows users and SQL login accounts. In most cases Read Only permissions in Security Manager are adequate for any task a Control Center user might perform other than those performed by Control Center administrators.

In some instances, you may need to configure permissions in Control Center that require more than Read Only and less than full Administrator permissions in Security Manager. The following list indicates tasks in Control Center that require more than Read Only permissions in Security Manager:

Check in Knowledge Scripts

To check Knowledge Scripts into an AppManager repository, a Control Center console user requires the Check In a Knowledge Script functional right assigned to them in Security Manager either through a custom role or by assigning the user to the Standard User role. This permission only needs to be set on the primary AppManager repository. Knowledge Script synchronization handles the replication of the new Knowledge Script to any secondary repositories.

Create Job

To create jobs on managed resources, a Control Center user requires the Check In a Knowledge Script functional right on any AppManager repository where the user wants to create jobs.

Create a Knowledge Script Group

To create a Knowledge Script Group in Control Center, the user requires the Administrator role in Security Manager on the primary AppManager repository.

Copy a Knowledge Script or Knowledge Script Group

To copy a Knowledge Script or a Knowledge Script Group in Control Center, the user requires the Administrator role in Security Manager on the primary AppManager repository.

Modify a Knowledge Script Group

To modify a Knowledge Script Group, such as removing Knowledge Scripts from a group, the user requires the Administrator role in Security Manager on the primary AppManager repository.

The restriction of permissions in the Control Center console based on an assigned role in the Operator Console does not apply to any user added to the default AppManager Administrator group in the Control Center console. Members of this group are granted the sysadmin role in SQL Server on any AppManger repository managed by the Control Center console. The sysadmin role overrides any limitations set by Operator Console roles on an AppManager repository. For more information about the Administrator group in the Control Center console, see "Understanding the Administrator Group" on page 50.

Chapter 4
Managing Jobs

This chapter provides information on managing AppManager jobs.

As the number of jobs increases, managing your environment becomes more challenging. This chapter discusses ways to simplify job management. If you have not done so already, you should familiarize yourself with the basic functionality of AppManager by reading the *Control Center Console User Guide for AppManager*.

Deploying to a Pilot Group

Beginning the process of monitoring your environment with a small, pilot group of computers helps you determine the most critical components to monitor and the most likely source of problems before you begin monitoring and managing on a large scale. This helps ensure a smooth implementation by reducing the likelihood of creating too many events or leaving critical gaps in the systems and applications you are monitoring.

Depending on your organization's size, the importance of your monitoring needs, your deployment team's expertise, and the resources available to you, the pilot deployment might involve a small but representative number of computers or all of the computers you intend to monitor. NetIQ Corporation recommends installing on enough computers to get a realistic view of the full-scale deployment. The pilot deployment should last from two to four weeks and reveal the following information:

- Problems that need immediate attention, such as computers that are low on disk space
- Environmental issues you need to address, such as insufficient privileges or instability
- How closely the computers you want to monitor conform to your expectations

During the pilot deployment, focus on the following goals:

- Running the recommended core set of Knowledge Scripts on agent computers

  For more information about working with Knowledge Scripts and jobs, see the *Control Center User Guide for AppManager*. For more information about the recommended core set of Knowledge Scripts, see "Implementing Core Monitoring Support" on page 70.

- Identifying and correcting problems with running the core set of jobs

  For example, you might find problems with the required accounts and permissions.

- Gaining experience viewing and responding to events

  For more information about how AppManager raises events and using the Control Center console to view and respond to them, see the *Control Center User Guide for AppManager.*

- Identifying normal operating values and adjusting thresholds for your environment

  For more information about identifying normal operating values, see "Setting and Adjusting Event Thresholds" on page 72.

Implementing Core Monitoring Support

In planning an AppManager deployment, you should first identify a specific set of Knowledge Scripts you want to run. Although the list is likely to change over time, your initial **core set** of Knowledge Scripts should monitor basic server health and availability and your most important application resources. At a minimum, for example, most organizations monitor CPU and memory usage, disk space, disk I/O activity, network connections or activity, and the availability of specific computers or specific processes.

In addition, many organizations monitor computer hardware components and application-specific resources, such as mailbox size for messaging servers and database connections for database servers.

Tip

The core set of Knowledge Scripts should consist of the Knowledge Scripts you want to run at regular intervals for monitoring performance and availability. In general, you should identify a relatively simple set of scripts to act as the core set. You can then extend the core set with additional Knowledge Scripts to perform more detailed analysis, assist you in troubleshooting, or collect data for reports.

In a typical environment, you run approximately 20 jobs on each agent computer at regular intervals to ensure basic operational health and availability. You run additional jobs less frequently to diagnose problems or take corrective action. Although running around 20 jobs is typical, the core set of Knowledge Scripts you initially run might include fewer jobs.

NetIQ Corporation recommends initially running a core set of Knowledge Scripts from the General and NT Knowledge Script categories. The following table describes the recommended core set of Knowledge Scripts. For more information about using these Knowledge Scripts and setting parameters, see the *AppManager Knowledge Script Reference Guide*.

Knowledge Script	Description
General_EventLog	Monitors and filters information in the Windows Event Log and allows you to track log entries that match filtering criteria Initially, NetIQ Corporation recommends monitoring all logs for error events. You can further filter the log entries to include or exclude other criteria such as specific IDs, descriptions, user names, or computer names.
General_MachineDown	Detects whether the computer on which you run the script can communicate with one or more specified Windows computers and raises an event if communication attempts fail
NT_MemUtil	Monitors physical and virtual memory and the paging files and raises an event if a monitored metric exceeds the threshold
NT_LogicalDiskSpace	Monitors logical disk space usage and raises an event if either the percentage of used space or the free space exceeds the threshold
NT_CpuLoaded	Monitors total CPU usage and queue length to determine whether the CPU is overloaded and raises an event when both the total CPU usage and CPU queue length exceed the thresholds

Knowledge Script	Description
NT_LogicalDiskBusy	Monitors logical disk activity and raises an event if a monitored value exceeds the threshold
NT_PhysicalDiskBusy	Monitors physical disk activity and raises an event if a monitored value exceeds the threshold
NT_ServiceDown	Monitors whether specified Microsoft Windows services are stopped or started, and, optionally, starts any stopped service
NT_TrustRelationship	Tests the domain trust relationship from the computer on which you run the script to a specified domain and raises an event if a problem exists with the domain trust

Collecting Data

To identify normal baseline operating values before you set thresholds for events, set all Knowledge Scripts only to collect data (that is, not to raise events) and run reports for at least one week. From the reports, you can review the high, low, and average values for core statistics. You can configure several basic report Knowledge Scripts to create reports.

To create reports about your environment:

1. Install at least one report-enabled agent.
2. Run the Discovery_ReportAgent Knowledge Script on the report-enabled agent computer.
3. In the Report view, click through tabs in the Knowledge Script pane to select the reports to run.

At the end of the collection period, evaluate the information to determine a baseline for a normal operating environment. After you complete your evaluation, remove the data you collected from the QDB. For information about removing data from the QDB, see "Removing Archived Data and Events" on page 124.

When you are ready to raise events, set only those Knowledge Scripts that address critical issues in your environment to raise events, and set the remaining Knowledge Scripts to collect data. You can employ this approach enterprise-wide or only on the computers you identify as needing immediate attention. To help you tune your system later, track the frequency of events and the number of data points collected.

Based on the data you collect, you can adjust thresholds to more accurately reflect your environment's specific characteristics. If you see too many events, the thresholds might be too low for your environment, the intervals might be too short, or you might need to address critical resource issues.

Basic AppManager reporting provides detailed information about the computers in a single management site. When you expand your deployment to multiple management sites with multiple QDBs, you might want the more sophisticated reporting available with NetIQ Analysis Center.

Setting and Adjusting Event Thresholds

Once you've identified a core set of Knowledge Scripts and baseline operating values for monitoring basic computer resources, such as CPU, memory, and disk, and critical application resources, create a Knowledge Script Group from those Knowledge Scripts and run them on a pilot group of computers.

The servers in your pilot group should have similar configurations and be similarly loaded. For example, you may want to set different event thresholds for servers that perform transactional operations than for servers that perform batch operations, so you would organize transactional and batch servers into separate management groups or views.

With a group of similarly configured and loaded servers, you should run the core set of Knowledge Scripts to raise events only for critical issues in your environment. You can use the default threshold values or your own estimation for initial threshold settings based on the results of your initial data collection.

Tip

Using a monitoring policy may simplify event threshold configuration. With a monitoring policy, the jobs are started automatically, changes to Knowledge Script group member properties are automatically propagated to policy-based jobs, and when you remove the policy, the jobs are automatically stopped and deleted.

The process of establishing effective event thresholds includes several basic steps. By following these steps with a pilot group of servers, you establish threshold values you can use through the rest of your enterprise:

- Identify a group of servers that have a similar configuration.
- Identify the event conditions most relevant to you for those servers.
- Identify the Knowledge Scripts you want to run to monitor the event conditions you identified.
- Run monitoring jobs and make adjustments to the event conditions and event thresholds as needed. The goal is to set event thresholds you believe to be accurate for the servers and applications most critical to your business.

The purpose of running a core set of jobs on a pilot group of computers is to reveal:

- Serious problems that need immediate attention—for example, computers that are dangerously low on disk space or that have high CPU usage
- Any environmental issues you need to address—for example, problems with insufficient account privileges, network instability, or the availability of SNMP or other services that need to be installed
- Threshold levels and job properties that are appropriate to your specific environment and which you can standardize, either across your entire organization or across specific departmental or functional group

If you are seeing too many events, the thresholds may be set too low for your environment, or the interval for running the job may be too short. Events should not be raised unless something has happened that merits a response. Responses include acknowledging the event, running another Knowledge Script to remotely diagnose the problem, or diagnosing the system in person.

Deploying a core set of Knowledge Scripts also prevents your staff from being overwhelmed by a sudden barrage of events. By focusing on a limited number of key Knowledge Scripts and the most critical problems you need to address early in the deployment, you can develop an understanding of the events generated, implement a methodology for responding to those events, and effectively troubleshoot any issues that arise.

In your initial deployment, therefore, the core Knowledge Scripts should not perform responsive actions when events are raised. Avoiding actions in the earliest stages of deployment prevents an unnecessary surge of e-mail or pager messages being sent for events caused by thresholds that have been set too high or too low. Once you have determined appropriate thresholds for your environment, you can test responsive actions and choose an appropriate notification method, such as MAPI mail, SMTP mail, or a paging system.

Establishing a Manageable Level of Event Activity

If you are receiving too many events, you may need do some or all of the following:

- Adjust thresholds. Whether they need to be higher or lower depends on your environment, on your reasons for monitoring a particular computer, and on how particular computers are being used. For example, when monitoring the computers in a lab to determine when you are nearing capacity, you may set thresholds lower than when monitoring users desktop computers or computers that store archived information that rarely changes.
- Change the job schedule (increase or decrease the monitoring interval).
- Change the number of consecutive times that a condition must be detected before an event is raised. For more information, see "Adjusting Consecutive Intervals" on page 92.
- Modify the computer configuration to bring non-conforming computers in line with the benchmark settings or manage the non-conforming servers using another management group.

Developing a Data Collection Strategy

Once you are monitoring for events on your core systems and applications, you are ready to collect data for charts and reports. When considering your reporting needs, determine the following information:

- Standard AppManager reports to generate and the Knowledge Scripts required to generate those reports
- Who should receive the reports and how frequently
- Whether to generate reports automatically on a scheduled basis or manually on demand
- Who will generate reports

  For example, you might want to restrict access to the Report view or assign Exchange reports to an Exchange administrator and SQL Server reports to your DBA group.
- Whether to format reports in table format, in charts, or both
- Whether to deliver reports through e-mail, a Web site, or the Report Viewer

The following table describes report Knowledge Scripts that NetIQ Corporation recommends running to generate standard reports. For more information about using these Knowledge Scripts and setting parameters, see the *AppManager Knowledge Script Reference Guide*.

Knowledge Script	Description
ReportAM_EventSummary	Summarizes events per computer
ReportAM_SystemUpTime	Details the uptime and downtime of monitored computers
ReportAM_CompDeploy	Details the number of instances of each AppManager component installed on computers in an AppManager site
ReportAM_WatchList	Details the top or bottom n computers (by number or percent) generating the selected data streams
NT_Report_CPULoadSummary	Summarizes CPU usage and queue length for selected computers
NT_Report_LogicalDiskUsageSummary	Summarizes the percentage of disk space used and the amount of free space (in MB) for selected computers

When collecting data, you should familiarize yourself with how AppManager collects data for charts and reports. You should set repository preferences and job properties so that you only collect and maintain the data you need. Storing additional data can quickly consume repository database resources and negatively impact performance. For more information, see "Managing Data" on page 99 and "Managing a QDB" on page 107.

If you need to report on more than three months' worth of data, consider using AppManager Analysis Center. The aggregate reporting capabilities available with Analysis Center are powerful and can avoid the performance problems potentially associated with storing large amounts of AppManager data for reports.

Expanding the Scope of Your Deployment

When you feel comfortable with the core set of Knowledge Scripts and your environment's stability, consider expanding your deployment. During the expansion stage, focus on the following goals:

- Deploying AppManager to additional computers

  Large or widely distributed organizations typically phase in a full AppManager deployment over a period of several weeks or even months. For example, if your organization is going to monitor a group of computers in the United States, Germany, and Spain, you might decide to deploy AppManager first in Germany, stabilize the environment there, and then expand the deployment to include computers in Spain and the United States. Or you might decide to expand the deployment to include the computers in Spain, allow time to uncover problems and stabilize that environment, and deploy to the computers in the United States later.

- Running additional Knowledge Scripts beyond the core set

  For more information about additional recommended Knowledge Scripts, see "Running Additional Knowledge Scripts" on page 75.

- Adding responsive and corrective actions to Knowledge Scripts

  AppManager Knowledge Scripts can automatically take corrective actions, notify selected people in response to certain events, and acknowledge events. To take advantage of Knowledge Script automation capabilities, you might need to install additional components, such as an agent that can send email responses to events. For more information about enabling agents to send email responses to events, see the *Installation Guide for AppManager*. For more information about responsive and corrective actions, see the *Control Center User Guide for AppManager*.

Running Additional Knowledge Scripts

During the expansion stage, add Knowledge Scripts beyond the core set. The following table describes Knowledge Scripts that NetIQ Corporation recommends adding. For more information about using these Knowledge Scripts and setting parameters, see the *AppManager Knowledge Script Reference Guide*

Knowledge Script	Description
General_AsciiLog	Monitors one or more ASCII text files for specific strings and messages
General_Counter	Monitors any System Monitor counter
NT_NetworkBusy	Monitors the traffic on network interface cards (NICs) and raises an event if the network interface's bandwidth utilization exceeds the threshold
NT_PagingHigh	Monitors reads and writes per second to the pagefile and raises an event if the number of reads and writes per second exceeds the threshold
NT_PrinterHealth	Monitors printer health and raises an event if the printer is paused, the queue length exceeds the threshold, or there is some other error such as a jammed printer
NT_PrinterQueue	Monitors printer queue length and raises an event if the number of queued jobs exceeds the threshold
NT_RunAwayProcesses	Detects runaway processes on the specified computer based on sustained high CPU usage and raises an event if a process exceeds the CPU usage threshold
NT_SystemUpTime	Monitors the number of hours a computer has been operational since it was last rebooted and raises an event if the computer was rebooted within the monitoring interval

Once you select a set of Knowledge Scripts for monitoring basic server health and key application resources, you can plan for and implement policy-based monitoring. For information about implementing monitoring policies, see the *Control Center User Guide for AppManager.*

Strategies for Managing Systems and Applications

Once you have established your core monitoring needs and identified appropriate monitoring thresholds, you are ready to manage these systems and applications on a daily basis. Depending on how and where you deploy your core Knowledge Scripts, you may be able to better manage the resulting jobs now and in the future.

AppManager simplifies the management of your Windows, UNIX, and Linux systems by automatically managing similarly configured and loaded systems and applications. When the servers in a group are similarly configured and loaded, they are *conformant*. By organizing the systems and applications in your environment into groups of conformant servers, you can easily monitor for event conditions and collect data on those servers using a standard set of Knowledge Scripts.

Ideally, implement core monitoring using monitoring policies, with ad hoc jobs used only to diagnose problems detected by these policies. A simple strategy for managing your environment with AppManager is to:

- Identify the critical systems and applications in your environment and configure jobs that raise an event if something goes wrong with those systems.
- Organize your critical systems and applications so that you can effectively manage them on a daily basis.
- Configure jobs to collect data for historical reporting and trend analysis.
- Manage additional systems and applications by organizing conformant systems and applications under existing monitoring policies.
- Remotely diagnose problems on your policy- systems and applications by running additional jobs.

Managing Systems and Applications with the Operator Console

The following sections outline how you can use the Operator Console to successfully monitor the systems and applications in your environment.

Managing Systems in the Master View

The **Master** view in the Operator Console displays all discovered resources. You can use the Master view to discover and monitor all of the resources on a server, including the operating system, hardware, and application resources. Depending on your environment, you may not want to allow your operations staff to have access to the Master view.

When deploying monitoring policies in the Master view, only apply monitoring policies to **a group of servers**. If you attach a monitoring policy to the Master view itself, the only way to stop policy-based monitoring on a server is to remove the policy from the Master view or delete the server from the TreeView. In either case, the policy-based jobs are stopped and deleted, which inhibits reporting.

You can reduce the load placed on the repository each time you change a monitoring policy by implementing monitoring policies on nested groups of servers. Using a "layered" approach to organizing servers and applying monitoring policies reduces the number of jobs that are affected by a policy change.

Managing Systems in a Snapshot View

A snapshot view allows you to discover and monitor servers that are organized into a logical "window" into the repository. The resources that can be displayed in a snapshot view must be discovered and visible from a standard view or the Master view. As new resources are discovered, you must manually update a snapshot view to add the newly discovered resources.

The *Operator Console User Guide for AppManager* contains instructions for creating a snapshot view.

Organizing systems into snapshot views creates the following advantages:

- You can organize systems into nested groups as you would from the Master view.
- AppManager's role-based security allows you to restrict AppManager access to the snapshot view. Depending on your environment, you may not want to give access to the Master view.

When using snapshot views, keep the following in mind:

- To use a snapshot view to monitor all resources on a server, the snapshot view must be based on the Master view.
- When you create a snapshot view, the organization of the servers in the Master view from which the snapshot was derived is copied into the snapshot view, allowing you to mirror the organizational structure that exists in the Master view.
- You cannot change the parent job properties from a snapshot view. To change the job properties for all servers managed by a parent job, you must separately update each child job.

Managing Systems in a Standard View

Standard views allow you to view and manage only the resources that correspond to a particular resource type. For example, the SQL view only displays SQL resources and SQL-related Knowledge Scripts. To run Knowledge Scripts on NT resources, such as ASYNC and NTAdmin Knowledge Scripts, you must use another view. You cannot discover resources from a standard view.

You may find it advantageous to implement a monitoring policy on a standard view when you want to automatically monitor a particular system or application as it is discovered. For example, a monitoring policy on the SQL view ensures that as SQL Servers are discovered, they are managed.

Managing Systems in a Dynamic View

A dynamic view allows you to discover and monitor servers that are organized into a logical view. Unlike snapshot views and the Master view, a dynamic view uses rules to automatically display and organize systems and applications.

Note
If your business needs require a rule-based approach to managing your systems and applications, NetIQ recommends that instead of a dynamic view, you implement a rule-based management group in the Control Center console. In the Control Center console, you can easily configure rules that have more power and flexibility to select the resources you want. See the Control Center console online help for more information.

A dynamic view provides the flexibility to select conforming systems and applications as well as logically group systems using custom properties. Dynamic views work well when you have less information about the system that is being managed and you need to rely on the view rules to select the correct system, or you do not want to give access to the Master view.

Dynamic views are particularly useful when you want to:

- Organize a subset of servers into a view that can be managed by your operations staff. For example, if you configure a dynamic view that selects a custom property value, you can easily control the servers that can be managed by your operations staff.
- Automatically monitor conforming systems and applications. For example, you can configure a dynamic view to automatically select similarly configured and loaded systems, and automatically monitor those systems by policy.
- Implement view-based reporting. For example, you can use a dynamic view to select similarly configured systems and run a report on the servers in that dynamic view.

If you are planning to implement a dynamic view, keep the following in mind:

- You must use the Control Center console to configure custom property information on a managed computer. From the Operator Console, you can only view custom property information.
- You cannot delete a server from a dynamic view. When configuring a dynamic view, it is a good idea to select a custom property value so that, if necessary, you can change the custom property value to remove the server from the dynamic view.
- You cannot create nested groups in a dynamic view. This means that if you want to monitor by policy, you will need to organize servers into separate groups rather than use the "layered" approach that is available in snapshot views and the **Master** view.

Managing Systems and Applications with the Control Center Console

The Control Center console uses management groups to organize managed computers and discovered resources. The console has a default Master management group that includes all managed computers and discovered resources in all QDBs you are managing with the Control Center console.

You can create additional management groups and determine membership in the management group in several different ways:

- Ad Hoc membership. Membership is determined by including specific managed computers. This is similar to a server group in the Operator Console.
- Repository View. Membership is determined by discovered resources on the managed computer, such as IIS or SQL. This is similar to a standard view in the Operator Console. You can also define membership based on the Master repository view, which includes all managed computers and resources.
- Server Group. Membership is determined by membership in an Operator Console server group. You cannot create server groups in the Control Center console.
- Rule. Membership is determined by one or more rules. Rules evaluate a managed computer and its discovered resources and either includes or excludes the managed computer based on the criteria defined in the rule. This is similar to a dynamic view in the Operator Console.

You can combine all of these methods to determine membership in a given management group. For more information about management groups, see the *Control Center User Guide for AppManager*.

The following sections outline how you can use the Control Center console to successfully monitor the systems and applications in your environment.

Managing Systems in the Master Management Group

The Master management group is a default group that displays all managed computers and discovered resources in all QDBs managed by the Control Center console. You can use the Master view to discover and monitor all of the resources on a server, including the operating system, hardware, and application resources. Depending on your environment, you may not want to allow your operations staff to have access to the Master management group.

Since the Master management group includes all the managed computers in your environment, it is not recommended that you apply any monitoring policies to this group. Doing so could adversely affect the performance of AppManager and your ability to manage the resources in your environment. You should also exercise caution in running any ad hoc jobs in the Master management group since this will create jobs on every agent computer.

Managing Systems in a Management Group Based on an Ad Hoc List

A management group can include computers that you identify specifically for inclusion in the management group. This allows you to create a static set of computers that you want to manage as a group. You can include any managed computer. Unlike rule-based management groups, any computer you add to the ad hoc list will remain as a member until you remove it from the list.

Managing Systems in a Management Group Based on a Repository View

Management groups based on a repository view allow you to view and manage only the resources that correspond to a particular resource type. For example, the SQL view only displays SQL resources and SQL-related Knowledge Scripts. To run Knowledge Scripts on NT resources, such as ASYNC and NTAdmin Knowledge Scripts, you must use another view. You can include more than one repository view in a management group.

You may find it advantageous to implement a monitoring policy on a repository view when you want to automatically monitor a particular system or application as it is discovered. For example, a monitoring policy on the SQL view ensures that as SQL Servers are discovered, they are managed.

If you create a management group based on the Master repository view, the management group includes all managed computers and discovered resources just as the default Master management group does.

Managing Systems in a Management Group Based on a Server Group

Management groups based on an Operator Console server group include only those computers that are members of the server group. This allows you to take advantage of any server groups you have already established in the Operator Console for use in the Control Center console.

Managing Systems in a Rule-based Management Group

A rule-based management group allows you to discover and monitor servers that match a specific set of criteria. Unlike other management groups, rule-based management groups uses rules to dynamically update membership in the management group.

A rule-based management group provides the flexibility to select conforming systems and applications as well as logically group systems using custom properties. Rule-based groups work well when you have less information about the system that is being managed and you need to rely on the rules to select the correct system or you do not want to give access to the Master view.

Rule-based groups are particularly useful when you want to:

- Organize a subset of servers into a group that can be managed by your operations staff. For example, if you configure a rule-based management group that selects a custom property value, you can easily control the servers that can be managed by your operations staff.
- Automatically monitor conforming systems and applications. For example, you can configure a rule-based management group to automatically select similarly configured and loaded systems, and automatically monitor those systems by policy.
- Implement group-based reporting. For example, you can use a management group to select similarly configured systems and run a report on the servers in that group.

Managing Existing Jobs

Once you have implemented your core Knowledge Scripts and have established a data-collection strategy, you may need to manage existing jobs by adjusting job properties, by adding new servers, or expanding your core set of Knowledge Scripts.

When adding new systems to your environment, plan to run your core Knowledge Scripts to validate event thresholds before including them in a monitoring policy.

For suggestions on how to move servers into a dynamic view, see "Strategies for Managing Systems and Applications" on page 76.

Changing Job Properties

After you implement your core Knowledge Scripts for monitoring and data collection, the job properties may need adjustment. Changes are propagated automatically to policy-based jobs. For one-time jobs or those not associated with a monitoring policy, you must manually propagate changes to job properties.

When manually propagating job properties, make sure the Knowledge Script is configured with the same action as the running job, if applicable.

If you want to monitor additional systems or applications with a one-time job, manually add the additional server(s) to the existing parent job by right-clicking the parent job, clicking **Add Child Jobs**, and selecting the objects you want to monitor. (A monitoring policy does this for you automatically.)

In the Operator Console, when working with different views that display policy objects, it can be difficult to identify the view where a monitoring policy was created or the Knowledge Script Groups that compose the monitoring policy. In the Extended Support section of the NetIQ Technical Support Web site, the AppManager Knowledge Depot features an Operator Console plug-in and a report that provide information about all existing monitoring policies, such as the view where the policy was created and the Knowledge Script Group members:

- The Operator Console plug-in, `MonitorPolicies.vbs`, adds two Extensions menu commands to display monitoring policy information in a dialog box or write the information to a text file.
- The ReportAM_PolicyInfo Report Knowledge Script creates a report about monitoring policies with hypertext links to the parameter values of each Knowledge Script Group member.

Use your **myNetIQ** Account Login to access the Knowledge Depot at `http://www.netiq.com/support/am/extended/knowledgedepot/default.asp`.

In the Control Center console, it is easier to identify the monitoring policies that apply to a management group and the Knowledge Script Groups that the monitoring policies are based on. In the Enterprise Layout pane, right-click a management group and click **Management Group Properties > Policies**. The Policies tab of the Management Group Properties dialog box lists any Knowledge Script Groups assigned to the management group as a monitoring policy.

Checking Job Status with the JobInfo Report

The JobInfo report is useful for finding out when jobs are stopped, pending, or errored out. We recommend scheduling this report job to run in the morning and to email the report to your team for review.

When responding to:

- **Stopped** jobs, you should restart them if necessary.
- **Pending** jobs, you should run **NetIQCtrl** on the agent computer to verify that the job status is Pending. If the job is Pending on the agent computer, delete the job, or in the case of a policy-based job, stop and delete the job by removing the server from the monitoring policy.
- **Errored** jobs, you should look at the error message and take an appropriate action.

  You can configure the number of times to restart errored jobs within the Properties dialog box of the monitoring policy or as a repository preference.

Suspending AppManager Monitoring

In many environments, you may need to perform maintenance on a computer. For example, an organization may have an Apache Web server that must be shut down. In this case, you can suspend some or all jobs running on a managed computer before you shut down the server.

There are two ways to suspend AppManager monitoring:

- **Machine maintenance**: From the Operator Console and Control Center console, you can suspend all jobs running on a managed Windows, UNIX, or Linux client computer. The maintenance mode status is reflected in the TreeView pane of the Operator Console and in the View pane of the Control Center console immediately, but it can take 10 minutes for the AppManager agent to block jobs and send its status to the repository.

  When reporting on service availability, machine maintenance periods are reported as down time.

- **Scheduled maintenance**: The AMAdmin_SchedMaint Knowledge Script suspends monitoring for a particular Knowledge Script category or all monitoring on a Windows computer for a specified period. On a UNIX computer, use the AMAdminUNIX_SchedMaint Knowledge Script. Scheduled maintenance is initiated on the AppManager agent at the scheduled time, but it can take up to 10 minutes for the TreeView pane of the Operator Console or the View pane of the Control Center console to display the scheduled maintenance status.

  When reporting on service availability, scheduled maintenance periods are not reported as down time.

You **cannot** use ad hoc maintenance to turn off scheduled maintenance. Therefore, it is good practice to avoid using both at the same time.

Suspending Remote Monitoring Knowledge Scripts

Knowledge Scripts that remotely monitor a server, such as the NT_RemoteServiceDown Knowledge Script, continue to monitor a remote server that is in maintenance mode. However, if an event condition is detected while a remote server is in maintenance mode, it is not displayed in the Operator Console or Control Center console. Also, if the remote Knowledge Script is configured to run a responsive action on the **management server**, the action is suppressed.

Do **not** configure a Knowledge Script to run a responsive action on the remote computer. If an event condition is detected, no event will appear in the Operator Console or Control Center console, but the action will run.

Resource Dependencies and Job Schedules

In certain situations, it's a good idea to control job scheduling by setting a resource dependency. For example, if regularly scheduled maintenance periods aren't reliable or are hard to anticipate, you may want to specify that jobs only run when required file-system related resources or specific services are available. Setting a resource dependency is especially useful when running Knowledge Scripts on MSCS (clustered) resource objects to avoid duplicated events and data.

You can use the AMAdmin_SetResDependency Knowledge Script to specify resources and services that must be active and available for the jobs to run. If any resource or service is not available, the jobs are suspended until the specified resource or service becomes available.

For example, if you're monitoring Exchange Server, you may want to check that the `MSExchangeDS`, `MSExchangeIS`, and `MSExchangeSA` services are running before running Exchange jobs. Even if you have established a maintenance period and the maintenance period has expired, if these services are not running, the Exchange jobs are prevented from restarting if those services are offline. For more information, see the online Help for the AMAdmin_SchedMaint or AMAdmin_SetResDependency Knowledge Scripts.

Reviewing and Refining the Deployment

Once basic monitoring is underway, it becomes easier to fine-tune thresholds and job intervals, articulate and automate event-response policies, and tailor event notification, data collection, and the user interface to suit your needs. As you refine your deployment, focus on the following goals:

- Managing events and event notification

  For more information about developing and refining your policies for handling events, see the chapter "Managing Events" on page 85.

- Handling data-collection and archiving data

  For more information about developing and refining your data handling policies, see the chapter "Managing Data" on page 99.

- Controlling communication between agent computers and management servers

  For more information about communication between agent computers and management servers, see the chapter "Site Communication and Security" on page 11.

- Managing security and security roles within AppManager

  For more information about controlling access to tasks and configuring security settings, see the chapter "Managing Security for AppManager and Control Center" on page 29.

- Adding management servers and configuring primary and secondary management servers for agent computers

  For more information about setting up primary and backup management servers, see "Configuring a Primary and Secondary Management Server for Windows Agent Computers" on page 25 and "Configuring a Primary and Secondary Management Server for UNIX Agent Computers" on page 25.

- Organizing the computers in your network into meaningful groups

  For more information about using management groups to manage a group of computers, see the *Control Center User Guide for AppManager*.

- Identifying and establishing Knowledge Script Groups, dynamic views, and monitoring policies for the computers in your environment

  NetIQ Corporation recommends implementing policy-based monitoring in a test environment before you implement it in your production environment. For more information about initiating policy-based monitoring, see the *Control Center User Guide for AppManager*.

Chapter 5
Managing Events

Before you deploy AppManager across your enterprise, you need to develop and refine your policies for handling events. For example, you may want certain events to trigger automated responses, such as corrective actions or notifications. This chapter addresses the impact of event notification, strategies for managing events, and the options you should consider when you run Knowledge Scripts that raise events or perform actions.

In determining how you want to handle events in your organization, you need to consider your internal procedures, departmental structure, and management goals. You also need to understand how your event-handling policies relate to AppManager user preferences and the amount of attention you'll need to devote to database management.

Deciding When to Raise Events

Some AppManager events are required for monitoring server health and availability and important application resources. But if you generate too many events, you risk overwhelming your staff, who might then ignore or overlook critical events.

Typically, you generate events when you want to find out what is wrong with the computers on your network and to quickly locate current and potential problems. You might also raise events for visibility and tracking. Some events let you know that a situation occurred and you intend to address it, either by acknowledging the event, running another Knowledge Script to remotely diagnose the problem, or diagnosing the system in person.

Always have a clearly defined purpose when generating events. For example, you may decide to raise events only for critical issues in your environment that need immediate attention, such as computers that are dangerously low on disk space or that have dangerously high CPU usage. If you are unsure of what to expect in your current environment, collect data for a period of time without raising events to determine a reasonable baseline for the computers you monitor. Once you have a better understanding of your environment, you can modify threshold settings and begin monitoring less critical event conditions, servers, and applications. Having a clear purpose and understanding the impact of the events you are generating helps you make informed decisions about when and how to generate and display events and can also help prevent your staff from being besieged by a sudden barrage of events.

Therefore, the first step in defining event-handling policies is to make a list of the specific types of events you need and your core Knowledge Scripts for monitoring basic computer resources. For example, most organizations begin by monitoring CPU, memory usage, disk space, disk I/O activity, network connections or activity, and the availability of specific computers or processes. In addition, many organizations monitor computer hardware components and application-specific resources, such as mailbox size for messaging servers and database connections for database servers. Although your list is likely to change over time, identifying a few specific Knowledge Scripts early on can help you avoid generating more events than you need.

Once you understand the type of events that require you to be notified, you can identify the Knowledge Scripts to raise the relevant events and create jobs to display those events.

Understanding Events and Event Messages

When you run a Knowledge Script that generates events, each time the Knowledge Script runs and detects that a threshold has been crossed or a process is down it generates a parent and child event and detailed information about the event and stores the information in the AppManager repository. Once the information is stored in the repository, you can:

- View event alerts in the TreeView pane using the Operator Console.
- View parent and child events in the Events tab in the Operator Console.
- View detailed information for specific child events in the Message tab in the Event Properties dialog box in the Operator Console.

For more information about viewing and working with events, see the chapter about responding to events in the *Operator Console User Guide for AppManager*.

Event Collapsing and Duplicate Events

When you run a Knowledge Script and enable events, AppManager creates both a parent and child event for the first occurrence of the event condition. For subsequent occurrences, AppManager creates additional child events under the parent event and updates an event counter indicating the number of child events. AppManager can detect unique and duplicate child events. An event is considered a duplicate when it occurs with the same object name, event message, severity, and job ID as a previous event within a certain period of time.

Although duplicate events are typically valid, it isn't useful to receive multiple events caused by the same condition. For example, if you are monitoring a disk every 10 seconds and at 18:00 the disk crosses the threshold you specify, AppManager can generate a new event every 10 seconds. In addition, if you associated an e-mail action with this job, AppManager can send an e-mail message every 10 seconds containing the same information.

By default, AppManager reduces the number of individual events (and actions) you receive by collapsing duplicate events into a single event and incrementing the event counter. In this way, AppManager reduces the "noise" from a recurring or persistent issue. You are still informed that the event occurred multiple times, but you are not overwhelmed with event messages or redundant e-mail messages.

AppManager uses a time limit for collapsing these duplicate events. For example, if an event occurs at 18:00, by default, 20 minutes must elapse in which the condition does not recur before a new event is generated. If an event occurs every 10 seconds, a new event is never displayed; AppManager simply increments the event count. If the event occurs at 18:00 and the time frame is 18:20, when the event occurs again at 18:00:10 the time frame is adjusted to 18:20:10. When the event occurs again at 18:01 the time frame is adjusted to 18:21, and so on.

For an individual job, you can adjust the time interval by selecting **Initial occurrence** in the Advanced tab in the Knowledge Script Properties dialog box. Or you can change the default behavior by setting the Advanced Properties repository preference. Use this preference if your monitoring is critical and you want to receive events and actions on a regular basis until the problem has been resolved. For example, if the event occurs at 18:00 and then again at 18:01, you receive one event showing an event count of 2. Then, if the event occurs again at 18:20, you receive a new event (and action). By default, you would have waited until 18:21 for the new event.

Note
If you acknowledge or close an event and the condition recurs, a new event is generated. Event collapsing only occurs while the original event is open.

Setting Preferences for Event Information

AppManager performance and availability information cannot remain available indefinitely. In addition, displaying too much information in the Operator Console can impede system performance. AppManager repository preferences therefore can help you determine how long to keep event information and set options for archiving events. In addition to repository preferences, you can also set event-handling properties for jobs. For more information about configuring job event settings, see "Using Advanced Event-handling Properties" on page 91.

To help consolidate events and simplify the information displayed in the Operator Console, by default, AppManager collapses duplicate events (events with the same object name, event message, severity, job ID) into a single event. For example, after an event is raised instead of creating new child event entries, duplicate events, associated with the same computer, job, and event condition, are collapsed into the original child event and the child event count is increased.

AppManager collapses duplicate events within a specified time interval (20 minutes by default). You can configure this time interval to begin:

- When the first event is raised. All duplicate events within the time interval (static period of time) are collapsed into one event.
- Each time an event is generated (it is not a static period of time). For example, using the default time of 20 minutes, if a job generates duplicate events every 5 minutes, the 20 minute interval is restarted every 5 minutes, meaning it never effectively expires — unless you set an option for AppManager to ignore events.

After the original child event is closed, or after the event collapsing time interval expires, a new child event is created if the event condition is detected.

In addition, when event collapsing is enabled and a duplicate event is raised for an event you previously acknowledged, by default AppManager changes the status of the acknowledged event to open and increments the event counter. In most cases, this default behavior is appropriate, giving you visibility that even though an event has been acknowledged by an operator it hasn't been closed and the case should be re-opened to resolve the problem. In some cases, however, you may want to leave an acknowledged event in the acknowledged state to indicate someone has responded to the problem but still be notified that the event condition has extended beyond the event collapsing interval. You can choose one of the following AppManager preferences to control this behavior:

- If you want to see a new open child event when the event condition persists beyond the event collapsing window for an acknowledged event, use the **Create a new child event for acknowledged event during event collapsing** preference.
- If you want to increment the child event count only when the event condition persists beyond the event collapsing window for an acknowledged event, leaving the status of the acknowledge event unchanged, use the **Increment the event count only without changing status event during event collapsing** preference.
- You can change the way AppManager displays event information for duplicate events that are raised (and collapsed) after you acknowledge an event.

Note
These options are not available in the Control Center console, though how you set these options in the Operator Console also affect how the Control Center console displays event data.

To change the handling of duplicate events when event collapsing is enabled:

1. Click **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, then click **Event** in the Event options group.
3. Click the event handling preference you want to use.
   - Click **Create a new child event for acknowledged event during event collapsing** if you want AppManager to create a new child event and sets the status of the event to open.
   - Click **Increment the event count only without changing status event during event collapsing** if you want to leave the status of an event as acknowledged but also to increment the event count to indicate there has been a new occurrence of the event.

By default, AppManager keeps event information available for display in the TreeView pane, **Events** tab, and **Message** tab indefinitely. Over time, events can accumulate in the Operator Console, which can affect the performance of the Operator Console, or they can become out-of-sync with your jobs. For example, if you delete jobs but not their associated events, your Events list will include events for which no job information is available. While this may not be a problem if you manage a small number of jobs and events, it can become a problem when the number of jobs and events increases. Managing the list of events and identifying useful and relevant events can become increasingly difficult.

To prevent this situation, in most cases you should plan to remove events when you delete the associated jobs. AppManager provides a repository preference to help you manage event information for deleted jobs.

Note
This option is not available in the Control Center console, though how you set this option in the Operator Console also determines whether or not events are deleted when deleting jobs in the Control Center console.

To delete all event information when a job is deleted in the Operator Console:

1. Click **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, then click **Event** in the Event options group.
3. Click **Remove associated events when jobs are deleted**.

   When you delete a job in the Operator Console, AppManager removes the associated event information from the AppManager repository. However, if you archive events, this information is still available in the event archive tables in the repository.

   Note
   If you do not remove events when jobs are deleted, you should periodically remove events from the repository manually. If you do not remove them, these "orphan" events without jobs associated with them will consume database resources and may impact performance. For information about removing events from the repository, see "Removing Archived Data and Events" on page 124.

4. Click **OK** in the Preference - Event Options dialog box, then click **OK** in the Preferences dialog box.

Acknowledging and Closing Events

To respond to an open event and turn off the event alert, you need to either acknowledge or close the event. How you respond to an event and use the Acknowledge or Closed status depends on your system management policies. For example, you might immediately acknowledge the event, check the server, and try to solve the problem, or you might acknowledge the event and run other Knowledge Scripts to collect data or to further diagnose the problem.

Acknowledging the event turns off the event alert and changes the status of the event in the Events tab to "Ack." When you have resolved the problem that caused an event, you can then close the event. You do not need to acknowledge an event before closing it. However, to prevent accidental deletion of open or unresolved events, you must close an event before you can delete it.

To acknowledge or close events, you can:

- Individually acknowledge or close child events in the Events tab (or acknowledge or close all child events at once by acknowledging or closing a parent event).
- Acknowledge or close all events associated with an application server, a group of servers, or all servers in a view in the TreeView pane.
- Individually acknowledge or close an event after viewing detailed information in the Message tab in the Event Properties dialog box.

AppManager also provides a repository preference to automatically close events based on their severity level. This preference can be useful when:

- You want to save historic information about an informational or diagnostic event but do not want to manage event status.
- You want to know that an event occurred, but you don't need to address the issue right away. You would therefore would like to automatically close, but not delete, the event.

For example, if you are raising an event when a condition no longer exists and you receive an informational event indicating that the condition ended, you may want to automatically dismiss (close but not delete) this type of event.

Note
This option is not available in the Control Center console, though how you set this option in the Operator Console also determines whether or not events with a specified severity are closed in the Control Center console.

To automatically close events:

1. Click **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, then click **Event** in the Event options group.
3. Click **Automatically close event when severity is greater than N**.
4. Click **OK** in the Preference - Event Options dialog box, then click **OK** in the Preferences dialog box.

Note
Use caution when setting this preference. Depending on the value you set for this preference, you can accidentally acknowledge and close other events. NetIQ Corporation recommends setting a unique severity level for this preference (for example, a special value you do not typically use, such as 40) and if you have other Knowledge Scripts using this severity level, set their event severity level to another value, such as 39.

Understanding Event Archiving

By default, AppManager keeps event information available for display indefinitely and stores the event information in the AppManager repository in event and event archive tables. Because this information can accumulate over time and put a strain on your resources, you should manage the archiving, purging, and removing of events carefully. For example, moving old events to event archive tables when events are closed or have been kept for a specified amount of time can help keep the Operator Console clear of unimportant information and save you time by dealing with these less severe events automatically.

QDB preferences let you change the default period for keeping events available in the Operator Console and automate your event handling.

To change the setting for keeping event information:

1. Click **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, then click **Event** in the Event options group.
3. Check **Move aged events to archive tables in repository when**.
4. Select an option to archive events based on status, severity, or a period of time. You can choose to archive events:
   - When an **Event is closed**
   - When an **Event stays open status for** a certain number of days
   - When an **Event stays closed status for** a certain number of days
   - When a **Closed event severity is greater than** a specific severity level

   For more information on these options, consult the Help.

5. Select the time, in days, for purging archive events from the repository. You can choose to purge events:
   - When the archived **Event has been kept for** more than a certain number of days in the repository
   - When the **Oldest events exceed number of records** allowed in the repository

   For more information on these options, consult the Help.
6. Click **OK** in the Preference - Event Options dialog box.
7. Click **OK** in the Preferences dialog box.

Using Advanced Event-handling Properties

In addition to the repository preferences for event handling and archiving, AppManager provides preferences to specify how jobs generate events. You can set these preferences for individual jobs by clicking the **Advanced** tab in the Knowledge Script Properties dialog box. Or you can set the default behavior for these preferences by modifying Advanced Properties repository preferences. These advanced properties for jobs and events specify preferences for:

- Configuring Event Collapsing for Jobs
- Adjusting Consecutive Intervals
- Raising an Event When a Condition No Longer Exists

Note
Any default values you set for how the Operator Console generates events will not apply to the Control Center console. As a result, if you open the same Knowledge Script in both the Operator Console and Control Center console, the default values on the Advanced tab of the Knowledge Script Properties dialog box may not be the same. If you are going to create jobs from the same Knowledge Script in both the Operator Console and the Control Center console, be sure any settings on the Advanced tab of the Knowledge Script Properties dialog box are the same in both consoles.

Configuring Event Collapsing for Jobs

When you run a Knowledge Script and raise events, AppManager creates both a parent and child event for the first occurrence of the event condition. For subsequent occurrences, AppManager creates additional child events under the parent event and updates an event counter indicating the number of child events.

To change the default behavior for collapsing duplicate events:

1. Select **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and click **Advanced Properties** in the Knowledge Script options group.
3. Click **Collapse duplicate events into a single event**.
4. Set the time interval for collapsing duplicate events in the **Time interval for event collapsing** field.
5. Click **Initial occurrence** to indicate that you want to use a static period of time to regulate event collapsing.
6. Click **OK** in the Preference - Knowledge Script Advanced Options dialog box. Then click **OK** in the Preferences dialog box.

NetIQ Corporation recommends using event collapsing and selectively setting the number of consecutive occurrences. When setting an event collapsing interval, keep the following in mind:

- The schedule interval must be set to at least 10 minutes.
- If the schedule interval is shorter than the collapsing interval and the job detects an event every interval, new identical events are collapsed into the same child event.
- If the job schedule is longer than the collapsing interval, you will see multiple child events even though the events are identical.
- If you set the number of consecutive occurrences to a value greater than 1, be sure to consider the time schedule interval. For example, if a job runs once every 12 hours and you set the number of consecutive occurrences to 3, you won't be notified of the event until the third occurrence — 36 hours later.
- Child events accumulate in the database. Try to strike a reasonable balance between timely notification and eliminating trivial or redundant events.

Adjusting Consecutive Intervals

If you only want to receive an event if a condition crosses a threshold a certain number of times, you can specify the number of times a consecutive duplicate event (events with the same object name, event message, severity, job ID) must occur before AppManager generates an event message. Using this approach you can hide trivial spikes that can occur when a Knowledge Script runs at frequent intervals. Events triggered by these temporary spikes are not always useful and your operations or administrative staff can spend vital time responding to events for conditions that do not cause any disruption to your environment.

For example, say you are monitoring CPU and it hits 99%. You might not consider this a problem because you know the system has a heavy processing load, or because this represents an uncharacteristic spike in the activity while the system is performing a specific processing task. In this case, you might not want to receive an event message. However, if CPU is at 99% for 10 minutes, you might have a problem.

By default, AppManager alerts you every time CPU crosses the threshold you specify. You have two options for changing this:

- Set the number of times you want the condition to occur and the number of iterations for the job in the **Raise event if condition occurs N times within N job iterations** field on the **Advanced** tab in the Knowledge Script Properties dialog box for individual jobs.
- Change the default behavior by setting the Advanced Properties repository preference. See the instructions below.

  For this example, you might specify that you want to receive an event only if CPU is over 99% 10 times in 2 job iterations.

  Note
  Typically, the more frequently the job is scheduled to run, the higher you can set the number of consecutive intervals before raising an event. NetIQ Corporation recommends setting this preference in the range of 3 to 5 occurrences for volatile performance statistics.

To change the default behavior for consecutive intervals:

1. Select **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and then click **Advanced Properties** in the Knowledge Script options group.
3. Specify the number of times you want the condition to occur and the number of iterations for the job in the **Raise event if condition occurs N times within N job iterations** field.
4. Click **OK** in the Preference - Knowledge Script Advanced Options dialog box. Then click **OK** in the Preferences dialog box.

Raising an Event When a Condition No Longer Exists

For some event conditions, it is useful to raise an event when the condition is first detected and then raise a second event when the condition no longer exists. For example, you can use this option if you want to be automatically notified when a problem that raised an event has gone away.

To illustrate, suppose you want to receive a severity 5 event when CPU utilization reaches 99% and then an informational event message with a severity level of 35 when CPU utilization returns to 10%. In this case, you have two options:

- You can select **Generate a new event when original event condition no longer exists** and specify a severity level (for example, 35) in **Severity of new event** on the **Advanced** tab in the Knowledge Script Properties dialog box for an individual job.
- You can change the default behavior by setting the Advanced Properties repository preference. See below for instructions.

You can also select **Automatically close original event** to close the original severity 5 event when CPU utilization falls below the threshold you set.

For example, if a job detects that physical memory usage has exceeded the threshold you set, AppManager raises an event. This event condition continues until memory usage falls below the threshold. Because at this point the original event condition no longer exists, AppManager automatically closes the original event and raises a new informational event with a severity you have specified.

Note
In general, you should set the severity level for the informational event to a unique or rarely-used severity and use a severity level that is clearly distinguishable from the original event.

To change the default behavior for raising events when an original event condition no longer exists:

1. Select **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and then click **Advanced Properties** in the Knowledge Script options group.
3. Click **Generate a new event when original event condition no longer exists** and specify a severity level in **Severity of new event**.
4. Click **Automatically close original event**.
5. Click **OK** in the Preference - Knowledge Script Advanced Options dialog box, then click **OK** in the Preferences dialog box.

Notifying Individuals of Events

Event notification policies vary from one organization to another. In some cases, it is a restricted process, in others it is highly automated. In general, it is a good idea to carefully define a notification strategy tailored to your unique requirements. For example, do you want to:

- Send an e-mail after an event?
- Send a page after an event?
- Redirect a message after an event?

This section describes strategies you should consider before implementing event notification.

Sending an Email Event Notification

AppManager provides several options for sending an e-mail message in response to an event:

- Messaging API (MAPI) mail messages
- Simple Mail Transfer Protocol (SMTP) mail messages
- Lotus Notes mail messages

To determine which type is best for your environment, consider the following:

- **Your e-mail software**. For example, to send MAPI mail, you must have Microsoft Exchange or Outlook installed. To send SMTP mail, you must have an SMTP mail server. For Notes mail, you must have a Lotus Domino server and the Notes mail client installed.
- **Where you are sending the e-mail**. For example, consider how you have set up mailing lists, aliases, and network connectivity. If you defined e-mail aliases for different groups and the addresses are all internal, you can use MAPI or Notes mail so that you are not sending messages through your Internet gateway to deliver internal mail.
- **Where the e-mail originates** (from a central location or at each monitored server). Consider the requirements for each type of mail and how those requirements suit your environment. For example, MAPI mail requires you to install a MAPI client (such as Outlook) on the server that sends the mail message. Notes mail must be sent from a Domino server, so the mail must be sent from each server you are monitoring.

In most cases, SMTP mail provides the best method for e-mail notification. It does not require a client to be installed, so it can be easily configured to run from either the management server or the managed computers you are monitoring. The major drawback to using SMTP mail is that your SMTP gateway can have security rules that limit the users who can send mail through the gateway or the servers from which SMTP mail messages can originate. You should check with your SMTP gateway administrator to ensure the user account(s) that the AppManager agent services run under have permission to send mail on the servers from which the mail originates.

Depending on your environment you can use more than one mail method for event notification. For example, assume you have several Exchange servers you want to monitor on one continent and your SMTP gateway on another continent and IT staff that supports the Exchange servers is located in the same location as the Exchange servers, but the main administrative staff is located elsewhere. You might want the events associated with the Exchange servers forwarded to the IT staff using MAPI mail to avoid sending mail to another continent to reach the SMTP gateway then back to the local site for notification. Events that need to reach the main administrative staff might be configured to send SMTP mail and be routed through the SMTP gateway for distribution.

Sending a Page as an Event Notification

Depending on the paging system you use and the types of messages it can receive, there are several ways you can use AppManager to forward event information or custom information to the paging system. In most cases, you can send event information to a pager by:

- Using the Action_Page Knowledge Script
- Using SMTP Mail Messages
- Using MAPI or Lotus Notes Mail Messages
- Using Log Files or Other Sources to Send Messages

Using the Action_Page Knowledge Script

The Action_Page Knowledge Script relies on the `%SYSTEMROOT%/NetIQpage.ini` file located on each monitored server. The `NetIQpage.ini` file is pre-configured with information necessary to send paging messages to several common paging systems.

To use Action_Page, install the client software for the paging application on the server where you run Action_Page. You typically install the client software on the AppManager management server and the management server initiates the paging action.

For more information on how to edit the `NetIQpage.ini` file and use the Action_Page Knowledge Script, select the Page Knowledge Script in the Action tab in the Operator Console and press `F1`.

Using SMTP Mail Messages

Some paging applications accept SMTP mail messages to send pages. If your paging application accepts SMTP mail, you can configure an action to be initiated on the management server or on the individual servers being monitored.

To use SMTP, configure the Action_SMTPMail Knowledge Script with the recipient's name in the proper format. Different paging applications have their own formatting requirements for SMTP mail recipients. For example, the recipient's name parameter can be an individual's name, alias, group, pager number, or PIN depending on the paging application (such as, `5443221@skytel.com` or `bridge12@samplepage.com`).

For the SMTP server machine name, type the name of the SMTP gateway server. The SMTP server sends the message to the appropriate paging system domain and the domain delivers the e-mail message to the pager.

Using MAPI or Lotus Notes Mail Messages

Some paging applications accept MAPI or Lotus Notes mail messages to send pages. As mentioned in "Sending an Email Event Notification" on page 94, both MAPI and Notes mail messages have special requirements for sending mail:

- **MAPI mail**. To use MAPI mail, you must install and configure client software. It is usually best to set up MAPI mail as an action that executes on the management server.
- **Notes mail**. To use Notes mail, you must run Notes from the Domino server(s) you are monitoring.

Using Log Files or Other Sources to Send Messages

Some paging applications can accept input from other sources such as ASCII log files or the Windows Event Log. For these applications, you can use the Action_WriteMsgToFile or the Action_NTEventLog Knowledge Script, respectively. You should review your paging application's documentation and the AppManager Help to see how to properly configure these Knowledge Scripts.

In addition, some unsupported Knowledge Scripts that interface with paging applications without using the `NetIQpage.ini` are available from the Knowledge Depot on the NetIQ Support Web site.

You can also write custom Knowledge Scripts that use the paging system's API to perform tasks. If you decide to write your own scripts and create your own interface, see the *Developing Custom Knowledge Scripts Guide for AppManager* for information.

Sending Event Information to Another Console

In larger organizations, administrators often integrate AppManager events into another management console or redirect events to a Help Desk application or tracking system. To help make this as transparent as possible, AppManager supports a variety of "connectors" to the most common event management consoles and help desk applications. For example, AppManager has connectors for:

- Computer Associates Unicenter NSM
- HP OpenView Network Node Manager (NNM)
- HP OpenView Operations
- Micromuse Netcool
- Microsoft Operations Manager (MOM)
- Tivoli Enterprise

You can use these connectors to integrate events into the management console applications you want to use. Connectors are installed on AppManager management servers, and they forward events to the management console as events are sent from the agents. For more information on AppManager connectors, visit the NetIQ Web site or contact a NetIQ technical representative.

If NetIQ Corporation does not currently have a connector for your management console application, you can often use Action Knowledge Scripts to transfer the event information to the application. For example, you can use Action Knowledge Scripts to:

- Send an SNMP trap
- Save information to the Windows Application Log
- Write information to an ASCII log file

You can also create your own custom Knowledge Scripts and call a COM object or use a command-line interface to transfer the event information to the application of your choice.

Using SNMP Traps

Sending an SNMP trap is a simple and cost-effective solution for transferring event information to another application. It does not require any additional client software to configure or maintain or any additional client licenses for your event console or help desk application. If you use SNMP traps, you can run them as automated actions associated with a Knowledge Script, and configure two network paths for events to travel to separate applications.

Using Log Files or Command-line Interfaces

You can configure automated Knowledge Script actions that write to the Windows Event Log or to an ASCII log file, or send command-line arguments to run on the managed computer or the management server. These methods usually require you to install, configure, and maintain a client executable for the product to which you want to connect on each managed computer that will be raising events. Therefore, you may incur additional licensing costs.

For these reasons, if you decide to forward events through the Windows Event Log, ASCII log files, COM, or a command-line interface, you may want to configure the event-forwarding action to take place at the management server to reduce costs.

Triggering Corrective Actions for Events

Before you set your Knowledge Scripts to perform an action when AppManager raises an event, you should fine-tune the Knowledge Script threshold settings to determine the appropriate thresholds and actions for your environment. You don't want to receive a high volume of e-mails or pages for events caused by thresholds that have been set too high or too low. When you're ready to define actions for events, select a notification method: MAPI mail, SMTP mail, or a third-party paging software.

Most actions involve event notification and visibility. However, Action Knowledge Scripts can also be used to automatically correct problems. Typically, corrective actions run a command or execute SQL statements on the managed computer in response to an event. Remember that the corrective action must take place on the managed computer, not on the management server.

The Action_DosCommand, Action_DumpTran, and Action_RunSql Knowledge Scripts are good examples of Knowledge Scripts that run corrective actions.

To run a corrective action on a specific managed computer:

1. Open the Properties dialog box for the Knowledge Script monitoring job. Click **New** on the **Actions** tab to create a new action.
2. Select an Action Knowledge Script from the **Action** drop-down list. If you are monitoring a Windows computer, UNIX actions are not available.
3. Select **MC** from the **Location** list to specify that you want AppManager to run the action on the managed computer.

   Note
   Some actions must be run on the managed computer because they perform an operation on that computer. Other actions can be run on either the managed computer or the management server.

4. Configure the action **Type** to run the first time an event is generated (a unique event), after a duplicate child event is created a specified number of times, or when the event condition no longer exists. When monitoring UNIX and Linux computers, the Type is always **New Event**.
5. Select an action schedule from the **Schedule** drop-down list to specify the available hours during which the action can run. When monitoring UNIX and Linux computers, action schedules are not applicable.

6. Click **Properties** to set the properties for the Action Knowledge Script.

   Most actions require you to set some additional properties. For example, if you select an e-mail action you need to specify an e-mail recipient. For more information about Action Knowledge Scripts and their parameters, see the AppManager Help.

7. Click **OK** in the Action Properties dialog box, and then click **OK** in the Knowledge Script Properties dialog box.

When configuring an automated Knowledge Script action, check the following:

- Whether the account the AppManager agent service (`NetIQmc`) runs under (whether `LocalSystem` or a specific user account) has permission to execute the desired action on the computer where the action runs.
- Whether the action requires environment variables to be set and whether those environment variables run properly. If your action requires environment variable changes, determine whether the environment variables are instantiated by the Action Knowledge Script or are preset as system variables in the System Control Panel for the computer. For example, using Action_DosCommand to call the Attention! executable (`attn.exe`) requires you to set the `attnsrv=`*`servername`* environment variable on the computer running the action.
- Whether the command you want to run requires user interaction, such as input. For example, if you decide to run a command to delete files, use `del /Q` rather than just `del` to ensure that any wildcard deletions take place without prompting for confirmation.

Chapter 6
Managing Data

Before you deploy AppManager across your entire enterprise, you need to develop and refine your data handling policies. In determining how you want to manage AppManager data in your organization, you need to consider your group's internal procedures, department structure, and management goals. For example, if your manager has asked you to supply a weekly performance summary for your Network Operations Center (NOC), you need to collect data for those servers and run reporting Knowledge Scripts at least once each week. It is important to understand how your data-handling policies will affect the availability of the specific charts, graphs, and reports you are interested in, as well as the level of database management you will need to perform.

The topics in this chapter address the impact of data collection and the options you should consider when planning to run Knowledge Scripts that collect data.

Deciding When to Collect Data

It is important that you collect only the data you require for an adequate understanding of your systems or for charts, graphs, and reports. If you collect too much data, it can be hard to manage and require constant database maintenance.

Typically, you collect data when you want to:

- Identify normal operating environment performance values or baseline performance values
- Diagnose problems (for example, to view the top CPU consumers after finding that processes are failing)
- Create data streams for real-time charts or graphs to identify short-term trends or compare performance data
- Store historical information for trend analysis, capacity planning, or service-level reporting

Although most organizations collect data for a combination of these reasons, it's important to consider which jobs you use to collect data, how frequently the jobs run, how much data is returned, and how you will use the data in specific charts, graphs, and reports. Don't collect data for all jobs; only collect it from the jobs from which you need charts, graphs, and reports.

As an example, consider a Knowledge Script that checks server connectivity. You might want to run this monitoring job every 5 minutes to ensure prompt notification if the server connection goes down. However, with data collection enabled, you could run the script less frequently, or only during business hours. By adjusting the schedule for data collection, you avoid cluttering the database with unnecessary information.

In some cases, getting a data point every five minutes is useful. Often, however, frequent data collection doesn't really provide you with any more useful information than if you were collecting the data point at a longer interval. For example, CPU and memory usage can change significantly in a five-minute period, but logical disk space is not likely to change as frequently and can be effectively monitored at 12- or 24-hour intervals.

As a general guideline, when you monitor values that can change frequently, you should collect data more frequently so that a statistical analysis of the data can provide a realistic picture of activity on the monitored system. For example, if you collect data on CPU utilization once an hour, or even every 15 minutes, you can capture data spikes or lulls that do not accurately reflect performance, resulting in reports that provide an inaccurate view. When monitoring values that change more slowly, like logical disk space or database growth, you can collect data at longer intervals and still achieve an accurate assessment.

The best practice is to have a clearly defined purpose for collecting the data, such as producing a weekly report of server availability or the top ten e-mail users. It is also important to keep in mind that every data point stored in the repository requires more data space and more database management. Having a clear purpose and understanding the impact of the data you are collecting helps you make the most intelligent decisions about when and how to collect data.

To illustrate the importance of this point, consider a Knowledge Script such as NT_TopCpuProcs, which reports the CPU utilization of all processes running on a managed computer. In most cases, you should only run this job when investigating a problem on the computer. If you enable data collection, you should not run this job on multiple computers or on a regular schedule because of the potential overhead in database space and performance involved in returning so much data.

Therefore, the first step in defining your data-handling policies is to make a list of the specific charts, graphs, and reports you need and the period of time for which you want to view information. For example, you should determine whether you are interested in real-time charts and graphs only on a daily basis or want to keep charts and graphs active for one or two weeks. Similarly, you should consider whether you are interested in hourly data or weekly summaries and in individual data points or averaged values. Although your list is likely to change over time, identifying a few specific charts, graphs, and reports early on can help you collect only the data you need.

Understanding Data Collection for Charts, Graphs, and Reports

When you run a Knowledge Script that collects data, each time the Knowledge Script runs, it collects an individual data point and stores the information in the AppManager repository. Once the information is stored in the repository, you can:

- Display it in **real-time charts and graphs** using the Operator Console, Control Center console, or Chart Console. Real-time charts represent active data streams that are visually updated as new data points are received.
- View it in **static HTML reports**, generated using report Knowledge Script templates, using the Report Viewer. Generated reports represent a snapshot of data from the repository for the period of time you specify. Generated reports can include charts, but the charts are static and specific to the report period you define.

Because you cannot keep information available indefinitely, AppManager provides repository preferences to help you manage how long to keep data for real-time charts and graphs and options for archiving and summarizing data for reports.

Setting Preferences for Real-time Charts and Graphs

By default, AppManager keeps data points available in a data table for immediate display in real-time charts and graphs for eight days, and every six hours removes any data points that are more than eight days old. One way to think of this is that every six hours the oldest data points "expire" and are no longer available for real-time charting and graphing. However, the information is still available in data archive tables for generated reports.

Because real-time charts and graphs put greater strain on your database resources than archived or summarized data, you should manage the expiration and removal of data points carefully. If you increase the number of days to keep data available for real-time charts and graphs, you increase the amount of database space you need to hold the extra data points and resources required to retrieve and display the additional data. Conversely, if you decrease the number of days to keep data points available, you free up database space and resources but can risk missing crucial information in your real-time charts and graphs.

AppManager's default repository preference settings are intended to give your organization roughly one week's worth of data for real-time charting and graphing.

Note
This option is not available in the Control Center console, though how you set this option in the Operator Console also determines the amount of data available to the Control Center console for real-time analysis and diagnosis.

To change the default period for keeping data points in the repository:

1. Click **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and then click **Data** in the Data options group.
3. Set **Default period to keep points in data table** to the maximum number of days you want AppManager to keep data points available for real-time charts and graphs.

   Set this preference to reflect the maximum number of days that information should remain available for real-time analysis and diagnosis. Reducing this value reduces data overhead and improves database performance. Changing this preference does not affect the information available for reports.

   Note
   This preference only affects new jobs. You must update existing jobs individually to change the number of days that they keep data points in the data stream.
4. Specify the time, in hours, to remove old or "expired" data points from the repository in **Time interval to purge old points in repository**.

   AppManager runs the NetIQ `PurgeData QDB` scheduled task at the interval you specify and deletes any data points older than the value you specified in **Default period to keep points in data table**. For more information about the impact of these settings on repository maintenance, see "Managing a QDB" on page 107.
5. Click **OK** in the Preference - Data Options dialog box.
6. Click **OK** in the Preferences dialog box.

Understanding Data Archiving and Reporting

By default, the data you collect with Knowledge Scripts is stored indefinitely in the AppManager repository in data archive tables for reporting purposes. Because generating static and historical reports places less strain on your database resources than real-time charting and graphing, AppManager does not require you to remove this data at any set period of time.

Although keeping data available for reporting for an indefinite period of time gives you greater flexibility for performing historical analysis of your environment, you must manage the data carefully to ensure the AppManager repository does not grow too large and become unstable or adversely affect the performance of the SQL Server. AppManager provides repository preferences to help you manage this archive data and prevent the data from impacting database performance.

Summarizing the Data Used for Reports

To prevent the archive data used for reporting from impeding database performance, you can set AppManager repository preferences to periodically archive and aggregate data information.

To set repository preferences for archived data:

1. Select **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and then click **Data** in the Data options group.
3. Click **Move archived points to aggregate table** to summarize archive data and move the summarized data to aggregate data tables.

   Set this preference to instruct AppManager to summarize archive data, move the data to aggregate tables in the AppManager repository for historical reporting, and purge the data from the data archive tables. By default, information remains in the data archive table indefinitely.
4. Specify the time, in months, to move the summarized data to aggregate tables in **Move points after N months** (for example, the three most recent months).

   Note
   Move points after N months requires you to keep data points in the data archive tables for at least one month. A minimum of one month's worth of data must be stored for every Knowledge Script collecting data.
5. Click **OK** in the Preference - Data Options dialog box.
6. Click **OK** in the Preferences dialog box.

When you set these repository preferences, AppManager treats archive data as follows:

- For all archive data older than the number of months you specify in **Move points after N months** (for example, three months), AppManager calculates an hourly average, minimum, maximum, sum, and count value. AppManager then moves these hourly values to the hourly aggregate table (ArcAvgHourlyData). Each hour of data is then represented by five data points (avg, min, max, sum, count). The hourly aggregate table keeps three months of data.
- For all hourly aggregate data older than three months, AppManager calculates a daily average, minimum, maximum, sum, and count value. AppManager then moves these daily values to the daily aggregate table (ArcAvgDailyData). Each day of data is then represented by five data points (avg, min, max, sum, count). The daily aggregate table keeps six months of data.
- For all daily aggregate data older than six months, AppManager calculates a monthly average, minimum, maximum, sum, and count value. AppManager then moves these monthly values to the monthly aggregate table (ArcAvgMonthlyData). Each month of data is then represented by five data points (avg, min, max, sum, count). The monthly aggregate table keeps data indefinitely.

Once AppManager moves information from the source table to the destination table, it deletes it from the source table. AppManager performs hourly and daily aggregations once a week and the monthly aggregations once a month.

Notes

- Although summarizing and moving archive data helps decrease the database requirements while preserving historical information, you should periodically back up and purge the data from the QDB. For more information about backing up and removing data, see the chapter ""Managing a QDB" on page 107."
- These data aggregation options are not available in the Control Center console, though how you set them in the Operator Console also affects the data available for reporting in the Control Center console.

Disabling Data Archiving

Some organizations are only interested in real-time analysis of their environment and do not require saving data for static reports. If you don't need access to the historical data at a later time, you can choose not to archive data points at all. Eliminating archive data saves database space and can improve database performance in some environments.

If your primary interest is in current data (what's happening right now or in a small window of time such as daily), you can choose not to archive data and to frequently purge old data points from the repository to keep your database requirements small. You can instead specify that AppManager should not write data to the data archive table (`ArchiveData`) by using QueryAnalyzer to update the ArchiveData entry in the GlobalPref table.

To update the ArchiveData entry:

1. Open the GlobalPref table in SQL.
2. Find the `ArchiveData` entry with an ID=3.
3. Set the ArchiveData entry value to 0. For example: `UPDATE GLOBALPREF SET VAL=0 WHERE ID=3`. AppManager does not write the data to the data archive table.

Using Advanced Data-handling Properties for Jobs

AppManager provides preferences for you to specify how you want your jobs to collect data. You can set these preferences for individual jobs by clicking the Advanced tab in the Knowledge Script Properties dialog box or you can set the default behavior for these preferences by modifying Advanced Properties repository preferences. You can set preferences for collecting detail data, the schedule for collecting data, and collecting data in response to an event.

Collecting Detail Data

Every time you run a Knowledge Script and collect data, AppManager returns a data point value (for example, 50) and a detailed description (for example, CPU utilization is 50%). In general, when you view the data in charts, graphs, and reports, you are usually focused on the value and not the details. However, this can depend on the amount of detail in the detail message and the Knowledge Scripts you are running. For example, the NT_TopCpuProcs Knowledge Script includes a list of the processes consuming the most CPU in its detail message, and you might need this additional information.

To reduce the amount of space taken up by the data in your repository, use the options available on the Advanced tab of the Knowledge Script Job Properties window to configure AppManager to collect data details differently for charts and reports:

- For charts and graphs, the **Collect data details with data point** preference is selected by default to collect the value of the monitored resource and detailed information, such as server name and collection time.
- For reports that display detail data, such as ReportAM_DetailData, the **Do not archive data detail** preference is selected by default to only collect the value of the monitored resource; detailed information, such as server name and collection time, is not collected.

Use the **Advanced** tab in the Knowledge Script Properties dialog box to configure detail data collection for individual jobs. Or you can change the default behavior by setting the Advanced repository preference.

Note

Changing the default behavior for data collection in the Operator Console does not affect default settings for the Control Center console.

To change the default behavior for collecting data details:

1. Select **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and then click **Advanced Properties** in the Knowledge Script options group.
3. Select an option to specify whether to collect data details:

Select	To
Collect data details with data point	This option only applies to data collection for AppManager graphs and charts. Deselect this option to only collect the value of the monitored resource; detailed information, such as server name and collection time, is not collected. This option is selected by default.
Do not archive data detail	This option only applies to data collection for AppManager reports that display detail data, such as ReportAM_DetailData. Select this option to only collect the value of the monitored resource; detailed information, such as server name and collection time, is not collected. This option is selected by default.

4. Click **OK** in the Preference - Knowledge Script Advanced Options dialog box.
5. Click **OK** in the Preferences dialog box.

Modifying the Schedule for Collecting Data

When you run a Knowledge Script and collect data, AppManager returns a data point value for every job iteration, storing each data point in the repository–requiring more data space and more database management.

In some cases, collecting a data point at every job iteration may not provide useful information. If you want to run a Knowledge Script on a frequent basis (for example, every 5 minutes) but don't need to collect and store a data point for each iteration, you can set the **Collect data every N job iterations** and **Calculate average** preferences to reduce overhead and, in some cases, more accurately reflect real system performance.

Instead of collecting 12 data points per hour (while the Knowledge Script is running every five minutes), AppManager collects a data point every n iterations. Fewer data points are stored in your database. Each data point is an average of all the values collected in each of the n iterations. With values averaged over several iterations, you can see a more accurate reflection of system performance.

You can set these preferences on the **Advanced** tab in the Knowledge Script Properties dialog box for individual jobs. Or you can change the default behavior by setting the Advanced Properties repository preferences.

Note
Changing the default behavior for data collection in the Operator Console does not affect default settings for the Control Center console.

To change the default behavior for collecting and averaging data:

1. Select **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and then click **Advanced Properties** in the Knowledge Script options group.
3. Set the number of times a job should run before collecting a data point in the **Collect data every N job iterations** preference in the Data options group.
4. Click **Calculate average** in the Data options group.
5. Click **OK** in the Preference - Knowledge Script Advanced Options dialog box.
6. Click **OK** in the Preferences dialog box.

Collecting Data in Response to an Event

Each time you run a Knowledge Script and collect data, AppManager returns a data point. However, if you are more interested in getting information to diagnose a problem and not for charting, graphing, and reporting, you can set repository preferences to collect data in association with event conditions. Using these preferences, you can collect data when the server, application, or process you are monitoring crosses the threshold you specify and raises an event.

For example, you can run the NT_TopCpuProcs Knowledge Script to raise an event if CPU usage on a computer exceeds a 90% threshold. Set this job to **Start collecting data when an event is generated** because when CPU utilization reaches 90%, you want to know what processes are responsible and create a report that lists the causes of the high CPU. AppManager only creates a data point when the threshold you specify (in this case, 90%) is exceeded. This prevents your database from filling up with information you might never use.

You can then use the **Stop collecting data when event condition no longer exists** preference to stop data collection after you fix the problem that generated the event.

You can set these preferences on the **Advanced** tab in the Knowledge Script Properties dialog box for individual jobs. Or you can change the default behavior by setting the Advanced Properties repository preferences.

Note
Changing the default behavior for data collection in the Operator Console does not affect default settings for the Control Center console.

To change the default behavior for collecting data only when an event condition exists:

1. Select **File > Preferences** in the Operator Console.
2. Click the **Repository** tab, and then click **Advanced Properties** in the Knowledge Script options group.
3. Click **Start collecting data when an event is generated** in the Data options group.
4. Click **Stop collecting data when event condition no longer exists** if you want jobs to stop collecting data when the event condition no longer exists.
5. Click **OK** in the Preference - Knowledge Script Advanced Options dialog box.
6. Click **OK** in the Preferences dialog box.

Chapter 7
Managing a QDB

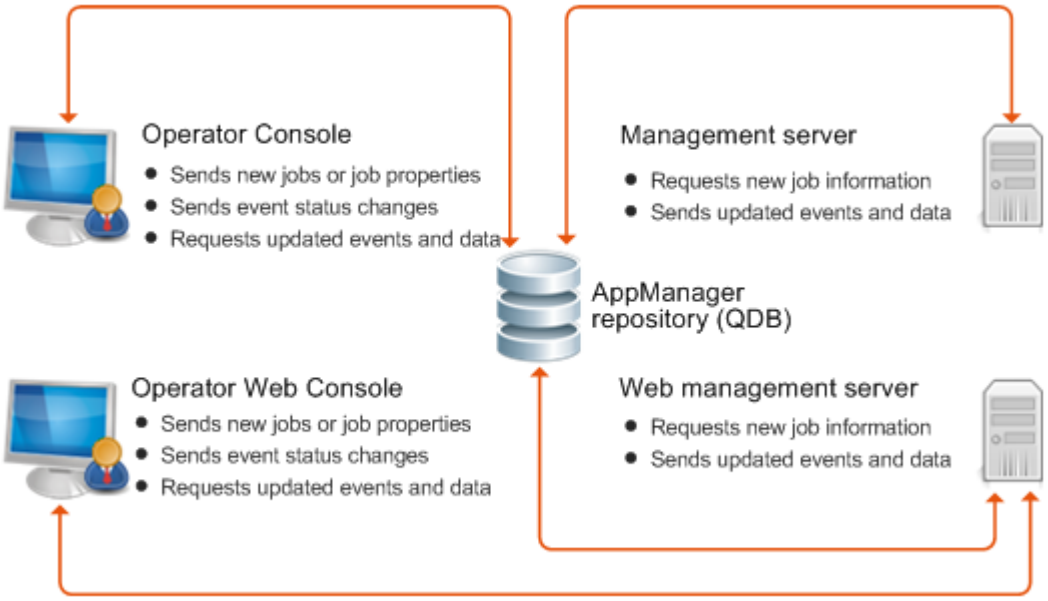
This chapter provides an overview of the information stored in the QDB and describes how to perform routine database maintenance. For more information about how to perform any of these tasks using Microsoft SQL Server Management Studio or other SQL Server tools, see your Microsoft SQL Server documentation.

Understanding the AppManager Repository

The QDB is a SQL Server database that stores information about the following:

- The jobs you run
- The events and data that AppManager collects
- The managed computers that AppManager monitors
- The options you set for viewing and managing jobs, events, and data
- Stored procedures that AppManager uses in the background to perform the database operations that you request
- Scheduled jobs to perform certain maintenance tasks at set intervals

The management server, Operator Console, and Control Center send information to and request information from the database using SQL ODBC connections.



Understanding the Tables

The AppManager repository includes more than 100 database tables to store details about the computers you monitor, the jobs you run on each computer, the job properties you set, the events, and the data that AppManager collects. In most cases, you only need to be familiar with a few of these tables to plan your maintenance and backup strategies.

The tables are as follows:

Table Name	Information stored
Event	Basic event identification, including: • Numeric identifiers for parent and child events • Name of Knowledge Script and numeric identifier for job that raises the event • Name of computer that raised event • Event severity, time of last event occurrence, and short message associated with event Additional event information is stored in related tables linked to this table through the EventID field. Related tables include: • EventDetail • EventDetailAction • EventHistory • ArchiveEvent • ArchiveEventDetail
DataHeader	Data stream identification and related properties, including: • Numeric identifier for the data stream • Numeric identifiers for the parent and child job collecting the data • Name of Knowledge Script collecting the data • Numeric identifier and name for the computer running the job • Short description (legend) associated with data stream • Minimum, maximum, and last values returned Additional information, such as detail message associated with data stream, is stored in related tables linked to this table through the DataID field. Related tables include: • Data • DataHeaderDeleted • DataRejected • ArchiveDataHeader • ArchiveData
Data	Individual data point values, timestamp, short and long detail messages associated with data.
ArchiveDataHeader	Data stream identification and related properties for archived data streams.
ArchiveData	Individual data point values, timestamp, short message and detail message for archived data.
ArcAvgDailyDataHeader	Data stream identification and related properties for archived data streams, aggregated and averaged daily.
ArcAvgDailyData	Individual data point values, timestamp, short and detail message for archived data, aggregated and averaged daily.
ArcAvgHourlyDataHeader	Data stream identification and related properties for archived data streams, aggregated and averaged hourly.
ArcAvgHourlyData	Individual data point values, timestamp, short and detail message for archived data, aggregated and averaged hourly.
ArcAvgMonthlyDataHeader	Data stream identification and related properties for archived data streams, aggregated and averaged monthly.

Table Name	Information stored
ArcAvgMonthlyData	Individual data point values, timestamp, short and detail message for archived data, aggregated and averaged monthly.
Job	Basic job identification, including: • Name of the Knowledge Script and the numeric identifier for the parent and child job • Numeric identifier for the computer running the job • Job status indicator • Time the job was submitted, last run, last paused, and last restarted Additional job information, such as the audit trail associated with the job, is stored in related tables linked to this table through the JobID field. Related tables include: • JobHistory • JobObject

Stored Procedures

The AppManager repository includes numerous stored procedures that perform operations when you use the Operator Console or Control Center console to complete tasks, or when the management server updates the database. These stored procedures contain the logic that serves as the backbone for creating jobs, handling events, managing data, and producing reports.

The only stored procedures you can view in their entirety are those used to generate reports. The stored procedures for reports use a special naming convention (`NetIQrp*` and `rp*`). The stored procedures are not encrypted and you can use them as templates to create new and custom reports.

SQL Server Jobs

The AppManager repository includes predefined SQL Server jobs that run at regular intervals to perform specific data management tasks. Depending on the task performed, changing the SQL Server job interval can affect database performance or efficiency. AppManager uses the following SQL Server jobs for managing data.

Note
Because these SQL Server jobs handle internal data management, you should not change the default schedule or attempt to modify the SQL Server job steps unless instructed to do so by a NetIQ Technical Support representative or implementation consultant, or unless you fully understand the impact of making a change.

Job	Description
`NetIQ AMDC daily QDB`	Removes data points from jobs that collect diagnostic data when that data is more than 1 day old. The job runs every day at 1:30 AM.
`NetIQ Archive Event QDB`	Moves events from the `Event` tables to the `ArchiveEvent` tables if you have set the repository preference to archive events. The job runs every 2 hours.
`NetIQ CC Daily Task`	Removes items marked for deletion from the `Event`, `Job`, and `Computer tables`. The job runs daily at midnight.
`NetIQ CC Half-Hourly Task`	Maintains the `Queue` table, which is used by the command queue service for tasks submitted by the Control Center console. The job runs every 30 minutes.

Job	Description
NetIQ CC Hourly Task	Maintains several background tables, such as the ArchiveQueue and Object tables, and processes custom properties and security configuration information. The job runs every 1 hour.
NetIQ CC Manage SQL Jobs	This task provides for backward compatibility only. The NQSYNCDB process now handles actions once taken by this task, and maintains SQL Server jobs for each data source or registered QDB. The job runs every minute.
NetIQ CC SMV Hourly Task	Deletes information that has been marked for removal from various tables related to Service Map Views. The job runs every 1 hour.
NetIQ Chart Console Backup QDB	Takes a backup of the Blob table in the QDB to preserve chart data that you use in the Chart Console.
NetIQ Chart Console Restore QDB	Restores the backed up Blob table data in the QDB.
NetIQ Daily QDB	Performs internal data maintenance operations, such as removing jobs, events, actions, objects and data that you mark for deletion during the day and running a consistency check for frequently used tables. The job runs every day at 1:00 AM.
NetIQ Dynamic View QDB	Ensures that relevant information is shown in any existing dynamic view. Also ensures that any job affected by parameter overrides restarts with the override values if the corresponding custom property value has changed. The job runs every 1 minute.
NetIQ Hourly QDB	Performs internal data maintenance operations, such as updating table statistics for frequently accessed tables and recompiling frequently used procedures. The job runs every 2 hours.
NetIQ License Audit QDB	Reviews the installed license keys against the information in the AppManager TreeView pane. The job runs every 3 months on the first day of the month.
NetIQ Minutely QDB	Performs internal data maintenance operations, such as attempting to restart jobs that encountered errors and attempting to transfer rejected data to the Data table. The job runs every 5 minutes.
NetIQ CC Minutely Task	Updates event severity in the Control Center repository, and ensures that the severity settings are correct for each QDB. This task refers to the GlobalPref table in the QDB to correct the severity settings. The job runs every 5 minutes.
NetIQ Monitoring Policy QDB	Applies monitoring policies defined in Control Center and in the Operator Console to matching servers. This task creates new monitoring policy jobs where it detects they are missing and removes existing jobs where it detects they should no longer be running. The job runs every 1 minute.
NetIQ Policy for CCMP QDB	Ensures that any monitoring policy defined in Control Center is applied within the QDB. The job runs every 1 minute.
NetIQ MS Healthcheck	Raises an event if it detects a downed management server. This stored procedure also closes any events when the management server is restored.
NetIQ Purge Archived Event QDB	Deletes archived event information from the ArchiveEvent and ArchiveEventDetail tables based on the repository preference you have specified. For more information about setting the repository preference used by this job, see “Managing Events” on page 85. The job runs every day at 2:00 AM.

Job	Description
NetIQ PurgeData QDB	Deletes data points from the DATA tables used for real-time charts and graphs. By default, information is stored in the DATA tables for 8 days. This job runs every 6 hours to remove data that is more than 8 days old to, keeping data streams efficient and manageable. You can change the schedule for this job by setting a repository preference. For more information about setting a repository preference, see the chapter ""Managing Data" on page 99" and "Managing Data Streams" on page 112.
NetIQ Rule Based Dynamic View QDB	Ensures that the rule-based dynamic views correctly represent the servers. The job runs every 1 minute.
NetIQ CC Upgrade QDB 6.0 To 7.0 through Linked Server in CCDB Task	Upgrades the QDB from version 6.0 to version 7.0 through the Linked Server in the Control Center repository. It upgrades the QDB so that you can use it in Control Center version 7.0 and above. This job is only present in environments from version 6.0 that have been upgraded to later versions. New version 8.0 environments do not have this job.
NetIQ Update MG Server-Membership QDB	Updates the Control Center tables in the QDB, which enables the information to be moved into the Control Center repository for display in the Control Center console. The job runs every 1 minute.
NetIQ Uphold Parameter Overrides QDB	Reconciles the overrides for monitoring policy and ad hoc Knowledge Script Group parameters with the information configured in Control Center: it compares job override settings with the settings put in the ParameterOverride table by the Control Center repository. The job runs every 1 minute.
NetIQ VSG Modtime Update (Runs Continuously) QDB	Updates the modification time in tables related to servers and views. Control Center uses this information to determine whether to update the Control Center repository. The job runs every 1 minute.
NetIQ Weekly QDB	Performs internal data maintenance operations, such as re-indexing tables and performing data archiving operations. The job runs every Sunday at 3:00 AM.

These jobs require the SQL Server Agent service to be running. You can check the status of the jobs by viewing SQL Server Agent jobs or SQL Server activity in the SQL Server Management Studio. If a job did not complete successfully at its last iteration, check the status of the SQL Server Agent. In addition, you should periodically back up SQL Server jobs as part of regular database maintenance. For more information about backing up the repository, see "Backing Up the QDB" on page 120.

Managing Data Streams

In the QDB, *current* data point information is stored in the **DataHeader** and **Data** tables. These two tables provide the information for real-time charts and graphs.

- The **DataHeader** table stores identifying information about the data stream, such as the job ID, computer name, script name, maximum days, current points, and the data legend. Each data stream has a unique identifier, or primary key, which enables the information to be referenced by the **Data** table.
- The **Data** table stores the individual data point values collected and the time each data point was collected.

In addition to these two tables, collected data is simultaneously stored in the **ArchiveData** and **ArchiveDataHeader** tables. The **ArchiveData** and **ArchiveDataHeader** tables provide the information used in reports.

The amount of data AppManager collects at each interval can quickly grow unmanageable and affect database and Operator Console performance. By default, the `NetIQ PurgeData QDB` job runs every six hours to remove information from the `DataHeader` and `Data` tables. This interval is controlled through the repository preference **Time interval to purge old points in repository**. For more information about time intervals, see "Setting Preferences for Real-time Charts and Graphs" on page 101.

When the SQL Server job runs, it checks the `DataHeader` and `Data` tables for information that is older than the time period you specified for the **Default period to keep points in data table** repository preference (8 days, by default) and removes those records. Although the information is no longer available for real-time charts and graphs, it continues to be stored in the `ArchiveDataHeader` and `ArchiveData` tables and is available for generating reports.

Because information continues to be stored in the `ArchiveDataHeader` and `ArchiveData` tables, by default, the `ArchiveDataHeader` and `ArchiveData` tables increase in size continuously and can threaten database performance and consistency. To prevent this, you can set the **Move archived points to aggregate table** repository preference to average and aggregate the collected data and move the information to the `ArcAvg*` tables. For more information about data average, see "Summarizing the Data Used for Reports" on page 102.

As with other database applications, however, you should periodically remove information that you no longer need. Regardless of whether you choose to move data to the aggregate tables, you must perform this type of database maintenance. The database size should not expand indefinitely, or database corruption could occur, making your environment unstable or unusable.

Instead, establish a regular backup strategy for saving historical information, or follow a policy of manually removing data after a certain period of time. For more information about removing data, see "Removing Archived Data and Events" on page 124.

Note

If a data stream is deleted from the Graph Data tab in the Operator Console, the record is deleted from the DataHeader table. You can set the Remove associated data when jobs are deleted repository preference to delete data from both the Data and DataHeader tables when jobs are deleted from the Operator Console, no matter how long the information has remained in the tables. For more information about deleting data, see the chapter "Managing Data" on page 99.

Managing Event Information

The QDB stores *current* event information in the `Event` and `EventDetail` tables.

- The `Event` table stores basic event information that corresponds to the columns available on the **Events** tab in List pane of the Operator Console.
- The `EventDetail` table stores the details about events that correspond to the information found in the Event Properties dialog box.

By default, event information remains stored in these tables until you delete an event in the Operator Console. Although the event tables do not typically increase in size or affect database performance as quickly as the `Data` tables, there are several ways you can manage event information in the repository to ensure database efficiency.

To keep the `Event` and `EventDetail` tables from growing too large, you should enable the repository preference **Move aged events to archive tables in repository**. This preference also lets you specify conditions for moving event information from the `Event` and `EventDetail` tables to the `ArchiveEvent` and `ArchiveEventDetail` tables. For example, you can choose to archive events that remain closed for a certain number of days or that have a certain severity. This option also allows you to specify conditions for removing archived events from the database.

Another good practice is to set the **Remove associated events when jobs are deleted** repository preference to delete data from both the `Event` and `EventDetail` tables when jobs are deleted from the Operator Console. This preference acts independently of your event-archiving settings. For more information about event settings, see "Managing Events" on page 85.

As with any database application, however, you should periodically remove information that you no longer need. Regardless of whether you choose to move events to the archive tables, you should plan for this type of database maintenance. For more information about archiving events, see "Removing Archived Data and Events" on page 124.

Managing Audit Trails

For security reasons, AppManager automatically collects audit information about when computers are placed in maintenance mode and about user logon activity. You can also configure AppManager to collect an audit trail for jobs, events, or actions using the Miscellaneous repository preference.

If you enable auditing for jobs, events, or actions, AppManager records information about every related operation in the repository. For example, if you enable auditing for job-related operations, the audit trail includes chronological information about who started each job, changes to job properties, and changes to job status.

Because the QDB stores audit information and can increase in size continuously, you should periodically remove the information that you no longer need from the tables that store the audit information you collect:

Repository Table	Type of Information Stored
ActionHistory	Action-related activities
Event History	Event-related activities
Job History	Job-related activities
MachineMntHistory	Maintenance mode activity
QLogonoffHistory	Logon and logoff activity

Whether you choose to enable auditing for jobs, events, or actions, you should plan for periodic maintenance of the `MachineMntHistory` and `QLogonoffHistory` tables. For example, you should plan a regular backup strategy for saving historical information and establish a policy for manually removing audit information after a certain period of time.

Note
Do not remove information from the KSHistory table. The QDB manages the contents of this table internally.

Checking SQL Server Configuration

SQL Server includes many configuration options for tuning server and database operations. For most of these options, SQL Server dynamically adjusts the configuration when needed, for example by allocating more memory or adding user connections.

In most environments, you do not need to manually set or adjust any of these options. In rare cases, however, you may find it useful to override SQL Server's dynamic allocation and manually set the value of a configuration option. The specific settings you should use can vary greatly, depending on your deployment and your database management practices.

However, the SQL Server configuration options you are most likely to use for AppManager are:

SQL Server Option	Description
`locks`	The maximum number of available locks. By default, SQL Server can allocate and deallocate locks dynamically, based on changing system requirements. The dynamic lock manager can allocate available memory for lock structures up to a maximum of 40% of the total memory allocated to SQL Server. You can use the `locks` advanced option to override SQL Server default lock allocation. The internal memory cost of locks is relatively low, so you may want to manually set this option to avoid running out of locks on busy systems. In general, you can calculate the value for locks by estimating (user connections) x (maximum number of tables simultaneously accessed by any user at the same time) x (maximum rows per table accessed by any user at the same time). This estimation assumes no lock escalation and more page activity than is likely to occur, but is a good starting point.
`min server memory` `max server memory`	The bottom and top thresholds of available RAM for SQL Server. By default, SQL Server automatically adjusts memory based on need and availability, but you can use these options to control the range of memory configured automatically. Maximizing I/O speed is a critical factor in SQL Server performance. Increasing SQL Server's memory allocation reduces I/O requirements and increases data caching. In general, you should configure SQL Server with as much RAM as you can without causing Windows to page. But you should also consider the size of your SQL Server environment and how your AppManager components are distributed. For example, if you are using a single server for both the repository and management server, you should configure SQL Server to use less memory--to accommodate Windows and the management server--than if the server is a dedicated repository server.

SQL Server Option	Description
`open objects`	The maximum number of database objects that can be open at one time on a SQL Server instance. By default, SQL Server sets this value depending on the current needs of the system. You can use the `open objects` advanced option to increase the maximum number of open objects. To set this value, estimate the maximum number of tables, views, rules, stored procedures, defaults, and triggers that will be open at any one time for the SQL Server. It is better to overestimate than underestimate this value. A typical configuration in an active environment is 5000 open objects. If you are collecting or purging a large amount of data, you should increase the number of open objects, for example, to 20,000, or even 50,000. **Note** Open objects consume memory, so increasing this value reduces the memory available for other SQL Server operations. It may require a larger amount of memory dedicated to the server.
`user connections`	The maximum number of simultaneous SQL Server connections allowed. By default, SQL Server can dynamically adjust the maximum number of user connections as needed, up to the maximum value allowable. You can use the `user connections` advanced option to manually set the maximum number of simultaneous connections to SQL Server. To determine the maximum number of user connections that your system allows, use the following statement: `SELECT @@MAX_CONNECTIONS` Typically, AppManager requires 80-100 user connections. The NetIQ Corporation AppManager Management Service (`NetIQms`) uses 40-60 connections. Each console simultaneously connecting to the repository uses 1-5 connections. If your organization has several console programs connecting to the same repository at the same time, you may need to increase this value. For example, if, on average, you have 12-15 console programs simultaneously accessing the same repository, you might set the user connections option to a value of 150 to 200 (depending on the amount of buffer you want for connections that are blocked and must time out). **Note** Each connection takes approximately 40 KB of overhead, regardless of whether the connection is being used.

Note
You should check your Microsoft SQL Server documentation and your current server configuration to determine which options, if any, should be changed.

In addition to these server-level configuration options, SQL Server includes several database-level configuration options that you can use to specify the characteristics of each database. These options help you to control specific behavior for a database, such as automatic operations and recovery options. In general, these options do not affect AppManager operation or performance. You should, however, consider these options when planning your backup and database management strategy and when deciding whether you will do full backups, incremental backups, or both.

Expanding the Size of a Repository

When you install the AppManager repository, you set the initial size of the repository's data and log files. After installation, you can use the Management Studio to expand the size of the data and log files or add new data and log files for the repository to expand into over time. In planning for repository growth, consider the following:

- SQL Server is configured by default to automatically grow the database file size without restriction. NetIQ Corporation recommends that you restrict database file growth so that the database cannot consume all of the disk's resources. If the database consumes all available disk space, you cannot perform any maintenance on it, including deleting or truncating data.
- If you allow SQL Server to grow automatically, you will experience some performance degradation while SQL Server is in the process of expanding the database.
- You can reduce disk fragmentation by configuring the database file size to expand in larger increments.
- Increasing the size allotted for the data file or adding data files before a significant amount of information is stored in the repository helps to maximize performance.

To expand the size of a QDB, first increase the size of the database data file.

To increase the size of the data file:

1. In SQL Server Management Studio, expand the server that the database resides on, then expand the **Databases** folder.
2. Right-click the QDB database and select **Properties**.
3. Click the **Files** node.
   - To change the size of an existing database file, click the **Initial Size** column for the data file that you want to expand and specify a new database file size.
   - To add a new database file, click **Add**. In the new row type a name in the **Logical Name** column and specify the file size.
4. Click the button in the **Autogrowth** column and configure the following database autogrowth parameters:
   - Select **Enable Autogrowth** to enable the database size to grow automatically.
   - Under **File Growth**, select **In Percent** and specify the percentage by which you want the database size to grow, or select **In Megabytes** and specify the number of megabytes by which you want the database size to grow.
   - Under **Maximum File Size**, select **Restrict file growth (MB)** and set a maximum file size for the data file if you allow data files to grow automatically.

   Restricting growth ensures that the database will not consume all available disk space if left un.

You can also expand the size of the existing log file or add new log files for the repository. Click the **Transaction Log Shipping** node in the Properties dialog box and set a new log file size or add a new log file.

Checking the Integrity of the Repository

Periodically, you should check repository integrity to prevent problems such as uncontrolled database growth, corruption of database tables, or inconsistencies in database structure.

As with other database maintenance tasks, the frequency of database consistency checking depends on several factors, including the number of jobs you run, the number of events you generate, the number of data points you collect, and the stability of network communication. In general, the larger the repository, the more frequently you should check database consistency.

Several SQL Server `dbcc` commands can be used to check table consistency, segment usage, page allocation, pointer operations, and index operations. You can also use the AMAdmin_DBHealth Knowledge Script to perform a more limited check of the `syslog` system table and the main tables that store AppManager activity, such as the tables for events, data, and jobs.

Note
The AMAdmin_DBHealth Knowledge Script also checks the percentage of data space and log space used by the Update Definition in User Variables. repository, the time it takes an Update Definition in User Variables. query to execute, and the status of Update Definition in User Variables. scheduled SQL Server jobs. For more information about using this Knowledge Script, select the Knowledge Script help.

The following `dbcc` commands can be executed in any SQL Server query tool, such as SQL Server Management Studio or `isql`. For more information about the syntax to use with these commands, see the *Microsoft Transact-SQL Reference Guide*.

Command	Description
`DBCC CHECKCATALOG`	Checks the system tables for consistency and display segment information. This command is highly effective and typically takes less time to complete than other consistency-checking commands.
`DBCC NEWALLOC`	Ensures that page allocation is correct and that the page structure for the data and index pages is consistent. **Note** Run this command in single-user mode.
`DBCC TEXTALL`	Checks text and image allocation errors for all tables that contain text and image columns.
`DBCC CHECKTABLE`	Ensures that all pointers and data and index pages in a specified table are consistent and properly linked.
`DBCC CHECKDB`	Ensures that all pointers and data and index pages for all tables in the database are consistent and properly linked.

Because most database consistency checks can take several hours to run, you should plan to run these processes during off-peak hours. You should also include these checks in your overall database maintenance plan and backup strategy to ensure you always back up a clean database. For example, if you are nearing maximum capacity on a server, you should check the database consistency in preparation for backing up.

The AM Self Monitoring module, also known as AM Health, includes the AMHealth_QDBComponentsHealth Knowledge Script for monitoring the health and availability of SQL Server resources for the QDB. For more information, see "AM Health Knowledge Scripts" on page 241.

Backing Up the QDB

In planning your database maintenance strategy, evaluate the following:

- Your data collection policies
- Your requirements for producing charts and reports
- How you address event activity

You can then use this information to help you establish a schedule for performing regular backups of the AppManager repository.

You also need to back up the QDB if you move the database server to another computer, or if you upgrade from a 32-bit SQL Server installation to a 64-bit SQL Server installation. If you are moving the 32-bit QDB to a 64-bit SQL Server computer, make sure you have the latest 32-bit hotfixes and modules. After you move the QDB to the 64-bit SQL Server computer, you cannot install the 32-bit hotfixes and modules.

Notes

- The `master` database holds all of the device configuration information, SQL Server login information, and extended stored procedures. Therefore, you should periodically back up the `master` database even if you are not backing up the AppManager repository—especially if you add data devices, databases, or users, or change any configuration options.
- If you want to move the 32-bit QDB to a 64-bit SQL Server, you need to update you current AppManager installation to AppManager version 7.0.3 and migrate your data. For more information about updating to AppManager version 7.0.3, contact NetIQ Technical Support.

You can achieve the following results if you take regular scheduled backups of the QDB:

- Ensure the integrity of the data stored in the QDB
- Provide a means for archiving historical information
- Prevent data loss
- Facilitate disaster recovery
- Enable the movement of the QDB from a 32-bit SQL Server computer to another 32-bit or 64-bit SQL Server computer

Although you can back up the repository manually at any time using SQL Server Management Studio or with the SQL_RunSQL Knowledge Script, both the SQL Server Management Studio and the RunSQL script allow you to automate backups by scheduling them to run at set intervals or at specific days and times.

Note

You must install the SQL module to take a back up with the SQL_RunSQL Knowledge Script.

The frequency of your full or incremental backups depends on the nature of your environment, including the size of the data and log files, the number of computers you monitor, the number of Knowledge Scripts you run, how much data you collect, how you use charts and reports, and how quickly you acknowledge and close events.

As a general rule, you should perform a complete or incremental backup at least once a week.

Disconnecting Connections to the QDB

Before you back up the repository database, disconnect any connections to the repository. If you created SQL users from Security Manager, you need to note down the roles that you assigned to each of the SQL users. You will need to re-create the SQL users after you restore the QDB on the new SQL Server computer.

To disconnect the connections to the repository:

1. Stop the following services and agents:
   - SQL Server Agent service on the QDB server.
   - NetIQ AppManager Management Services on management servers that connect to the QDB.
   - NetIQ AppManager Performance Profiler 4.0.2 (or 4.1.2) Analytics service. You must stop this service at the same time as you are stopping the management servers.
   - NetIQ AppManager Client Resource Monitor service on the management servers that connect to the QDB.
   - NetIQ AppManager Client Communication Manager service on the management servers that connect to the QDB.
   - NetIQ AppManager Control Center Command Queue service on the Control Center computer if Control Center manages the repository.
   - SQL Server Agent service on the Control Center repository server if Control Center manages the repository.
   - Report agents that connect to the QDB.
   - You must also stop AppManager Connectors like the AppManager Connector for Micromuse Netcool/OMNIbus, or the AppManager Connector for Security Manager that directly connect to the QDB.

   Notes
   - If you set the service to automatically restart when it stops, you must disable the service.
   - If you want to move the QDB to a different computer, you should start all services only after you complete all the steps for moving the QDB.
2. Close any AppManager console applications, such as the Operator Console or Control Center console, that are connected to the QDB.
3. If the QDB is a data source for Analysis Center, stop Analysis Center ETL jobs.
4. Verify that there are no open connections to the QDB by running the following SQL query:

```
USE master
GO
Exec sp_who2
GO
```

5. If no programs are listed in the query results, you are now ready to back up the QDB.

Backing up the Repository with SQL Server Management Studio

After you complete the steps given above you can take a backup of the repository.

To back up the QDB:

1. Start SQL Server Management Studio and expand the computer where the QDB is located.
2. Expand the **Databases** folder and select the QDB.
3. Right-click and select **Tasks > Backup**.
4. The Back Up Database dialog box displays.

   Specify the following details:

Field	Description
Database	Select the QDB that you want to back up from the list.
Recovery Model	Displays the type of recovery model that the database follows. This field is disabled by default. For more information about recovery models, see the SQL Server documentation.
Backup Type	Select **Full** as the backup type to create a full backup copy of the repository.
Backup Component	Select Database as the backup component.
Backup set will expire	Select **After** and retain the default setting of zero such that your backup does not expire. If you select the **On** option, you need to specify a date on which the backup expires.
Destination	Select **Disk** to copy the backup to a particular folder. Select a backup destination, if the destination you want to use is listed. To select a destination or backup device that is not listed, click **Add**.

5. Click **Options** under the Select a Page pane.
6. Under Overwrite Media, select **Back up to the existing media set**.
7. To add this backup:
   - To existing backups, select **Append to the existing backup set**.
   - To discard existing backups, select **Overwrite all existing backup sets.**
8. Click **OK**.

   After the SQL Server Management Studio completes the backup operation, it displays the following message:

   `"The backup of database <QDB name> is completed successfully."`

9. Click **OK**.
10. Run the following SQL statement to add a backup device for the transaction log and to back up the transaction log:

```
USE master
EXEC sp_addumpdevice 'disk', 'dump_device_log', 'C:\QDBBACKUP\QDB_Log.bak'
GO
BACKUP LOG QDB TO dump_device_log
GO
sp_dropdevice 'dump_device_log'
GO
```

where *QDB* is the name of the QDB and *dump_device*_log is the name of the backup device or file.

Restoring the Repository with SQL Server Management Studio

This section explains how to restore a QDB from a database backup. This process assumes you are moving to the same version of AppManager using the same version of SQL Server on the same platform (32-bit or 64-bit). If you are attempting to restore a QDB as a way to migrate from a 32-bit to a 64-bit version of SQL Server or to upgrade to a newer version of SQL Server or AppManager, see the *Upgrade and Migration Guide for AppManager*.

To restore a QDB:

1. Start SQL Server Management Studio and expand the computer where the QDB is located.
2. Click the **Databases** folder and select the QDB.
3. Right click and select **Tasks > Restore > Database**. The Restore Database dialog box displays.
4. Under **Destination for restore**, select the QDB from the list.
5. Select the earliest backup from which you want to restore information. If you have multiple backup files, you can choose the files to use. The default is the most recent possible backup file.
6. Under **Source for restore**, select the **From device** option and click the Lookup (...) button.
7. In the Specify Backup dialog box, specify the following:
   - **Backup media -** Select **File** from the list since you have saved your database backup as a file.
   - **Backup location -** Click **Add** to browse for the location where you have saved the database backup.
8. Click **OK**.
9. Select the **Restore** option under Select the backup sets to restore, to restore the backup set you have saved.
10. Click **Options** in the Select a page pane.
11. Select **Overwrite the existing database** under **Restore Options** and retain the default option under **Recovery State.**
12. When you finish the restore, restart the **SQL Server Agent** services you stopped previously.

13. Run the following SQL statements in Query Analyzer from QDB database (which was recently restored):

```
EXEC task_util 1,1
EXEC task_utilExt 1,1
```

14. ***If you restored the repository to a different computer,*** update the management server configuration. For more information about updating the management server configuration, see "Moving the QDB" on page 126.

Note
If you restore the QDB to a different computer, and it is managed by Control Center, you need to update the QDB connection information in the Control Center repository. For more information, see the *Upgrade and Migration Guide for AppManager*.

Removing Archived Data and Events

In the AppManager Operator Console, you can set repository preferences for archiving data and events. These options allow you to move historical data information to archive tables where it is still available for reporting. You can also choose to periodically move archived data to aggregate tables to further reduce database space requirements.

Over time, these tables increase in size and can eventually threaten database performance and consistency. As with any database, therefore, you should periodically remove archived information that it is no longer needed to prevent the database from growing continuously.

Using DeleteOldArchiveData to Remove Data

To simplify the periodic maintenance of the ArchiveData tables, AppManager includes the `DeleteOldArchiveData` stored procedure. This stored procedure lets you specify the number of days of data to keep in the repository. For example, to delete all of the data from the `ArchiveData` table that is more than 100 days old, run the following SQL command:

```
DeleteOldArchiveData 100
```

Note
This stored procedure can impose a heavy load on your SQL Server. Run it only during off-peak hours and when the NetIQ PurgeData QDB job is not running. If you have accumulated a large amount of data (for example, two million rows or more) before running this stored procedure for the first time, gradually decrease the number of days so that you incrementally delete the oldest data first. For example, if you have 120 days' worth of data, run DeleteOldArchiveData 115, then DeleteOldArchiveData 110, then DeleteOldArchiveData 105, until the desired amount of data has been deleted.

Using Standard SQL Statements to Remove Data

In addition to the `DeleteOldArchiveData` stored procedures, you can use standard Microsoft SQL Server tools, such as Query Analyzer and Management Studio, to periodically view, export, or remove archived data and events and perform other administrative tasks. With these tools, you can remove archived data based on specific criteria such as the `DataID` or a date by executing standard SQL statements. For example, to remove all records for the data stream with `DataID` 2, use a SQL statement like this:

```
DELETE dbo.ArchiveData WHERE DataID = 2
```

Typically, the `DELETE` and `TRUNCATE TABLE` statements are used to remove some or all rows from the archive or aggregate tables. You should, however, use caution in performing database operations with these statements.

Note
Removing data and events can impose a heavy load on your SQL Server. Only perform these operations during off-peak hours when the NetIQ PurgeData QDB job is not running.

Using the DELETE Statement

The `DELETE` statement removes rows one at a time and logs each deletion, based on the conditions specified in the `[WHERE]` clause. `DELETE` permission defaults to the table owner, who can transfer it to others.

The syntax of the `DELETE` statement is:

```
DELETE [FROM] {table-name | view_name} [WHERE clause]
```

For example, to remove all records collected on a managed computer before a specified time, enter a SQL statement similar to:

```
DELETE dbo.ArchiveData WHERE Time < time_in_UTC_format
```

Using the TRUNCATE TABLE Statement

The `TRUNCATE TABLE` statement removes all rows by logging only the page deallocations. Using this statement is faster than using the `DELETE` statement. `TRUNCATE TABLE` permission defaults to the table owner and is not transferable.

The syntax of the `TRUNCATE TABLE` statement is:

```
TRUNCATE TABLE [[database.]owner.]table_name
```

For example, to remove all the data from the `ArchiveData` table, use a SQL statement like this:

```
TRUNCATE TABLE dbo.ArchiveData
```

Removing Events and Data for Deleted Jobs

If you do not automatically remove events and data when jobs are deleted, periodically remove this information from the repository manually. To remove this obsolete information for jobs that have been deleted, you can use the `CleanOldEventAndData` stored procedure. This stored procedure enables you to specify whether you want to delete events, data, or both. You can also use this procedure to remove information for a specific data stream by DataID or a specific event by EventID. The syntax for this stored procedure is:

```
CleanOldEventAndData Type, [DataID, EventID]
```

where *Type* can be `0` to remove both data and events, `1` to remove only data, or `2` to remove only events. The `DataID` and `EventID` are optional parameters. Use them to delete information for a specific `DataID` or `EventID`.

For example, to remove all events and data for all deleted jobs, you would run the following SQL command:

```
CleanOldEventAndData 0
```

To remove all events for all deleted jobs without removing any data, you would run the following SQL command:

```
CleanOldEventAndData 2
```

To remove all orphan events but only the data associated with the `DataID 123`, run the following SQL command:

```
CleanOldEventAndData 0, 123
```

Moving the QDB

This section provides a brief overview of how to move the QDB from one SQL Server computer to another SQL Server computer.

If you use the repository only as a read-only archive of information, you can typically use SQL Server backup and restore capabilities to copy the repository to a new SQL Server computer. You cannot, however, use the basic backup and restore capabilities to move an active AppManager repository that receives connections from the management server and the Control Center console or the Operator Console.

You can move the QDB between two 32-bit or 64-bit SQL Server computers or from a 32-bit SQL Server computer to a 64-bit SQL Server computer.

Notes

- If you move the QDB from a 32-bit SQL Server to a 64-bit SQL Server computer, you must also move the Control Center repository to a 64-bit SQL server computer at the same time.
- You must disconnect both the 32-bit repositories when you move them to 64-bit. You can choose to move any secondary QDBs to 64-bit.

For detailed instructions about moving a QDB, see the *Upgrade and Migration Guide for AppManager*.

Using the Repository Browser

The Repository Browser allows you to browse the records in the QDB and use standard SQL commands to write queries to retrieve the information from the database. As some of the information stored in the repository may be sensitive, you should consider restricting access to the Repository Browser through the AppManager Security Manager.

Once you are connected to the repository, you can use the Repository Browser to:

- Select a table from the **Tables** list to view all records in a table
- Select predefined queries from the **Custom query** list to view records meeting a specific criteria
- Create and save custom queries
- Export records to text files

For more information about using the Repository Browser, see the AppManager Help.

Chapter 8
Managing Control Center

The Control Center repository is stored in a Microsoft SQL Server database. As with any database, you need to perform a number of procedures regularly to maintain the database and ensure database integrity. This chapter provides an overview of the information stored in the Control Center repository and describes how to perform routine database maintenance and ensure database consistency. For more detailed information about how to perform any of these tasks using Microsoft SQL Server Management Studio or other SQL Server tools, see your Microsoft SQL Server documentation.

Understanding the Control Center Repository

The Control Center repository is a core component of the AppManager architecture. It is a SQL Server database that stores all Control Center information, including details about all of the jobs you are running, the events being collected, the managed computers being monitored, and the options you've set for viewing and managing jobs and events. The Control Center repository also includes stored procedures that Control Center uses in the background to perform the database operations you request as well as several scheduled jobs to perform certain maintenance tasks at set intervals.

The AM Self Monitoring module, also known as AM Health, includes the AMHealth_QDBCComponentHealth Knowledge Script for monitoring the health and availability of SQL Server resources for the QDB. For more information, see "AM Health Knowledge Scripts" on page 241.

Backing Up the Control Center Repository

In planning your database maintenance strategy, you should evaluate how you address event activity. You can then use this information to help you establish a schedule for performing regular backups of the Control Center repository.

You can achieve the following results if you take regular scheduled backups of the Control Center repository:

- Ensure the integrity of the data stored in the Control Center repository
- Provide a means for archiving historical information
- Prevent data loss
- Facilitate disaster recovery

Although you can back up the repository manually at any time using SQL Server Management Studio or with the SQL_RunSQL Knowledge Script, both Management Studio and the RunSQL script allow you to automate backups by scheduling them to run at set intervals or at specific days and times.

The frequency of your backups depends on the nature of your environment, including the size of the data and log files, the number of computers you monitor, the number of Knowledge Scripts you run, and how quickly you acknowledge and close events.

As a general rule, you should perform a complete or incremental backup at least once a week.

Disconnecting Connections to the Control Center Repository

Before you back up the Control Center repository database, disconnect any connections to the repository.

To disconnect the connections to the repository:

1. Stop the NetIQ AppManager Control Center Command Queue Service.
2. Stop the SQL Server Agent service on the Control Center repository server.
3. Stop the NetIQ AppManager Deployment Services that connect to the Control Center repository.
4. Stop the World Wide Web Publishing Service that manages the ProxyDeploymentWebService and the WebDepot virtual directories.

   Note
   If you want to move the Control Center repository to a different computer, you should start all services only after you complete all the steps for moving the Control Center repository.

5. Close any open Control Center console applications.
6. Verify that there are no open connections to the Control Center repository (**NQCCDB**) by running the following SQL query:

   ```
   USE master
   GO
   Exec sp_who2
   GO
   ```

7. If no programs are listed in the query results, you are now ready to back up the Control Center repository.

Backing up the Control Center Repository with SQL Server Management Studio

To back up the Control Center repository:

1. Start SQL Server Management Studio and expand the computer where the Control Center repository is located.
2. Expand the **Databases** folder and select the Control Center repository (NQCCDB).
3. Right-click and select **Tasks > Backup**.

4. The Back Up Database dialog box displays.

   Specify the following details:

Field	Description
Database	Select the Control Center repository from the list.
Recovery Model	Displays the type of recovery model that the database follows. This field is disabled by default. For more information about recovery models, see the SQL Server documentation.
Backup Type	Select **Full** as the backup type to create a full backup copy of the repository.
Backup Component	Select **Database** as the backup component.
Backup Set	Specify a name and description for your database backup set.
Backup set will expire	Select **After** option and retain the default setting of zero such that your backup does not expire. If you select the **On** option, you need to specify a date on which the backup expires.
Destination	Select **Disk** to copy the backup to a particular folder. Select a backup destination, if the destination you want to use is listed. To select a destination or backup device that is not listed, click **Add**.

5. Click **Options** under the Select a Page pane.
6. Under **Overwrite Media**, select **Back up to the existing media set**.
7. To add this backup:
   - To existing backups, select **Append to the existing backup set**.
   - To discard existing backups, select **Overwrite all existing backup sets**.
8. Click **OK**.

   After the SQL Server Management Studio completes the backup operation, it displays the following message:

   ```
   "The backup of database <Control Center repository name> is completed
   successfully."
   ```

9. Click **OK**.
10. In SQL Query Analyzer, check if the Control Center repository database recovery model is **full** or **simple** by running the following SQL statement:

    ```
    SELECT DATABASEPROPERTYEX('NQCCDB','recovery') As Recovery
    ```

11. If the QDB database recovery model is **full**, run the following SQL statement to add a backup device for the transaction log and to back up the transaction log:

```
USE master
EXEC sp_addumpdevice 'disk', 'dump_device_log', 'C:\NQCCDBBACKUP\NQCCDB_Log.bak'
GO
BACKUP LOG NQCCDB TO dump_device_log
GO
sp_dropdevice 'dump_device_log'
GO
```

where *NQCCDB* is the name of the Control Center repository and *dump_device*_log is the name of the backup device or file.

Restoring the Control Center Repository

You need to restore the backed up Control Center repository on the new SQL Server Computer.

- The Control Center repository can reside on both 32-bit and 64-bit installations of Microsoft SQL Server 2005.
- If you want to move the 32-bit Control Center repository to a 64-bit SQL Server, you need to update your current AppManager installation to AppManager version 7.0.3 and migrate your data. For more information about updating to AppManager version 7.0.3, contact NetIQ Technical Support.

If you upgrade from a 32-bit to a 64-bit version of Microsoft SQL Server 2005, complete the following steps before you restore the QDB on a 64-bit Microsoft SQL Server installation.

Installing the Control Center Repository on a 64-bit SQL Server

For information about installing the Control Center repository on a 64-bit SQL Server, see the *Installation Guide for AppManager*.

Restoring the Control Center Repository

This section explains how to restore a Control Center repository from a database backup. This process assumes you are moving to the same version of AppManager using the same version of SQL Server on the same platform (32-bit or 64-bit). If you are attempting to restore a Control Center repository as a way to migrate from a 32-bit to a 64-bit version of SQL Server or to upgrade to a newer version of SQL Server or AppManager, see the *Upgrade and Migration Guide for AppManager*.

To restore a Control Center repository:

1. Stop the SQL Server Agent service.
2. Start SQL Server Management Studio and expand the computer where the Control Center repository is located.
3. Click the **Databases** folder and select the Control Center repository.
4. Right click and select **Tasks > Restore > Database**. The Restore Database dialog box displays.
5. Under **Destination for restore**, select the Control Center repository from the list.
6. Select the earliest backup from which you want to restore information. If you have multiple backup files, you can choose the files to use. The default is the most recent possible backup file.
7. Under **Source for restore**, select the **From device** option and click the Lookup (...) button.

8. In the Specify Backup dialog box, specify the following:
   - **Backup media -** Select **File** from the list since you have saved your database backup as a file.
   - **Backup location -** Click **Add** to browse for the location where you have saved the database backup.
9. Click **OK**.
10. Select the **Restore** option under **Select the backup sets to restore**.
11. Click **Options** in the Select a page pane.
12. Select **Overwrite the existing database** under **Restore Options** and retain the default option under **Recovery State.**
13. When you finish the restore, restart the SQL Server Agent service.
14. **If you restored the Control Center repository to a different computer**, you must perform additional steps to finish moving the repository. For more information, see "Moving the Control Center Repository" on page 133.

Moving the Control Center Repository

You can move the QDB between two 32-bit or 64-bit SQL Server computers or from a 32-bit SQL Server computer to a 64-bit SQL Server computer. For detailed instructions about moving a Control Center repository, refer to the *Upgrade and Migration Guide for AppManager*.

Moving the Deployment Service to a New Computer

If you want to move a Control Center deployment service to a new computer, after you install the deployment service on the new computer and uninstall the deployment service on the old computer, the old deployment service continues to be displayed in the list of deployment services in the Deployment Rule Wizard.

To resolve this issue, in SQL Query Analyzer, run the following SQL statement on the Control Center repository database (**NQCCDB**):

```
exec dplRemoveService 'Old_DeploymentService'
```

where `Old_DeploymentService` is the name of the computer where you uninstalled the deployment service.

For information on installing and uninstalling the deployment service, see the *Installation Guide for AppManager*.

Moving the Deployment Web Service to a New Computer

You can move the Deployment Web Service to a new computer.

To move the Deployment Web Service to a new computer:

1. Install the new Deployment Web Service. For more information, see "Installing the New Deployment Web Service" on page 134
2. Update the Control Center preferences to specify the new Deployment Web Service. For more information, see "Updating the Control Center Preferences to Specify the New Deployment Web Service" on page 134.
3. Reconfigure any deployment services that use the Deployment Web Service as a proxy. For more information, see "Reconfiguring Any Deployment Services That Use the Deployment Web Service as a Proxy" on page 134.
4. Update your AppManager agents to communicate with the new Deployment Web Service. For more information, see "Updating Your AppManager Agents to Communicate with the New Deployment Web Service" on page 135.
5. Uninstall the existing Deployment Web Service. For more information, see "Uninstall the Existing Deployment Web Service" on page 135.

Installing the New Deployment Web Service

For information about installing the Deployment Web Service, see the *Installation Guide for AppManager*.

Updating the Control Center Preferences to Specify the New Deployment Web Service

In the Control Center console, on the Main tab of the ribbon, in the Tools group, click **Options > Deployment > General** and update the **Web Server** field to specify the name of the computer where the new remote Deployment Web Service is installed.

Reconfiguring Any Deployment Services That Use the Deployment Web Service as a Proxy

To reconfigure the Deployment Service:

1. Right-click the `DeploymentService.exe.config` file under *<NetIQ-install_path>*`\AppManager\Control Center\bin` and click **Properties**.
2. In the Properties dialog box, remove the Read-only setting and click **OK**.
3. **In the** `DeploymentService.exe.config` file, under <appSettings>, change the value of the ProxyWebService parameter to specify the new name of the deployment web service, for example:

   ```
   <add key = "ProxyWebService" value = "DeploymentWebServer"/>
   ```

   where *DeploymentWebServer* is the name of the computer where the new Deployment Web Service is installed.
4. Close and save your changes to the `DeploymentService.exe.config` file.

5. Restart the NetIQ AppManager Deployment Service for your changes to take effect.
6. Repeat these steps for each Deployment Service in your environment that uses the Deployment Web Service as a proxy.

Updating Your AppManager Agents to Communicate with the New Deployment Web Service

By default, if an agent does not contact the Deployment Web Service within 3 days, you cannot deploy new installation packages to the agent without first restarting the AppManager agent. Until you reconfigure your agents to communicate with the new Deployment Web Service, the Control Center console will not display updated software inventory information.

To configure your agents to use the new Deployment Web Service, run the AMAdmin_SetDeploymentWebService Knowledge Script.

Uninstall the Existing Deployment Web Service

For information about uninstalling the Deployment Web Service, see the *Installation Guide for AppManager*.

Moving the Command Queue Service to a New Computer

You can move the command queue service to a new computer.

To move the command queue service to a new computer:

1. Delete the command queue service setting from the Control Center database by running the following SQL statement:

   ```
   delete Property where Scope = 'cqs'
   ```
2. Install the command queue service on the new computer.

Changing the Log Path for the Command Queue Service

You can change the folder location where the NetIQ AppManager Control Center Command Queue Service stores its log file.

To change the folder location:

1. On the command queue service computer, open the Services Control Panel and stop the NetIQ AppManager Control Center Command Queue Service.
2. Right-click the `NQCQS.exe.config` file under *`<Netiq_install_path>`*`\AppManager\Control Center\bin` and click **Properties**.
3. In the Properties dialog box, remove the Read-only setting and click **OK**.
4. In the `NQCQS.exe.config` file, under <appSettings>, change the value of the FilePath parameter to specify the new file path for log file, for example:

   ```
   <add key = "filepath" value = "e:\NetIQ_debug\CC_CQSTrace\"/>
   ```

5. Close and save your changes to the `NQCQS.exe.config` file.
6. Open a Command Prompt and change directories to `\NetIQ\AppManager\Control Center\bin` and run the following command:

   ```
   nqcqs.exe -i
   ```
7. Restart NetIQ AppManager Control Center Command Queue Service.
8. Validate the log file exists in the specified folder location.

Changing the Log Path for the Deployment Service

You can change the folder location where NetIQ AppManager Control Center Deployment Service stores its log file.

To change the folder location:

1. On the Deployment Service computer, open the Services Control Panel and stop the NetIQ AppManager Deployment Service.
2. Right-click the `DeploymentService.exe.config` file under *<NetIQ_install_path>*`\AppManager\Control Center\bin` and click **Properties**.
3. In the Properties dialog box, remove the Read-only setting and click **OK**.
4. **In the** `DeploymentService.exe.config` file, under <appSettings>, change the value of the **FilePath** parameter to specify the new name of the deployment web service, for example:

   ```
   <add key = "filepath" value = "e:\NetIQ_debug\CC_ADTrace\"/>
   ```
5. Close and save your changes to the `DeploymentService.exe.config` file.
6. Restart the NetIQ AppManager Deployment Service for your changes to take effect.

Changing Configuration Parameters

In slow or busy network environments, you may encounter timeout exception errors in the communication between the Control Center console and the Control Center repository. You can change certain SQL connection and command/query parameters to better adapt console and repository communication to your environment.

You can edit these parameters in two standard .NET XML configuration files:

- `AMCC.exe.config`
- `NQCQS.exe.config`

Before you edit these configuration files, contact Technical Support. The `AMCC.exe.config` file controls communication between the Control Center console and the Control Center repository. The `NQCQS.exe.config` file controls communication between the Control Center repository and any QDBs you are managing with Control Center.

These configuration files are located in the same directory where the associated application or service runs. For example, the default location for both the `AMCC.exe.config` and `NQCQS.exe.config` files is `C:\Program Files\NetIQ\AppManager\Control Center\bin`.

You can change the following parameters in both the `NQCQS.exe.config` file and the `AMCC.exe.config` file:

Connection timeout (ConnectionTimeOut)

The time to wait (in seconds) before terminating an attempt to establish a connection to an instance of SQL Server and generating an error.

The default value is 15 seconds.

Command timeout (CommandTimeOut)

The time to wait (in seconds) before terminating an attempt to execute a command and generating an error.

The default value for `NQCQS.exe.config` is 600 seconds and for `AMCC.exe.config` the default value is 90 seconds.

Packet size (PacketSize)

The size (in bytes) of network packets used to communicate with an instance of SQL Server.

The default value is 8192 bytes.

Retry count (RetryCount)

The number of times to retry a connection or command operation if the operation failed.

The default value is 1.

Log File Size (FileSize)

Specifies the maximum size, in bytes, for the log file. If the log file exceeds this threshold, a new log file is created.

Number of Logs (NumBackups)

Specifies the maximum number of log files. When the maximum number is reached, the oldest log file is overwritten. The default value is 1.

Log File Path (FilePath)

The path on the command queue service computer for the log file.

TraceLevel (TraceLevel)

Specifies the level of tracing information you want in the log file. These are the available tracing levels:

- **Off** to disable logging for non-Error events.
- **Error** to log program exceptions to the Windows Event Log and the command queue service (`NQCQS.exe.config`) or `AMCC.exe.config` log file. All critical messages are always logged to the Windows Event Log. This is the default for `NQCQS.exe.config`.
- **Warning** to log program recoverable errors to the command queue service (`NQCQS.exe.config`) or `AMCC.exe.config` log file. This is the default for `AMCC.exe.config`.
- **Info** to log program warnings and flow information to the command queue service (`NQCQS.exe.config`) or `AMCC.exe.config` log file.
- **Verbose** to log program debug and trace information such as variable values and thread state to the command queue service (`NQCQS.exe.config`) or `AMCC.exe.config` log file.

In most cases the default settings for these parameters provide optimal performance. Any adjustments you make to these settings will depend upon your environment and the kinds of communication issues you are experiencing.

If you make modifications to the `AMCC.exe.config` file you must restart the Control Center console for the changes to take effect. If you make modifications to the `NQCQS.exe.config` file, you must restart the command queue service for the changes to take effect.

Parameter values you specify in the configuration files override any default values. If a value set in one of these configuration files cannot be parsed into a number, the default value is used. If a value falls outside the allowed range, the closest allowed value is used.

The `NQCQS.exe.config` file contains additional parameters that you can configure. For more information about these additional parameters, see the *Control Center User Guide for AppManager.*

Chapter 9
Advanced Configuration for Management Servers

This chapter describes several ways you can customize and configure the operation of the AppManager management server.

Rules for Management Servers

Under normal conditions, you should not run any regularly scheduled monitoring jobs or reports on the computer you are using as the management server. If you avoid running jobs on the management server, you prevent resource competition between the agent services and the management server service and may improve management server processing capacity.

Specifically, you should avoid running jobs that perform remote monitoring operations. For example, you should not run the following jobs on the management server:

- General_MachineDown
- NT_RemoteServiceDown
- General_EventLog and General_ASCIILog
- Any Report Knowledge Scripts

Instead of running these Knowledge Scripts on the management server, you should select a specific agent computer to handle remote monitoring tasks and a specific agent computer for running reports.

In most cases, you can still use the remote monitoring Knowledge Scripts to monitor the availability of the management server, for example, by specifying the name of the management server in the list of computers to monitor when configuring the job, without running the job on the management server itself. You can also use Troubleshooter and `NetIQctrl` commands to check the operation of the management server and to diagnose problems and you can use the AMAdmin_MSHealth Knowledge Script to monitor the Windows event log for events generated by the management server. For information about using Troubleshooter and `NetIQctrl`, see "Troubleshooting and Diagnostic Tools" on page 173.

Using Anti-virus Software

In addition to restricting the monitoring jobs you run directly on the management server, you should use caution in running anti-virus software on the management server. In particular, you should not perform any real-time anti-virus scanning of the following `NetIQ` folders:

- `AppManager\dat\pioc`
- `AppManager\dat\mapqueue`
- `AppManager\bin\Cache`
- `Temp\NetIQ_Debug\`*`computer_name`*

These folders are updated frequently, and real-time scanning can cause resource contention. Therefore, you should exclude these folders from any anti-virus scanning activity.

Checking Management Server Status

As you increase the number of agents you monitor with a management server, it is also important to monitor the operational health and performance of the management server itself. The key indicators you should watch to determine the health of the management server are summarized in the following table:

Performance Counter	What to Look for
Processor: % Processor Time (All instances)	The percentage of processor time should remain less than or equal to a maximum of 80%. Although occasional spikes can be expected, the average percentage of processor time should not exceed 80%.
System: Processor Queue Length	The number of threads in the processor queue should remain less than or equal to a maximum of 3 ready threads per processor. If the number of threads in the processor queue begins to increase, it may indicate that the management server is becoming overloaded, for example, because it is attempting to process a large number of events or data points or because a slow connection has created a backlog of information to be transmitted.
NetIQms: IOC Coll. Events Queued IOC Data Queued IOC Events Queued	These counters should remain at or near zero (0), which indicates the queues are not growing. The IOC counters refer to disk-based queues that are used to store events and data when the management server is temporarily busy. Over time these should remain near empty, indicating events and data are being processed in a timely manner. If the queues grow over time, it indicates the management server cannot keep up with the load created by the agents.

If any of these counters consistently exceed the threshold indicated, or if the IOC counters grow continuously, it is an indication that the management server is either handling too many agents or that it is undersized for the load.

Changing the Polling Interval for Agent Computers

Periodically, each management server in a site checks the status of its agent computers.

There are registry keys that control how the management server determines the status of the agent computers it communicates with. These `HKEY_LOCAL_MACHINE\Software` registry keys are under `\NetIQ\AppManager\4.0\NetIQms\Config`. Because communication is handled differently for Windows agent computers and UNIX agent computers, there are separate keys for checking the status of Windows agent computers and UNIX agent computers.

Changing the Interval for Windows Computers

By default, the machine polling thread for Windows runs on the management server every 15 minutes. At each interval, the management server receives an updated list of its current agent computers and checks the availability of the designated primary management server for those agent computers.

Before changing this interval, you should evaluate the potential impact on your environment. If you lengthen the interval, it will take longer for job property or job status changes to be passed to your agent computers if the primary or backup management server fails. If you shorten the interval and have a large number of agent computers, it will increase the processing load on the management server and may degrade throughput performance. In general, if you have a large number of agent computers, you should not change the machine polling interval.

To change the machine polling interval for Windows agent computers:

1. In the Windows Registry Editor, expand `\HKEY_LOCAL_MACHINE`, to `\SOFTWARE\netiq\appmanager\4.0\netiqms\config`.
2. Double-click **Machine Poll Interval** to specify the number of seconds between updates. The default is 900 seconds. If desired, you can click the **Decimal** option to display the current value in decimal format.
3. Click **OK**.

Changing the Interval for UNIX and Linux Computers

For UNIX and Linux computers, the management server uses the agent heartbeat to determine the status of its agent computers. The registry keys that control how the management server determines the status of the NetIQ Corporation UNIX agents are the `Unix Machine Check Interval` and the `Unix Machine Timeout` keys.

At each `Unix Machine Check Interval`, the management server checks the timestamp of the last heartbeat signal from each of its UNIX agents. If the timestamp indicates that the UNIX agent has not sent a heartbeat signal within the period of time specified for the `Unix Machine Timeout`, the management server considers the UNIX agent unavailable and passes this information back to the repository and the computer is grayed out in the Operator Console.

Before changing the interval or the timeout period, you should consider the potential impact on your environment. If you lengthen the interval or the timeout setting, it may take longer to be notified when UNIX agents stop communicating with the management server. If you shorten the interval or timeout setting and have a large number of agent computers, it will increase the processing load on the management server and may degrade throughput performance. You should also keep in mind that these registry keys work in conjunction with each other so any changes should take in account both values.

To change the Unix Machine Check Interval or the Unix Machine Timeout period:

1. In the Windows Registry Editor on the management server, expand `\HKEY_LOCAL_MACHINE`, to `\SOFTWARE\netiq\appmanager\4.0\netiqms\config`.
2. Double-click **Unix Machine Check Interval** to specify the number of seconds between status checks.

   This interval controls how often the management server checks the timestamp of the last heartbeat signal from each of its UNIX agents. The default is 300 seconds. If desired, you can click the **Decimal** option to display the current value in decimal format.
3. Double-click **Unix Machine Timeout** to specify the maximum number of seconds between heartbeat signals.

   If the UNIX agent does not send a heartbeat signal within this period of time, it is deemed unavailable. The default is 1200 seconds. If desired, you can click the **Decimal** option to display the current value in decimal format.

   Note
   If you change the UNIX heartbeat interval, you may need to adjust the Check and Timeout intervals. For example, if you set a longer heartbeat interval to conserve network bandwidth, you should lengthen the Unix Machine Check and Unix Machine Timeout intervals to prevent the UNIX agent from appearing to be unavailable between heartbeat signals.
4. Click **OK**.

After you modify the registry entries, you must restart the NetIQ Corporation AppManager Management Service (`NetIQms`) for the changes to take effect. To restart the NetIQ Corporation AppManager Management Service (`NetIQms`), use the Services Control Panel.

Changing the Listening Ports

By default, the computer you designate as a management server listens on port number `9001` for communication from UNIX agents and on port `9999` for communication from Windows agents. You can modify these default ports during installation or by modifying the registry on the management server after installation.

To change the port numbers where the management server listens:

1. On the management server computer, from the Windows start menu, click **Run**, then type `regedt32.exe` to start the Registry Editor.
2. Expand the `HKey_Local_Machine\Software\NetIQ\AppManager\4.0\NetIQms` registry key.
3. Doubleclick **Port** to change the port for Windows agents or **Unix Port** to change the port for UNIX agents.
4. Select the **Decimal** option to display the current value in decimal format.
5. Type the port number you want to use.
6. Click **OK**.

For this change to take effect, you need to restart the computer.

After you change the registry entry for the Windows computer where the management server is installed, you also need to update the agent computers with the appropriate information.

- For Windows agents, modify the registry key `HKey_Local_Machine\Software\NetIQ\AppManager\4.0\NetIQmc\NetIQms Port.`
- For UNIX agents, you can change the port using UNIX Agent Manager, you can edit the default configuration file, `nqmcfg.xml`, or you can create a new configuration file. For more information, see the *AppManager for UNIX Servers Management Guide*.

Changing the Level of Detail in Trace Logging

By default, the management server records information about its operations in a log file. You can find this log file, `ms.log`, in the *`NetIQ`*`\Temp\NetIQ_debug\`*`computer`* directory where *`NetIQ`* is the AppManager installation path and *`computer`* is either the name of the computer where the management server is installed or the directory specified for the `Software\NetIQ\AppManager\4.0\Generic\Tracing\TraceLogPath` registry key.

Each line in the log file includes a timestamp in UTC format, a message type indicator, and the message body. For example:

```
987220342: info 1: computer name = MERCURY
987220342: info 1: host name = MERCURY
987220342: info 1: ip = 10.5.102.152
987220342: info 2: SocketServerThread, 2920
987220342: info 2: UnixAgentsThread, 3052
987220342: info 2: QUnixaConfigureThread, 2620
```

Typically, the information in the log contains little detail. You can, however, change the amount of information recorded in the log file by modifying registry keys.

Enabling logging for some types of information, such as data point tracing, can affect the performance of the management server. In most cases, you should use the default logging options unless you are troubleshooting problems with the management server and have been instructed by NetIQ Technical Support to trace additional information.

Changes to trace logging do not require you to restart the computer or the NetIQ Corporation AppManager Management Service (`NetIQms`).

To change the level of logging detail for the management server:

1. On the management server computer, from the Windows start menu, click **Run**, then type `regedt32.exe` to start the Registry Editor.
2. Expand the `HKey_Local_Machine\Software\NetIQ\AppManager\4.0\NetIQms\Tracing` registry key. Within this key, there are several entries for tracing management server activity. By default, all trace logging is disabled.
3. To enable tracing, select the type of tracing you are interested in, then doubleclick to open the DWORD Editor. For example, select `TraceSockets` to trace socket communication between the management server and UNIX agents or `TraceRpc` to trace the trace RPC communication between the management server and Windows agents.
4. In the DWORD Editor, select the **Decimal** option to display the current value in decimal format.

5. Set the logging level to one (1) to enable logging or to zero (0) to disable logging for the type of information you are interested in recording.
6. Click **OK**.

Your changes take effect immediately.

Moving the Management Server to a New Computer

Read the following instructions carefully before you attempt to:

- Rename the management server
- Change the IP address on the management server computer
- Replace an existing management server computer

Before you move the management server to a new computer, you must update the agent computers that communicate with the management server to enable communication with the new management server.

If you do not enable the agent computers to communicate with the new management server before you remove the old management server, you must manually update the registry on each agent computer to allow communication with the new management server computer.

To move the management server to a new computer:

1. To allow agent computers to conduct anonymous communication with the management server, run the AMAdmin_SetAllowMS Knowledge Script on all agent computers that communicate with the management server you want to move. Set the **New hostname(s) for AllowMS** parameter to an asterisk (*).

   For more information about the Knowledge Script, see the *AppManager Knowledge Script Reference Guide*.

2. ***If the following registry keys on the agent computers are not set to an asterisk (*)***, use the NTAdmin_RegistrySet Knowledge Script to add the name of the new management server computer to the key values:

   `HKEY_LOCAL_MACHINE\SOFTWARE\NetIQ\AppManager\4.0\NetIQmc\Security\AllowDosCmd`

   `HKEY_LOCAL_MACHINE\SOFTWARE\NetIQ\AppManager\4.0\NetIQmc\Security\AllowMS`

   `HKEY_LOCAL_MACHINE\SOFTWARE\NetIQ\AppManager\4.0\NetIQmc\Security\AllowReboot`

   Otherwise, certain actions will not be allowed after you move the management server. For example, the `AllowReboot` registry key will no longer allow Action_RebootSystem. For information about the parameters to specify in the Knowledge Script, see the *AppManager Knowledge Script Reference Guide*.

3. Install the management server on the new computer. For more information, see the *Installation Guide for AppManager*.

   If the new management server has a different IP address, be sure that DNS is updated and is replicating properly to other DNS servers, if applicable. The agent computers must be able to resolve the management server name to the new IP in this case before you can proceed to the next step.

4. For each agent that communicates with the management server, run the AMAdmin_SetAllowMS Knowledge Script and update the **New hostname(s) for AllowMS** parameter with the name of the new management server computer.

5. For each agent that communicates with the management server, run the AMAdmin_SetPrimaryMS Knowledge Script to update the management server name.

   Depending on whether the management server is primary or secondary for the agent, update the primary or secondary management server name.

   For more information about the Knowledge Script, see the *AppManager Knowledge Script Reference Guide*.

6. Uninstall the management server from the old computer.

7. In the TreeView pane of the Operator Console, select the computer where the old management server was installed and press **Alt+F8**. Make a note of the **ObjID** of the management server computer.

8. In Microsoft SQL Server Management Studio, right-click the QDB the management server communicates with and select **New Query**.

9. To change the status of the old management server computer to an agent computer so you can remove it from the Operator Console, in the query window, type the following SQL statement and click **Execute**:

   ```
   UPDATE dbo.Object
   SET Status = Status ^ 0x00000002
   WHERE ObjID = ObjID_of_old_management_server
   AND Status & 0x00000002 != 0
   ```

   where `ObjID_of_old_management_server` is the **ObjID** you noted in Step 7.

10. In the Operator Console, delete the old management server computer.

11. ***If the agent is still installed on the old computer***, use Control Center to add the computer and rediscover it to establish a new **ObjID** for the computer.

12. ***If you edited registry keys in Step 2***, use the NTAdmin_RegistrySet Knowledge Script to remove the name of the old management server computer from the key values.

    For information about the parameters to specify in the Knowledge Script, see the *AppManager Knowledge Script Reference Guide*.

Chapter 10
Advanced Configuration Options for Windows Agents

This chapter describes several ways you can control the flow of information from agent computers and customize the behavior of AppManager agents. Tuning the communication flow for agent computers is optional, but it allows you to tailor when and how agent computers communicate with the management server to suit your network requirements, bandwidth, latency, and operational policies.

Note
Some of the configuration options available for Windows-based agents and UNIX-based agents are similar but are controlled in different ways or through different scripts. If you are managing both Window-based agents and UNIX-based agents, be sure to review both this chapter and the *AppManager for UNIX Servers Management Guide*.

Understanding the AppManager Agent

When you install the AppManager agent on a Windows computer, the following key components are installed:

- NetIQ Corporation AppManager Client Resource Monitor agent service (`netiqmc`)
- NetIQ Corporation AppManager Client Communication Manager agent service (`netiqccm`)
- Local repository
- One or more objects for application monitoring (COM/OLE objects)

When you start a job, the Client Resource Monitor receives the job information from the management server and replies to the management server with job status (whether the job started, failed, or encountered an error).

In addition to notifying the management server of the job status, the Client Resource Monitor copies the jobs it receives from the management server into the local repository. When the agent computer is rebooted or services are stopped and restarted, the Client Resource Monitor reads the information from the local repository and restarts all of the jobs that were running before the shutdown.

If the jobs assigned to the agent computer start successfully, the Client Resource Monitor runs the local jobs, collects any data points and event information the jobs generate, and sends this information to the Client Communication Manager.

When the Client Communication Manager receives data points and event information from the Client Resource Monitor, it forwards the information to the management server as long as the management server is available to receive the information. To ensure the availability of the management server, the Client Communication Manager periodically runs a health check that polls the management server to determine its availability. If the management server is unavailable, the Client Communication Manager stores the information in the local repository until the management server becomes available.

The majority of advanced tuning options control communication between the agent services on the agent computer and the management server. By customizing this flow of information, you can optimize network communication to best suit your environment and network topology.

Understanding Agent Autonomy

As discussed in "Understanding the AppManager Agent" on page 147, the Client Resource Monitor and the Client Communication Manager work together to start jobs automatically if a computer is shut down and to retain information about jobs, events, and data in the agent computer's local repository if communication with the management server is interrupted. This default behavior is called **Autonomous** operation.

Autonomous operation requires the following:

- The Client Communication Manager service must be running to log events and data points locally on the agent computer when the agent cannot communicate with the management server.
- The Client Communication Manager service must be configured to periodically poll the management server and the Client Resource Monitor to determine the availability of the management server and whether events and data points should be stored locally or uploaded to the management server.

Although it ensures that data and events are not lost when communication with the management server is interrupted, autonomous operation requires the Client Resource Monitor and Client Communication Manager to be started together and to run continuously while a monitored computer is powered up.

Note
If the Client Communication Manager service is stopped, the Client Resource Monitor can send events and data directly to the management server. If the Client Communication Manager service is running, events and data are always passed by that service unless you explicitly disable Autonomy.

In some rare cases, you may need to temporarily disable Autonomous operation. If you disable Autonomous operation, be aware of the following:

- The Client Resource Monitor agent service must be running for jobs to run and events and data to be collected.
- The Client Resource Monitor agent service will bypass the Client Communication Manager and send events and data directly to the management server, if needed, as long as the management server is available. The Client Resource Monitor does not, however, write to the local repository. If the Client Communication Manager is stopped and the Client Resource Monitor cannot communicate with the management server, any event or data point generated while connectivity is down is lost.

Because of this potential loss of data, you should always run both agent services in Autonomous mode unless there is a specific need to temporarily stop the Client Communication Manager service and you can ensure connectivity between the Client Resource Monitor and the management server.

Disabling Autonomous Operation

To turn off autonomy so that updates occur without being routed through the Client Communication Manager service:

1. On the agent computer, in the Services Control Panel or from the command-line prompt, stop the NetIQ AppManager Client Communication Manager and NetIQ AppManager Client Resource Monitor services.
2. Start the Registry Editor and change the value of the following registry key from 1 (Autonomous operation) to 0 (non-autonomous operation):

   `HKEY_LOCAL_MACHINE\Software\NetIQ\AppManager\4.0\Netiqmc\Config\Autonomy`
3. In the Services Control Panel or from a command prompt, restart the AppManager services.

Controlling the Interval for Checking Connectivity

Periodically, the Client Communication Manager service checks the connectivity between the agent computer and the management server. Two registry keys control the interval for this checking under `HKEY_LOCAL_MACHINE\Software\ NetIQ\AppManager\4.0\NetIQccm\Config`:

Registry Key	Interval Controlled
PingMSInterval	Checking connectivity to each management server. The default interval is 30 seconds.
PollMCInterval	Polling the Client Resource Monitor service (`NetIQmc`) to find if there's been any data generated that needs to be sent to the management server. The default interval is 5 seconds.

At each `PollMCInterval`, the Client Communication Manager (`NetIQccm`) service checks the Client Resource Monitor for new event messages and collected data. If there have been any events or data collected, the service processes the information and prepares to send it to the management server or the local repository.

At each `PingMSInterval`, the Client Communication Manager checks whether it can communicate with the management server. As it checks connectivity, the service sets a flag for each management server to indicate whether communication was successful. If the flag indicates the agent computer can successfully connect to a management server, all events and data points received are uploaded. If the flag indicates the communication with the management server was not successful, the Client Communication Manager sends events and data points to the local repository until the next `PingMSInterval`.

Although you can adjust both of these intervals for checking connectivity, you should be careful about doing so. For example, if you have a slow network connection between the agent computer and the management server, you may want to lengthen the time for the `PingMSInterval` key, but this may put a strain on the agent computer and its local repository or may prevent you from seeing problems promptly.

Before changing these intervals for any agent computer, consider the following:

- The characteristics of the network connection between the managed computer and the management server computer. For example, if you have a high-speed, LAN connection, you should be able to maintain a shorter interval for checking the connection than if you have a WAN connection.
- The characteristics of the managed computer in terms of available disk, memory, and CPU for checking connectivity and storing data locally between connections to the management server.

- The type of monitoring you are doing and the intervals at which jobs run on the computer. For example, if jobs are scheduled to run at 15 minute or one hour intervals, you are less likely to generate a backlog of events and data points than when jobs run at two- or five-minute intervals.
- The frequency at which you are seeing events on the agent computer. For example, if events are rare for a particular computer, you may feel more confident in increasing the `PollMCInterval`, `PingMSInterval`, or both intervals.

Using a Windows User Account for Agent Services

On each agent computer, the Client Communication Manager (`NetIQccm`) service and the Client Resource Monitor (`NetIQmc`) service can run using either the `LocalSystem` account or a specific Windows user account. Both services must use the same account on any single agent computer. Although the `LocalSystem` account is used by default in most cases, there are many reasons to use a specific Windows account instead.

You should use a Windows account for the Client Communication Manager (`NetIQccm`) and Client Resource Monitor (`NetIQmc`) services if you are:

- Sending MAPI email from the agent computer as an action
- Enabling the reporting capability option on the agent computer to run reports
- Monitoring any Exchange Servers
- Monitoring SQL Server in Windows only authentication mode
- Running any jobs that require specific privileges or require a user account to perform specific monitoring tasks

Normally, you specify whether you want to use the LocalSystem account or a Windows user account when you install the AppManager agent. You can, however, change this information after installation, using the Services Control Panel program. If you change the account information after installation, be sure to use the same account for both agent services.

For information about Knowledge Scripts that have special requirements and changing the account you use for the agent services, see the AppManager online help.

Restarting Agent Services

By default, the AppManager agent services are started automatically whenever you restart an agent computer. Under normal conditions, the agent services are restarted using a **warm startup** that preserves information about the jobs that were running when the computer was shut down.

When the agent services are restarted using a warm startup, the Client Resource Monitor automatically restarts all of the jobs that were in progress when the computer is rebooted without requiring you to take any additional action. Because all of the job information is preserved between starts, an agent computer that uses a warm startup for the agent services is identified as being in **Persistent** mode.

If an managed computer is set to use the Persistent mode, any time jobs are interrupted because of power failures, system shutdowns, or network outages, the AppManager agent services are restarted using a warm startup.

It is also possible to perform a **cold startup** of the agent services. With a cold startup, the Client Resource Monitor does not remember the jobs that were running before the restart. Any monitoring jobs that were running are lost.

Performing a Cold Startup of the AppManager Agent

By default, the Client Resource Monitor performs a "warm" startup anytime the service is interrupted. This type of restart preserves information from ongoing jobs in the local repository. However, from time to time you may find it useful to perform a "cold" startup, which doesn't preserve information about jobs or data. For example, if you have accumulated a large number of jobs on a particular computer, you may want to remove those jobs and data.

To perform a cold startup, start the Client Resource Monitor service (`NetIQmc`.exe) using the `-oa` option from the command-line. For example, open a Command window, then type the path to the Client Resource Monitor service with these options:

```
C:\Program Files\NetIQ\AppManager\bin>netiqmc -oa
```

Once you perform a cold startup on an agent computer, you must run the discovery script, and then recreate the jobs you want to restart. For example, if you have customized the properties for a specific job but have not changed the default properties for the Knowledge Script, when you restart the job you must restore the customized property settings, such as the changes to threshold settings or data collection options. A cold start of the agent also deletes any delta discovery cache files for all delta discoveries performed by the agent.

Note
Before performing a cold startup, consider running an AppManager job report to collect details about the jobs you are running.

Setting the Agent Startup Mode

Normally, you only perform a cold startup of the AppManager agent if an agent service hangs on a computer or if you want to remove unwanted information. But if you find that you want to use this option regularly, you can manually instruct the Client Resource Monitor service to use the `-o` option at startup.

To manually set the startup parameters for the agent:

1. Click **Administrative Tools** in the Windows Control Panel, then click **Services**.
2. Select **NetIQ AppManager Client Resource Monitor** in the list of services.
3. Click **Stop** to stop the service.
4. Type `-o` in the **Start Parameters** field.
5. Click **Start**, then click **OK**.

To change the default Client Resource Monitor startup mode, you must edit a registry key.

To change the default agent startup mode:

1. Use **regedit** to find the following registry key:

   `HKEY_LOCAL_MACHINE/SOFTWARE/NETIQ/AppManager/4.0/NETIQMC/CONFIG`

2. Double-click **Persistent**.
3. Set the key value to `0` for a cold startup of the agent or to `1` for warm startup of the agent by default.

Agent Self Monitoring

In addition to monitoring the operating system, server hardware, and application resources, most organizations find it useful to monitor the operation of the AppManager components themselves. You can choose from three basic methods for monitoring the operation of AppManager agents on your Windows agent computers:

- Run the NT_RemoteServiceDown on one or more agent computers to remotely monitor the NetIQ Corporation AppManager Client Resource Monitor and NetIQ Corporation AppManager Client Communication Manager on other agent computers by listing the agent computer names for the **Machine list** parameter and `netiqmc.exe,netiqccm` for the **Services** parameter.
- Run the AMAdmin_AgentSelfMon Knowledge Script to monitor the status of the scripting engine and other low-level components that the Client Resource Monitor uses to ensure the agent is running jobs properly.

  When you run the AgentSelfMon Knowledge Script, the Client Resource Monitor sets a timestamp in the Windows registry at each interval. At subsequent intervals, the Client Communication Manager compares the timestamp value with a threshold that specifies the maximum amount of time, in seconds, that can elapse between timestamps. If the age of the timestamp value exceeds the threshold you specify, the Client Communication Manager (`netiqccm.exe`) automatically restarts the Client Resource Monitor (`netiqmc.exe`). If the timestamp is within an acceptable range, the job simply updates the timestamp value and waits for the next iteration.

- Run the AMAdmin_AgentHealth Knowledge Script to monitor the Windows Application log for events generated by the Client Resource Monitor and the Client Communication Manager that indicate general, communication, job, security, or upgrade problems. Both services log specific "self-monitoring" information, which the AgentHealth Knowledge Script can check for. You can further filter log entries by specifying a combination of include and exclude strings for the **Description** field.
- Run the AMHealth_HeartbeatWin Knowledge Script to monitor the heartbeat of the AppManager agent computer. A heartbeat is a periodic signal generated by an Appmanager agent computer to indicate that it is still running. For more information, see "AM Health Knowledge Scripts" on page 241.

Configuring Agents to Use a Hostname or IP Address

In some environments, the NetIQ Corporation AppManager Client Communication Manager (`NetIQccm`) may use an IP address instead of a hostname to locate and communicate with the management server. However, using an IP address can be problematic. For example:

- If your management server and agent computers are connected through a remote dialup and use DHCP, IP addresses are often assigned dynamically and change from one connection time to the next.
- If your management server is installed on a cluster, you must use a virtual server name associated with the cluster rather than a specific IP address.

  Note
  AppManager currently supports only Microsoft clusters.

- If you plan to periodically replace the computer you use as the management server, you may find using a hostname for communication is more convenient to change and less error-prone than maintaining an IP address.

For each management site, you should decide whether you want the NetIQ Corporation AppManager Client Communication Manager to use an IP address or hostname to locate the management server.

Once you select the best approach for your environment, you can use the AMAdmin_ConfigSiteCommType Knowledge Script to set or change the communication type. For more information about using this Knowledge Script, consult the online Help.

Configuring the Size of a Local Repository

In some environments, it is useful to be able to control the maximum size of the local repository to prevent a data or event "storm" from impacting performance. For example, if an agent computer is unable to connect to its management server for an extended period, it may generate a large number of redundant events that must be stored locally that you are not interested in transferring to the management server and repository when communication is restored. By limiting the size of the local repository, you can control both the strain put on the agent computer and the potential bottleneck involved in transferring a large number of unwanted events to the management server.

You can use the AMAdmin_SetLocalRPSize Knowledge Script to control the maximum number of events or data points that can be stored in an agent computer's local repository. If the agent computer is not able to communicate with the management server, the local repository for the agent computer will store the most recent events and data points up to this limit until communication with the management server is restored.

Note
If the number of events or data points exceeds the limit you have set (for example, because of an extended network interruption), the oldest event or data records are lost as new events or data points are recorded.

In deciding how to configure the number of rows for events and data in the local repository, assume that 1000 rows is roughly equivalent to 1 MB. You should also consider the types of jobs you run on the agent computer and the frequency with which you collect data. In a typical environment, you are more likely to generate a large number of data points for reporting purposes than a large number of events. For this reason, you may want to reduce the number of rows you reserve for events and increase the number of rows you reserve for data points on some of your agent computers.

You can use the SetLocalRPSize Knowledge Script in conjunction with the AMAdmin_SiteSchedUpload Knowledge Script to establish a schedule for the communication between each agent computer and its management server. You can also use the SetLocalRPSize Knowledge Script in conjunction with AMAdmin_SchedMaint Knowledge Script to set storage restrictions on data and events when a computer requires maintenance or in conjunction with the AMAdmin_DisableSiteComm and AMAdmin_EnableSiteComm Knowledge Scripts when you need to temporarily disable network communication with the management server.

For more information about using any of these Knowledge Scripts, double-click the Knowledge Script in the View pane of the Control Center console, select the Values tab, and click **Help**.

Adjusting the Flow of Network Traffic

The AMAdmin_ConfigSiteNetFlowCtrl Knowledge Script helps you manage network bandwidth and control the transfer of data from agent computers to the management server to suit your network capacity and make data transfers more efficient. With this Knowledge Script, you can restrict the amount of data the NetIQ Corporation AppManager Client Communication Manager sends at any one time and the frequency with which data is transferred by defining upper and lower bandwidth limits for the size of message batches transferred.

For example, assume you define a high watermark of 100 KB, a low watermark of 2 KB, and a communication interval of one hour (3600 seconds). With this configuration, the Client Communication Manager sends up to 100 KB of data per hour to the management server until the data waiting to be transferred falls below 2 KB. The Client Communication Manager then stores the data in the local repository. At the next interval, if the data to be transferred is greater than 2 KB, `NetIQccm` resumes sending the data to the management server. If the data package is still below 2 KB, Client Communication Manager continues to store the data in the local repository until the next interval.

The AMAdmin_ConfigSiteNetFlowCtrl Knowledge Script also provides dynamic tuning to allow the Client Communication Manager to respond to load changes on the management server. With dynamic tuning, each time the Client Communication Manager connects to transfer data, it checks the management server's current load. If the management server is busy and load has increased, the Client Communication Manager reduces the data sent and increases the communication interval. The Client Communication Manager continues to reduce the data sent until the amount of data falls below the low watermark or until the load on the management server decreases.

For more information about using this Knowledge Script, double-click the Knowledge Script in the View pane of the Control Center console, select the Values tab, and click **Help**.

Scheduling the Transfer of Events and Data

The AMAdmin_SiteSchedUpload Knowledge Script is used to specify a schedule for uploading data and events from an agent computer's local repository to the current management server.

This Knowledge Script allows you to store data points and events in the local repository until you're ready to upload it to the management server. By giving you the flexibility to transfer events and data during off-peak hours or when network traffic is light, the AppManager management server and repository can handle data from more servers and you can better manage network bandwidth.

For example, if you are collecting a significant amount of data on a few key agent computers, you may want to store the data locally on those agent computers while the network is busy, then transfer it to the management server at a time you know network traffic is light. In addition, you can schedule data from different agent computers to be uploaded at staggered times, further reducing the load on the management server and repository.

You can set up specific schedules for data, events, or both, as needed. The Client Communication Manager stores the events or data points in the local repository until the scheduled upload time. At upload time, Client Communication Manager reads the events or data points from the local repository and sends them to the management server.

The SiteSchedUpload Knowledge Script is often used in conjunction with other Knowledge Scripts to provide maximum control over data transfers:

- You can configure the size of message batches delivered in the upload with the AMAdmin_ConfigSiteNetFlowCtrl Knowledge Script. For more information, see "Adjusting the Flow of Network Traffic" on page 155.
- You can configure the maximum number of data points or events to store in the local repository with the AMAdmin_SetLocalRPSize Knowledge Script. For more information, see "Configuring the Size of a Local Repository" on page 154.

Best Practices. Configuring your agent computers to immediately forward events while storing performance data locally provides you with maximum flexibility in determining your transfer strategy without affecting event notification. If you are monitoring a WAN environment, however, you should use the SiteSchedUpload and ConfigSiteNetFlowCtrl Knowledge Scripts to control the data transfers from your remote agent computers to the management server.

For more information about using any of these Knowledge Scripts, double-click the Knowledge Script in the View pane of the Control Center console, select the Values tab, and click **Help**.

Configuring Designated Management Servers

As discussed in "Setting up Primary and Backup Management Servers" on page 23, the agent on each managed computer should be configured to use one **primary** management server and one **backup** management server to provide predictable failover support and static load distribution. For Windows agents, you designate the primary and backup management server during installation or by running the AMAdmin_SetPrimaryMS Knowledge Script. You can also use the AMAdmin_SetPrimaryMS Knowledge Script to change the primary management server, the secondary management server, or both.

Once you explicitly designate a primary management server, the agent services communicate exclusively with that management server.

Changing Agent Failover Configuration

By default, the Client Resource Monitor attempts to communicate with the primary management server once every minute to determine its availability. If the attempt to connect to the primary management server fails in three consecutive tries, the Client Resource Monitor determines the primary management server is not available and notifies the Client Communication Manager to begin sending events and data to the backup management server until it is able to re-establish communication with the primary management server.

You can change both the default interval and the number of connection attempts the Client Resource Monitor uses to determine the availability of the primary management server by modifying the Windows registry.

To change the failover configuration for an agent computer, expand the `\HKEY_LOCAL_MACHINE\SOFTWARE\NetIQ\AppManager\4.0\NetIQmc\config` folder:

Registry Key to Edit	Description
`PrimaryMSFailOverCtrlTimes`	The number of times the Client Resource Monitor should attempt to connect to the primary management server before failing over to a backup management server. The default is 3 attempts. You may want to increase this value if there are frequent, brief interruptions in communication or you decrease the interval. In general, you should not set this value to less than the default.
`PrimaryMSFailOverInterval`	The number of seconds between attempts to communicate with the primary management server. The default is 60 seconds. You may want to increase this value if there are frequent, brief interruptions in communication or if you use a schedule for transferring data from the agent computer. In general, you should not set this value to less than the default.

Before changing the failover configuration, you should consider the network connection between the specific agent computer and the management server. If they are connected through a wide area network or have a slow connection, you may need to increase the failover interval to prevent frequent or unnecessary failover. In practice, having an agent computer fail over from a primary management server to a backup management server should be a rare event and any changes to the failover interval or number of connection attempts should reflect this.

Removing a Designated Management Server

Normally, once you have explicitly designated a primary management server and a backup management server, there's no need to remove the designation using the AMAdmin_RemovePrimaryMS Knowledge Script. Instead, you can change the primary management server, the secondary management server, or both with the AMAdmin_SetPrimaryMS Knowledge Script. In some rare cases, however, you may want to remove a designation entirely for a computer.

Running AMAdmin_RemovePrimaryMS removes the primary and backup designations stored in the registry, potentially allowing any management server available to communicate with the agent computer, depending on the version of the AppManager agent installed and how you have configured management server restrictions with the AMAdmin_AgentConfigMSRestrictions Knowledge Script.

Keep in mind that you should only allow an undesignated management server to communicate with an agent computer if you are:

- Using a single management server and you are unsure of the management server name or suspect an incorrect name has been recorded (for example, because it has been edited manually after installation).
- Changing your site configuration from a multiple management server environment to a single management server environment and you are unsure of the management server name.
- Temporarily decommissioning both the primary management server and the backup management server and want to use any available management server until the primary and backup management servers are available or until you identify new management server names or IP addresses to explicitly designate as the new primary management server and the new backup management server.

For more information about using any of these Knowledge Scripts, double-click the Knowledge Script in the View pane of the Control Center console, select the Values tab, and click **Help**.

Manually Controlling Network Communication

You can use the AMAdmin_DisableSiteComm Knowledge Script to temporarily disable network communication from a Windows agent computer to the management server and repository. When communication is manually disabled with this Knowledge Script, information about events, data, and job status is stored in the local repository. The information in the local repository is then transferred to the management server when communication is re-enabled with the EnableSiteComm Knowledge Script. As discussed in "Adjusting the Flow of Network Traffic" on page 155, you can use the AMAdmin_ConfigSiteNetFlowCtrl Knowledge Script to configure the size and frequency of the batches to be transferred when communication is re-enabled.

These Knowledge Scripts allow you to intentionally stop the communication between agent computers and management servers, and by extension, their respective site repositories, at any time. For example, if you are experiencing network problems, you may want to temporarily disable communication while you troubleshoot the problem or if you are experiencing high network activity, you may want to disable communication to store data locally on an agent computer until the demand for server bandwidth is reduced.

For more information about using any of these Knowledge Scripts, double-click the Knowledge Script in the View pane of the Control Center console, select the Values tab, and click **Help**.

Controlling Access to an Agent's Local Repository

You can use standard Windows file system security to control who can access the local repository on an agent computer.

To set specific permissions for individual users or for groups:

1. Select the local repository file. By default, the local repository is located in the AppManager installation folder under `NetIQ\AppManager\db\Local-Repository.mdb`. For example, if you installed the AppManager agent in the folder `C:\Program Files\NetIQ\AppManager`, the default location for the local repository is `C:\Program Files\NetIQ\AppManager\db\Local-Repository.mdb`.
2. Right-click the file, then click **Properties**.
3. Click the **Security** tab.
4. Click **Add** to add users and groups to the access control list, if needed.
5. Select the permissions you want to Allow or Deny for each user and group. For example, to prevent a user or group from accessing the local repository, in the Deny column click **Full Control**. Similarly, to give a user or group permission to view but not to modify the local repository, in the Allow column click **Read**.

   Note
   In setting permissions for users and groups, be sure you do not change the permissions for the account the agent services use. If the agent services on an agent computer run under a Windows user account, that account must be allowed Full Control for the local repository file.

   If users are denied access to the local repository and attempt to open the file, they will see a message similar to the following:
6. Click **OK** when you have finished setting user permissions to access the local repository.

Chapter 11
Optimizing Performance

AppManager console command-line parameters enable you to specify logon information or customize the layout of the Operator Console when you start console programs. This chapter discusses how to use these command-line parameters and Operator Console filtering to optimize performance.

Using Command-line Parameters

AppManager command-line parameters allow you to specify logon information and customize the Operator Console layout from a command prompt or a shortcut. Specifying logon information at the command-line reduces the time it takes to start console programs. You can also use the command-line parameters to selectively hide and display Operator Console panes to improve the performance of the Operator Console.

To use command-line parameters, you must first open a command prompt window or a shortcut. You can open a command prompt by clicking **Start > Programs > Accessories > Command Prompt** or by clicking **Start > Run**, then typing or selecting `cmd`.

Specifying Logon Information

You can use command-line parameters to specify logon information when you start any AppManager console program that connects to the repository from a shortcut or a command prompt. For example, you can specify the logon information when starting any of the following programs:

- Operator Console (`netiq.exe`)
- Chart Console (`AMChartCon.exe`)
- Repository Browser (`dbBrowser.exe`)
- Security Manager (`SecMgr.exe`)

When using command-line parameters to specify logon information, consider your security requirements. For example, you may want to prompt for username and password information rather than store this information as a command-line parameter in a shortcut.

If you don't want to store password information in a command-line parameter, another option is to use the /TRUST parameter and validate the user based on the Windows user information. If SQL Server authenticates the Windows user account, you are allowed to log on. In this case, only two parameters, /REPOSITORY and /SERVER, are required to log on, and can be entered at the command-line or stored as command-line parameters in a shortcut.

Note
There are no command-line parameters for login information available for the Control Center console.

To specify logon information, use the following command-line parameters:

Parameter	Description
/S Server	The name of the SQL Server that manages the AppManager repository. When specifying a computer name, you can enter the Windows computer name or the IP address. For example, to specify a named instance on SQL Server 2000, you can enter 10.1.10.443\INST1.
/R Repository	The name of the AppManager repository you want to work with. For example, the default database name for AppManager is QDB.
/TRUST	Validate the user based on the Windows user information. The AppManager console computer needs to be part of a Windows domain or workgroup. The user needs to be validated as a Windows user before access is granted. Optional parameter. If you do not specify this parameter, SQL Server uses the standard SQL Server security validation to validate a specified login name and password.
/N Name	The user name of the SQL Server login account used to access the AppManager repository. This parameter is ignored when the /TRUST parameter is specified. **Note** The SQL Server login account must have permission to access AppManager. For more information about granting access to AppManager to SQL Server login accounts, see the *Installation Guide for AppManager*.
/P Password	The password for the SQL Server login account. This parameter is ignored when the /TRUST parameter is specified.

Customizing the Operator Console Layout

If you are monitoring a large number of servers, you can use command-line parameters to improve the startup performance of the Operator Console (netiq.exe) by customizing the console layout at startup.

If you start the Operator Console from a shortcut or command prompt, you can use command-line parameters to display only the tabs and panes that you want to view. Reducing the number of panes and tabs displayed can greatly improve the startup time and performance of the Operator Console. Through filtering and hiding views, tabs, and panes to only display the specific information you are interested in you can ensure maximum efficiency when you are working with the Operator Console.

Note
There are no command-line parameters for customizing the layout of the console available for the Control Center console.

To customize the Operator Console display, use the following command-line parameters:

Parameter	Description
/SHOWMASTERVIEW	Shows the Master view and splits the TreeView pane. Any other available views are hidden on startup. To show other available views, in the Operator Console, click **View > View Manager**, and select the available views. You can hide the right panel of the split TreeView pane by right-clicking the right panel and deselecting **Show Details Panel** (the TreeView pane remains split).
/SHOWEVENTSONLY	Shows event information in the List pane (and dims all other tabs and panes, including the TreeView pane). This parameter: • Optimizes the Operator Console startup by viewing events only. If you minimize the Operator Console, the taskbar button will not flash when events are open. • Dims the layout options in the Console tab of the Operator Console Preferences dialog box. To change the console layout preferences, restart the Operator Console without using this parameter.
/SHOWALL	Shows all tabs in the List pane—previously dimmed panes can be shown. This parameter overrides any other parameters and selects all of the console layout options in the Console tab of the Operator Console's Preferences dialog box. **Note** To show a hidden pane, in the Operator Console, click **View > TreeView Pane**, **List Pane**, **Knowledge Script Pane**, and **Graph Pane**.
/HIDEEVENTS	Dims the Events tab in the List pane and event-related menu commands (for example, List > Acknowledge Event). This parameter: • Always shows the TreeView pane • Deselects the **Work with events** check box in the Console tab of the Operator Console's Preferences dialog box To work with events: • Start the Operator Console using the /SHOWALL parameter or another /HIDE*name* parameter. • Select the **Work with events** check box in the Console tab of the Operator Console's Preferences dialog box.
/HIDEJOBS	Dims the Jobs tab in the List pane, job-related menu commands (for example, List > Close Job), and the Knowledge Script pane. This parameter: • Always shows the TreeView pane • Deselects the **Work with jobs** check box in the Console tab of the Operator Console's Preferences dialog box To work with jobs: • Start the Operator Console using the /SHOWALL parameter or another /HIDE*name* parameter. • Select the **Work with jobs** check box in the Console tab of the Operator Console's Preferences dialog box.

Parameter	Description
/HIDEDETAILS	Dims the Details tab in the List pane. This parameter: • Always shows the TreeView pane • Deselects the **View details** check box in the Console tab of the Operator Console's Preferences dialog box To view details: • Start the Operator Console using the /SHOWALL parameter or another /HIDE*name* parameter. • Select the **View details** check box in the Console tab of the Operator Console's Preferences dialog box.
/HIDEGRAPHS	Dims the Graph Data tab in the List pane, graph data-related menu commands (for example, Graph > Delete Graph), and the Graph pane. This parameter: • Always shows the TreeView pane • Deselects the **Work with graphs** check box in the Operator Console's Preferences dialog box Console tab To work with graphs: • Start the Operator Console using the /SHOWALL parameter or another /HIDE*name* parameter. • Select the **Work with graphs** check box in the Console tab of the Operator Console's Preferences dialog box.
/VIEW	Shows a specific view tab by view name. This parameter: • Always shows the TreeView pane • Limits the view displayed to the one specified by the parameter To display other tabs, start the Operator Console using the /SHOWALL parameter

Disabling Proxy Events in the TreeView Pane

To improve the overall performance of the Operator Console, you can disable the **Display "AppManager Proxy Events" as objects in treeview** option. This option is disabled by default.

This option displays event information for a computer that is monitored by a proxy Knowledge Script under its own parent event. For example, if you are monitoring the SJCCOMEYT01 and SJCCOMEYT03 computers with General_MachineDown, event information for each computer is organized under its own parent event.

To disable displaying proxy events as objects in the Treeview pane:

1. In the Operator Console, click **File > Preferences**.
2. In the Repository tab, click **Event** to open the Preference - Event Options dialog box.
3. Deselect the **Display "AppManager Proxy Events" as objects in treeview** option.
4. Click **OK** to apply your changes.

Getting Help for Command-line Parameters

The Operator Console provides information about command-line parameters. To view command-line Help from a command prompt window, type `netiq.exe` and `/?` or `/HELP`. For example, change to the AppManager `bin` folder and type `netiq.exe /HELP` to display the Help.

Chapter 12
Developing Scripts to Perform AppManager Tasks

AppManager allows you to automate many common tasks using command-line scripts and calls to the NetIQOLE automation object. This chapter discusses several sample scripts included with AppManager and explains how they are used to perform common AppManager tasks. For information about the NetIQOLE object, see the *NetIQ OLE Object Reference Guide for AppManager.*

Understanding Command-line Scripting

AppManager allows you to automate many common tasks using command-line scripts and calls to the NetIQOLE automation object. The NetIQOLE automation object uses the ODBC SQL Server driver to connect to the SQL Server where a QDB has been installed and can be used in conjunction with a scripting host to enable you to perform many AppManager activities without using the Operator Console interactively. Command-line scripts can be especially powerful because they give you the flexibility to create batch files that can:

- Encapsulate multiple activities
- Automate frequent tasks
- Run unattended to perform tasks at off-peak hours
- Carry out site-specific policies

For example, you may want to create batch files that start jobs or automatically acknowledge certain types of events.

To illustrate how you can use NetIQOLE in scripts, AppManager includes several sample scripts located in the `c:\Program Files\NetIQ\AppManager\scripts` folder on the computer where the Operator Console is installed.

About the Sample Command-line Scripts

AppManager includes several sample command-line scripts and a sample scripting host, `netiqcmd`, for running the scripts. These sample scripts only illustrate a few common activities. Depending on your level of programming skill and knowledge of AppManager, far more complex scripting is possible through NetIQOLE.

The following sample command-line scripts are available:

Script Name	Description
`ackevent.vbs`	Acknowledges an existing AppManager event.
`AddChildJob.vbs`	Adds a child job to an existing parent job.
`AddToMonPolicy.vbs`	Adds a Knowledge Script group to a monitoring policy.
`CloseEvent.vbs`	Closes an existing AppManager event.
`CloseJob.vbs`	Closes an existing AppManager job.
`CreateJob.vbs`	Creates a new AppManager job on a specified target computer.
`CreateKSGroup`	Creates a new Knowledge Script Group.
`DeleteEvent.vbs`	Deletes an existing AppManager event.
`DeleteJob.vbs`	Deletes an existing AppManager job.
`DumpGraph.vbs`	Dumps graph data from a data stream in comma-delimited form to the computer screen or a file.
`KSCheckin`	Checks a Knowledge Script into the QDB.
`KSCheckout`	Checks a Knowledge Script out of the QDB.
`RemoveFromMonPolicy`	Removes a Knowledge Script Group from a monitoring policy.
`RemoveKSGroup`	Deletes a Knowledge Script Group.
`StartJob.vbs`	Starts an existing AppManager job.
`StartKSGroup`	Creates new jobs based on the members of the Knowledge Script Group.
`StopJob.vbs`	Stops an existing AppManager job.

Refer to the command-line reference Help for each sample script to determine the information each script requires to run. For example, the `AckEvent.vbs` script requires the event ID.

At a command prompt, type `netiqcmd.exe` and the script name. For example, to display usage information for the `AckEvent.vbs` script, type:

```
c:\Program Files\NetIQ\AppManager\scripts\netiqcmd.exe ackevent.vbs
```

The Help includes a brief description of the sample script, a usage example, and a list of script parameters.

Running AppManager Command-line Scripts

To run AppManager command-line scripts, you must log on to the AppManager repository with a SQL Server login account that has permission to access AppManager. For more information, see "Changing User Roles" on page 40.

Open a command prompt, and type the path to `netiqcmd.exe`, followed by the command-line script filename and one or more required and optional parameters. For example:

```
c:\Program Files\NetIQ\AppManager\bin\netiqcmd.exe startjob.vbs /jobid=19
```

Include the path to `netiqcmd.exe` if you plan to run the script (or a batch file with a script command-line statement) from a scheduling program, such as the Task Scheduler. It is not necessary to include the path to the command-line script file.

Note
The required and optional parameters required vary depending on the purpose of the script. All scripts require information for logging on to the QDB. You can specify the logon information at the command-line or by creating a default logon profile in a text file.

If you have Windows Scripting Host (WSH) installed in your environment, you can also run scripts using WSH instead of using `netiqcmd.exe`. The syntax for running scripts with Windows Scripting Host is similar. For example:

```
cscript deleteevents.vbs /eventid=5
```

Creating a Default Logon Profile

All AppManager command-line scripts require you to log on to an AppManager repository. By default, AppManager scripts use the following login information:

Logon Parameter	Default
`/USER`	Windows user account name with which you logged on.
`/PASSWORD`	Password for the current Windows user account.
`/SERVER`	Windows name of the local computer.
`/DATABASE`	Repository name of `QDB`.

To create your own default logon profile for AppManager scripts, add a section called `[Default Logon]` to the `netiq.ini` file (located in the `%SYSTEMROOT%` directory) and enter the logon parameters. For example:

```
[Default Logon]
USER=FRED
PASSWORD=SCOOTER
SERVER=SHASTA
DATABASE=MY_QDB
DEBUGGING=TRUE
```

Note
In creating a default logon profile, consider your security requirements. For example, you may want to include only the /SERVER and /DATABASE parameters, requiring the /USER and /PASSWORD parameters to be entered at the command-line.

If you do not want to include the password in a default logon profile, change the SQL Server security mode to minimize the arguments entered at the command-line.

With Windows Authentication or Mixed security modes, SQL Server uses Windows security to authenticate a Windows user at the computer where the script is being run. The Windows user is allowed to log on if authenticated by SQL Server. In this case, only two parameters (`/DATABASE` and `/SERVER`) are required to log on. These can be included in the `netiq.ini` file or entered at the command-line.

Creating Jobs

The `CREATEJOB.VBS` script creates a new job on a specific computer. This script does not allow you to set properties from the command-line. Therefore, you should use this script only when creating jobs that use default Knowledge Script properties or when you have created custom Knowledge Scripts with the appropriate properties, including the scheduling interval, parameter values, actions, action parameters, and advanced options.

The following example illustrates the command-line statement for this script:

```
c:\Program Files\NetIQ\AppManager\bin\netiqcmd.exe createjob.vbs
/user=miles /password=pwd /server=shasta /database=qdb1 /ksname=NT_CpuLoaded /
target=mango
```

This script uses the following command-line arguments:

Parameter	Description
`/USER`	username of the SQL Server login account used to access the AppManager repository. Not required if using Windows authentication.
`/PASSWORD`	Password for the SQL Server login account. Not required if using Windows authentication.
`/SERVER`	Windows name of the server where the AppManager repository you want to work with is installed. The default is the name of the local computer. Not required if using the default.
`/DATABASE`	Name of the AppManager repository you want to work with. The default AppManager repository name is `QDB`. Not required if using the default.

Parameter	Description
/DEBUGGING	Flag to turn on debugging. If set to TRUE, the script displays descriptive information about any SQL errors that occur in a message box. The default is FALSE. Not required if using the default.
/KSNAME	Name of the Knowledge Script you want to run on the selected computer. Include the category prefix for the Knowledge Script. For example, Winbasic_CpuLoaded. Be sure the Knowledge Script properties have been set properly and saved, either under the existing Knowledge Script name or with a new Knowledge Script name. This parameter is required.
/TARGET	Name of the computer where you want the Knowledge Script to run. You can only specify one computer name. Server groups and multiple computer names are not supported. This parameter is required.

Starting, Stopping, Closing, and Deleting Jobs

Use these AppManager command-line scripts to work with existing jobs:

- STARTJOB.VBS changes the state of the job to running.
- STOPJOB.VBS changes the state of the job to stopped.
- CLOSEJOB.VBS changes the status of the job to closed.
- DELETEJOB.VBS deletes the job.

The following example illustrates the syntax for these scripts:

```
c:\Program Files\NetIQ\AppManager\bin\netiqcmd.exe closejob.vbs
/server=srv1 /user=miles /database=qdb /jobid=5
```

All of these scripts use the following command-line arguments:

Parameter	Description
/USER	Username of the SQL Server login account used to access the AppManager repository. If using Windows authentication, this parameter is not required.
/PASSWORD	Password for the SQL Server login account. If using Windows authentication, this parameter is not required.
/SERVER	Windows name of the server where the AppManager repository you want to work with is installed. The default is the name of the local computer. Not required if using the default.
/DATABASE	Name of the AppManager repository you want to work with. The default AppManager repository name is QDB. Not required if using the default.

Parameter	Description
/DEBUGGING	Flag to turn on debugging. If set to TRUE, the script displays descriptive information about any SQL errors that occur in a message box. The default is FALSE. Not required if using the default.
/JOBID	Identifier for the existing job you want to start, stop, close, or delete. This parameter is required. For example: /jobid=5

Acknowledging, Closing, and Deleting Events

Use these AppManager command-line scripts to work with existing events:

- ACKEVENT.VBS acknowledge the event.
- CLOSEEVENT.VBS closes the event.
- DELETEEVENT.VBS deletes the event.

The following example illustrates the syntax for these scripts:

```
C:\Program Files\NetIQ\AppManager\bin\netiqcmd.exe ackevent.vbs
/server=srv1 /user=miles /password=pwd /database=qdb1 /eventid=5
```

All of these scripts use the following command-line arguments:

Parameter	Description
/USER	username of the SQL Server login account used to access the AppManager repository. If using Windows authentication, this parameter is not required.
/PASSWORD	Password for the SQL Server login account. If using Windows authentication, this parameter is not required.
/SERVER	Windows name of the server where the AppManager repository you want to work with is installed. The default is the name of the local computer. If using the default, this parameter is not required.
/DATABASE	Name of the AppManager repository you want to work with. The default AppManager repository name is QDB. If using the default, this parameter is not required.
/DEBUGGING	Flag to turn on debugging. If set to TRUE, the script displays descriptive information about any SQL errors that occur in a message box. The default is FALSE. If using the default, this parameter is not required.
/EVENTID	Identifier for the event you want to acknowledge, close, or delete. This parameter is required. For example: /eventid=5

Exporting Data Streams

The `DUMPGRAPH.VBS` script exports data streams in comma-delimited format to a computer screen or text file.

The following is an example of the syntax for exporting data to the screen:

```
c:\Program Files\NetIQ\AppManager\bin\netiqcmd.exe dumpgraph.vbs
/server=srv1 /database=qdb1 /graphid=5
```

The following is an example of the syntax for exporting data to a file:

```
c:\Program Files\NetIQ\AppManager\bin\netiqcmd.exe dumpgraph.vbs
/user=miles /password=pwd /server=srv1 /database=qdb /graphid=5 > output.txt
```

This script uses the following command-line arguments:

Parameter	Description
`/USER`	Username of the SQL Server login account used to access the AppManager repository. Not required if using Windows authentication.
`/PASSWORD`	Password for the SQL Server login account. Not required if using Windows authentication.
`/SERVER`	Windows name of the server where the AppManager repository you want to work with is installed. The default is the name of the local computer. Not required if using the default.
`/DATABASE`	Name of the AppManager repository you want to work with. The default AppManager repository name is `QDB`. Not required if using the default.
`/DEBUGGING`	Flag to turn on debugging. If set to `TRUE`, the script displays descriptive information about any SQL errors that occur in a message box. The default is `FALSE`. Not required if using the default.
`/GRAPHID`	Identifier for the existing graph data stream you want to output. This parameter is required. For example: `/graphiid=20`
`/SHOWUTC`	Flag to control date formatting. If set to `TRUE`, date/time fields are displayed in Coordinated Universal Time (UTC) format. If set to `FALSE`, date/time fields are converted to a mm/dd/yy hh:mm:ss format. This parameter is not required.

Scheduling Scripts to Run

You can use the Microsoft Task Scheduler or another scheduling tool to create automated tasks to execute one or more AppManager command-line scripts on the computers you choose. To do this, you should create a batch file that starts the scripting host and runs the script(s) that perform the AppManager task you want to automate. For example, if you are using one of the sample command-line scripts provided with AppManager, the batch file might look similar to this:

```
REM
call c:\program files\netiq\appmanager\bin\Netiqcmd.exe startjob.vbs /
SERVER=alien1 /DATABASE=NYQDB /JOBID=22
```

Notes

- The scheduling program you use must interact with the Desktop and allow you to log on as a specific Windows user. Service-based scheduling programs such as SQL Server Management Studio (with its Scheduled Task feature) and the Windows AT command do not meet these requirements. They only allow you to log on under the system account, which doesn't have the full set of permissions needed to run AppManager command-line scripts, and they do not allow the command-line script to interact with the Desktop.
- Be sure to modify the command-line statement in the batch file to reflect the proper path to the scripting host and valid logon information for the repository.

Getting Help for Sample Scripts

Information is available to help you run AppManager command-line scripts. At a command prompt, type `netiqcmd.exe` and the script name. For example, to display usage information, type:

```
c:\Program Files\NetIQ\AppManager\scripts\netiqcmd.exe startjob.vbs
```

The Help includes a brief description of the sample script, a usage example, and a list of script parameters.

Chapter 13
Troubleshooting and Diagnostic Tools

This chapter describes how to use AppManager diagnostic tools and utilities to retrieve information about the NetIQ Corporation AppManager Management Service and AppManager agents and to identify problems within your environment. These tools and utilities allow you to view current activity and configuration settings for specified computers and are used primarily for diagnostic analysis and troubleshooting.

Understanding What AppManager Provides

AppManager provides several ways for you to uncover information about the activity of AppManager components and your AppManager deployment and locate and resolve any problems that may prevent you from monitoring your environment. With these troubleshooting and diagnostics tools, you can check AppManager activity, network communication, and configuration information for your agent computers, management servers, and the management site.

The following tools are available to perform these tasks:

- **Troubleshooter** is available through the Control Center console and the Operator Console and enables you to report information about management server and agent communication, server maintenance, job status, and configuration details such as a computer's time zone setting and upload schedule.
- **NetIQCtrl** provides a command line interface for accessing the information available using the Troubleshooter and options for a small number of additional reports that are not available through the Troubleshooter.
- **NetIQ Diagnostics** gathers log files and registry information for AppManager components. You can run the NetIQ Diagnostics utility on any computer that has at least one AppManager component installed.
- **Tracing registry keys and component log files** allow you to configure the level of log file tracing for AppManager components. Component log files can provide detailed information about the activity of AppManager components, depending on the level of tracing enabled. Changing the level of tracing may require editing the registry, however, and therefore should only be done if you have exhausted other sources of information or are instructed to do so by NetIQ Technical Support.
- **Log Analysis Tool** parses UNIX agent log files to consolidate job information, making the file contents easier to interpret.
- The AM Self Monitoring module, also known as AM Health, provides monitoring of the health and availability of SQL Server resources for the QDB, the Management Server, and the Control Center components, as well as monitoring of the AppManager agent heartbeat. For more information, see "AM Health Knowledge Scripts" on page 241.

Using the Troubleshooter

The Troubleshooter utility provides access to many different types of diagnostic reports about AppManager management servers and agent services through an easy-to-use console interface. Through the Troubleshooter, you can retrieve information about management server and agent communication, detailed and summary job status, detailed operational statistics, and configuration details such as a computer's time zone setting and upload schedule.

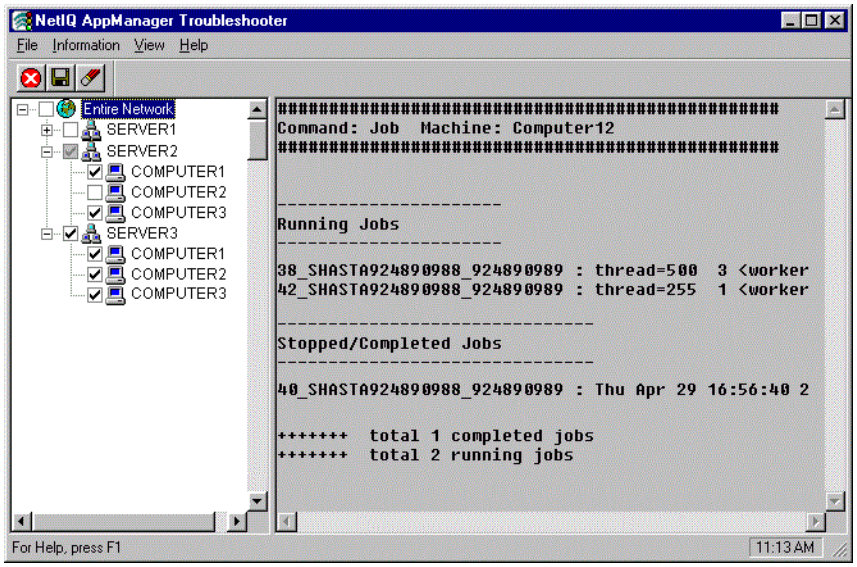
The Troubleshooter is not applicable when diagnosing issues on UNIX or Linux computers.

To use the Troubleshooter:

1. Open the Control Center console.
2. Select a Server view in the Enterprise Layout pane.
3. Right-click a server in the View pane, click **Utilities > Job Troubleshooter**, and then select a specific Troubleshooter report to view. For example, click **Utilities > Job Troubleshooter > Event Collapsing Summary**.

   For information about the information types and specific reports, see "Selecting Specific Troubleshooter Reports" on page 176.

   Once you select a report, the Troubleshooter window is displayed with the information you have selected displayed in the right pane.



Note
You can choose to hide or display the toolbar and status bar can by selecting **Toolbar** or **Status Bar** from the View menu. When displayed, a check mark appears by the menu item.

4. Once you open the Troubleshooter window, it remains displayed, allowing you to select additional computers and reports, until you close the window by clicking **File > Exit**.

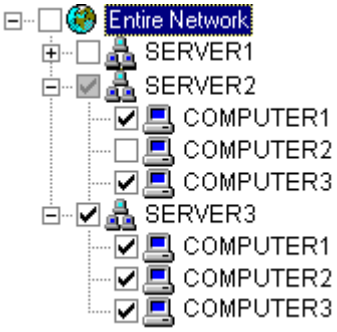
Generating Reports from within Troubleshooter

Once you have opened Troubleshooter from the Operator Console, you can generate additional reports for one or more computers or domains in your environment.

To generate reports with TroubleShooter:

1. Select the computers or domains you want to generate reports for by checking the box next to the computer or domain name.

   - If you check a **domain**, Troubleshooter generates diagnostic reports for each computer within that domain.
   - If you check **Entire Network**, Troubleshooter generates diagnostic reports for all of the computers in all of the domains.
   - If you select **one or more computers** within a domain, Troubleshooter generates diagnostic reports only for the computers you have checked.



2. After selecting the appropriate computers and domains, click **Files > Troubleshooter >** *information type* **>** *report* to display the report in the Information pane.

   For information about the information types and specific reports, see "Selecting Specific Troubleshooter Reports" on page 176. Depending on the report select, you may need to use the scroll bar to see an entire report.

Note

The information for any new report you run is appended to any existing information in the Information pane. The pane is not cleared of previous information until you explicitly decide to do so. Appending the information in this way allows you to scroll through or export all of the information for a computer or group of computers without having to rerun reports. For more information, see "Clearing the Diagnostic Report's Information Pane" on page 178.

---

3. To close Troubleshooter, click **File > Exit** from the menu.

Selecting Specific Troubleshooter Reports

Troubleshooter reports are organized by the type of information they provide. For each information type, you can select from several different reports to investigate specific areas of interest. In some cases, reports may overlap and provide similar information or give you the option to view information in summary or in detail form.

Before you select a specific report, you need to identify the general type of information you are interested in reviewing from the following information types:

- Job Info Reports
- Client Resource Monitor Info Reports
- Client Communication Manager Info Reports
- Management Service Info Reports

When you click an information type, you will see a list of the specific reports associated with that type. All of the reports are based on the output from `NetIQCtrl` commands. If you are not sure which report to run, you may want to review the sample output for each report by looking up the corresponding `NetIQCtrl` command in "Using the Command-line Program NetIQctrl" on page 179.

Job Info Reports

Job Info reports provide detailed and summary information about the jobs that are running, stopped, or pending on the selected agent computer or group.

Report	Description
Job Summary	Summary of all running and stopped or completed jobs for an agent computer. This report displays the output from the `job` command.
Event Collapsing Summary	Summary of event collapsing information including the collapse interval and the number of collapsed events for all jobs on an agent computer. This report displays the output from the `jobevt` command.
Maintenance Summary	Status information for jobs that are inactive on an agent computer during scheduled maintenance periods. This report displays the output from the `jobrsc` command.
Event Collapsing Detail	Detailed event collapsing information for all jobs on an agent computer. This report displays the output from the `profevt` command.
Job Detail	Detailed information about all jobs running on an agent computer. This report displays the output from the `profile` command.
Maintenance Detail	Detailed status information for jobs that are inactive on an agent computer during scheduled maintenance periods. This report displays the output from the `profrsc` command.

Client Resource Monitor Info Reports

Client Resource Monitor Info reports provide information about the operation, connectivity, and configuration for the NetIQ AppManager Client Resource Monitor service running on the selected agent computer or group.

Report	Description
Active Management Servers	A list of management servers that are communicating with an agent computer and the network communication status. Displays output from the `listms` command.
Application Monitoring Status	A list, by application, of jobs running on an agent computer and the applications affected by scheduled maintenance periods. Displays output from the `listrsc` command.
Time Zone Setting	The time zone setting of an agent computer. Displays output from the `localtime` command.
Connectivity	Verification that the Client Resource Monitor service is running on the server and configuration information for the service. Displays output from the `ping` command.
Statistics	Statistical information collected by the Client Resource Monitor service on an agent computer. Displays output from the `stat` command.
Upload Schedule	The upload schedule that has been defined on an agent computer for all management sites. Displays output from the `uploadsched` command.

Client Communication Manager Info Reports

Client Communication Manager Info reports provide information about the operation, connectivity, and configuration for the NetIQ AppManager Client Communication Manager service running on a selected agent computer or group.

Report	Description
Connectivity	Verification that the Client Communication Manager service is running on the server and configuration information for the service. Displays output from the `ping` command.
Active Management Sites	A list of all management sites that are monitoring an agent computer and configuration information for the Client Communication Manager. Displays output from the `listsite` command.
Statistics	Statistical information collected by the Client Communication Manager service on an agent computer. Displays output from the `stat` command.

Management Service Info Reports

Management Service Info reports provide information about the operation, connectivity, and configuration of the NetIQ AppManager Management Server service and the agent computers for which the management server is responsible.

Report	Description
Configuration Information	Configuration information for a management server. Displays output from the `infoconfig` command.
Managed Client Information	Information collected by a management server for all agent computers. Displays output from the `machine` command.
Time Zone Setting	The time zone configuration of a management server. Displays output from the `localtime` command.
Connectivity	Verification that the NetIQ Corporation AppManager Management Service is running on the server and configuration information for the service. Displays output from the `ping` command.
Management Site Information	Information about the management site that a management server is configured to monitor. Displays output from the `site` command.
Statistics	Statistical information collected by the Management Server service on a management server. Displays output from the `stat` command.
Threads	Control thread statistics maintained by a management server. Displays output from the `thread` command.

Clearing the Diagnostic Report's Information Pane

Unless you clear the Information pane, the information for any new report you run is appended to any existing information in the Information pane. Having new information appended to existing information allows you to scroll through or export all of the information for a computer or group of computers without having to rerun reports.

When attempting to diagnose activity on multiple computers or running multiple reports, however, you may want to clear the Information pane periodically to make the reports more readable.

To clear all information from the Information pane, click **Information > Clear** from the menu or click the **Clear Information** button on the toolbar.

Exporting a Diagnostic Report

The information returned by Troubleshooter can be exported to a text (`.TXT`) file.

To export a diagnostic report to a text file:

1. In the TreeView pane, right-click and select **Export** or click the Export button on the toolbar.
2. In the Save As dialog box, select a location to save the file and name the report.

   Note
   If the file already exists, the information is appended to the end of the file.

3. Click **Save As**.

Using the Command-line Program NetIQctrl

The `NetIQctrl` command-line program is an interactive program that allows you to view current activity and configuration settings for specified computers. It is used primarily for diagnostic analysis and troubleshooting.

Most of the information available through `NetIQctrl` commands is also available by clicking **TreeView > Troubleshooter** in the Operator Console. The command output is identical to what's displayed in the Troubleshooter Information pane. However, some report options are only available from the command line using `NetIQctrl`.

Starting NetIQctrl

You can start the `NetIQctrl` program from the AppManager Operator Console by clicking **Extensions > NetIQCtrl**, or from a Command Prompt window by running the executable `NetIQctrl.exe` (located in the AppManager `bin` directory).

Once you start the `NetIQctrl` program, you enter commands on the command line. The general format for `NetIQCtrl` commands is:

`command hostname component` [*`options`* …]

Parameter	Description
`command`	One of the available NetIQCtrl commands.
`hostname`	The name or IP address of the computer where the AppManager management server or agent is running. In the syntax descriptions: • `ms_hostname` indicates you should use the name of the computer where the management server is running. • `mc_hostname`.indicates you should use the name of an agent computer.
`component`	The appropriate AppManager component. • `NetIQms` • `NetIQmc` • `NetIQccm` Some commands can apply for more than one component, but you can only retrieve information for one component at a time. In addition, some commands require you to use a keyword in the command line to control the information retrieved.
`[options …]`	One or more optional parameters, such as a job ID number, that limit or refine the type of information that the command displays.

Available NetIQctrl Commands

The following table lists the `NetIQctrl` commands and describes how to use them. Most commands include an example of the output they produce. These examples are only intended to illustrate the type of information returned. You may see different or additional information when you run these commands.

Command	Description
debug	Sets a debug category and level to start tracing activity on a management server or an agent computer. The `debug` command is for diagnostic purposes only and is therefore of primary interest to programmers who are developing custom Knowledge Scripts. The categories and levels you specify depend on the component, `NetIQmc` or `NetIQms`, you want to trace. The higher you set the debugging level, the more verbose the debugging output. The basic syntax is: `debug` *`mc_hostname`* `NetIQmc` *`debug_category debug_level`* `debug` *`ms_hostname`* `NetIQms` *`debug_category debug_level`* For example, to set moderate tracing for the AppManager agent on the agent computer `shasta`, use: `debug shasta NetIQmc MC_ALL 256` To control where tracing output is sent, use the `output` command. For a list of tracing categories and levels, use the `info` command.

dictget	Displays Server-Side Job Configuration (SSJC) parameters from an agent's local repository. The basic syntax is:
	`dictget` *`mc_hostname`* `NetIQmc [`*`ms_host`*`]` *`param_name`*`|get_all`
	You can retrieve:
	• All of the Server-Side Job Configuration parameters in the dictionary using the `get_all` keyword.
	• Parameters with a specific name or with names that start with a specific string using a wild card (for example, `_NQ_L*`). Note that only trailing wild cards are supported. You can't specify a pattern such as `*logical*`.
	To retrieve Server-Side Job Configuration parameters associated with a specific AppManager repository, specify the name of a management server that communicates with that repository.
	For example, if the agent computer `starlet` is managed by two management servers, `sunset` and `venice`, with data and events stored in separate repositories, you may want to specify which repository you are interested in by including the management server name in the command line:
	`dictget starlet NetIQmc venice get_all`
	Output example
	`Results for site VENICE1029277802_1029277802:`
	`_NQ_LogicalDisk_DriveList = c:,d:`
	`_NQ_LogicalDisk_TH_FREE = 10`
	`_NQ_LogicalDisk_TH_READS = 50`
	`_NQ_LogicalDisk_TH_UTIL = 90`
	`_NQ_LogicalDisk_TH_WRITES = 50`
	`_NQ_LogicalDisk_TH_XFERS = 80`
	`_NQ_Service_ExcludeList =`
	`_NQ_Service_ServiceList = EventLog`

exit	Exits NetIQctrl. The basic syntax is: `exit`

Command	Description

healthcheck	Verifies that the job information stored in the AppManager repository is in sync with the agent's job information. Under normal conditions, the agent sends its job list to the management server periodically and the management server compares this list to the job list stored in the repository to ensure they are the same. This command allows you to check this activity.
	This command provides the following information:
	• The job list retrieved from the repository, the job list retrieved from the agent and the time the management server last performed the comparison.
	• The time of the last job check that the management server sent to the agent for verification of the job list if the management serve didn't receive the job list from the agent.
	• The number of jobs and the job IDs of jobs waiting in the cache. Pending jobs are stored temporarily in the management server cache only if they are detected when the management server retrieves agent information from the repository.
	This command can be useful if you are trying to determine whether changes to job status or the job list are being handled properly for an agent computer. For example, if a job appears to remain in a Pending state or has been stopped but still appears to be running in the Operator Console, you can use this command to see the time of the last status check by the management server and the list of active jobs from the repository and the agent.
	The basic syntax is:
	`healthcheck` *`ms_hostname`* `NetIQms mc_hostname`
	Output example
	`---------------------------`
	` Health Check Info`
	`---------------------------`
	`Last Health Check Time:  1029354749`
	`Job List from Repository:  12 18 20 22`
	`Job List from Agent:  22 18 20 12`
	
	`Last Job Check Time:  1029269190`
	`Job List to Agent:  22`
	
	`Number of Jobs in Cache:  0`
	`Job List in Cache:  None`

help	Displays the list of commands and usage information for NetIQctrl. The basic syntax is: `help` You can also use a question mark (?) to display help.

Command	Description

info	Displays a list of symbol-to-number mappings that can be used with the debug command to set the category and level of tracing information returned from a Knowledge Script while it is running. The basic syntax is:
	`info`
	Output example
	`MC_CORE, 65`
	`MC_RPC, 66`
	`MC_VBA, 67`
	`MC_CALLBACK, 68`
	`MC_REPOSITORY, 69`
	`MS_MS, 85`
	`MS_MC, 86`
	`MS_RP, 87`
	`-------`
	`LEVEL_VERBOSE, 65535`
	`LEVEL_TRACE, 256`
	`LEVEL_CRITICAL, 64`
	
	When specifying the debug category or level, you can use either the symbol (such as `MC_CORE` or `LEVEL_VERBOSE`) or the number mapping (such as `65` or `65535`).
	For example, to start tracing for the agent computer named `shasta`, you can use either:
	`debug shasta netiqmc MC_VBA LEVEL_TRACE`
	or
	`debug shasta netiqmc 67 256`
	To control where tracing output is sent, use the `output` command.

Command	Description

infoconfig	Displays configuration information for a management server. Some configuration parameters, such as polling intervals, control the behavior of a management server and can be used to fine-tune its performance.
	The basic syntax is:
	`infoconfig` *`ms_hostname`* `NetIQms`
	For example, to list the configuration for the management server `olympus`, use:
	`infoconfig olympus netiqms`
	Output example

```
-----------------------------------------------------
  MS Configuration Info
-----------------------------------------------------
** connectivity
    MS Port             :    9999
    MC Port             :    9998
    rpc comm wait       :    5000
    ccm flow ctrl       :    enabled
** execution
    mode                :    service
    start time          :    Tue Aug 13 10:04:53 2002
** repository
    server              :    OLYMPUS
    site name           :    OLYMPUS1028146695
    repository name     :    QDB
    ms UUID             :    c6ebacea-50d9-4e73-8e0f-f7654c713
    dsn                 :    QDBms
    user name           :    netiq
    machine id          :    34
    machine name        :    OLYMPUS
    wait on rp failure  :    300
    rp retry            :    0
** ioc threads
    data worker         :    2
    event worker        :    1
** thread sleep intervals
    job poll            :    5
    job stat            :    300
    job stat ip refresh :    1000
    unix machine check  :    300
    unix machine timeout:    600
    machine poll        :    900
    machine ping        :    5
    all machines ping   :    0
** event config
    record type         :    create open events (default)
    cache list max      :    10
    collapsed ioc retry :    1
** data config
    data batching       :    on
    data retry          :    1
```

Command	Description

job	Displays a summary of the jobs on an agent computer. For more detailed job information, use the `profile` command.
	
	The `job` command is useful for verifying the state of a job. For example, if a job appears in Pending state in the Operator Console, you can use the `job` command to determine whether the job has started on the agent computer. If the job has started on the agent computer but the Operator Console shows it as pending, you would want to investigate the connections between the agent computer, the management server, and the repository. The basic syntax is:
	
	`job` *`mc_hostname`* `NetIQmc [option]`
	
	Use the optional parameter to specify the type of job you want to display:
	
	• `run` for jobs that are currently running, including jobs that are in an active, inactive, and scheduled state.
	• `done` for jobs that stopped recently.
	• `active` for jobs that are currently running and active.
	• `inactive` for jobs that are running but are inactive, for example because of a scheduled maintenance period.
	• `sched` for jobs that are waiting for the start of their next scheduled running period.
	
	You can check the status of a specific job by specifying the management server name that started the job and the job ID.
	
	To see a list of all jobs on the agent computer named `paris`, use:
	
	`job paris NetIQmc`
	
	Output example
	
	`----------------------`
	`Running Jobs`
	`----------------------`
	`22_AJAX1028146695_1028146695 : thread=2012 15 <worker  running>`
	`<sleeping>`
	`18_AJAX1028146695_1028146695 : thread=2144 15 <worker running>`
	`<sleeping>`
	`20_AJAX1028146695_1028146695 : thread=768  3 <worker running> <sleeping>`
	`12_AJAX1028146695_1028146695 : thread=1652 14 <worker running>`
	`<sleeping>`
	`------------------------------`
	`Stopped/Completed Jobs`
	`------------------------------`
	`40_AJAX1028146695_1028146695 : Wed Aug 14 11:52:23 2002`
	`42_AJAX1028146695_1028146695 : Wed Aug 14 11:56:30 2002`
	`+++++++  total 2 completed jobs`
	`+++++++  total 4 running jobs`

Command	Description

jobevt	Displays event summary information for all jobs or for a specific job on an agent computer. The `jobevt` command is useful for verifying the number of events generated by a specific job. For example, you might compare the number of events reported by the `jobevt` command with the number of events displayed in the Operator Console for that job. If the numbers don't match, you might want to investigate the connections between the agent computer, the management server, and the repository.
	For detailed event collapsing information, use the `profevt` command.
	The basic syntax is:
	`jobevt` *`mc_hostname`* `NetIQmc [`*`ms_hostname job_id`*`]`
	The optional parameters let you specify a job ID. To identify a specific job, you must also provide the name of the management server that started the job.
	For example, to see the event collapsing information for job number `20`, which is running on the agent computer named `shasta` and was scheduled by the management server `ajax`, use:
	`jobevt shasta netiqmc ajax 20`
	Output example
	`----------------------`
	`Running Jobs`
	`----------------------`
	`Job [20]  =>`
	`  KS Name          :  1702:NT_LogicalDiskBusy`
	`  QDB Site         :  AJAX1028146695_1028146695`
	`  QDB Site UpdTime :  1028146695`
	`  MS machine       :  AJAX <10.5.10.92>`
	`  Collapse Intv    :  1200 sec`
	`  Occur Interval   :  1`
	`[1029258286_2]  inst 5  non_collapse 5  collapse 0 (cur=0)`
	`Job [12]  =>`
	`  KS Name          :  1702:General_ServiceDown`
	`  QDB Site         :  AJAX1028146695_1028146695`
	`  QDB Site UpdTime :  1028146695`
	`  MS machine       :  AJAX <10.5.10.92>`
	`  Collapse Intv    :  1200 sec`
	`  Occur Interval   :  1`
	`[1029258286_5]  inst 1  non_collapse 1  collapse 24 (cur=24)`
	`+++++++  total 2 running jobs`

jobmod	Indicates the time of the last modification to a job's properties. The modification time is displayed in UTC format. The basic syntax is:
	`jobmod` *`ms_hostname`* `NetIQms` *`job_id`*
	Output example
	`--------------------------`
	`Job Modification Info`
	`--------------------------`
	`job id:              4`
	`last mod time (utc): 1029432427`
	`last status:         513`

Command	Description

jobrsc	Displays status information for jobs that are inactive on an agent computer during a maintenance period. Maintenance periods are set by Knowledge Script category. The `jobrsc` command includes standard job information, a resource ID for each Knowledge Script category running on the agent computer, the number of job iterations skipped because of maintenance, and the last time an iteration was skipped.
	
	The `jobrsc` command is useful for determining how many times a job has not run because of a maintenance period. For example, you may be trying to discover why a job has not generated the expected number of data points. You can use the `jobrsc` command to determine how many times the job did not run, which could explain why fewer data points were collected.
	
	The basic syntax is:
	
	`jobrsc` *`mc_hostname`* `NetIQmc [`*`ms_hostname job_id`*`]`
	
	Optional parameters let you specify a job ID. To identify a specific job, provide the name of the management server that scheduled it.
	
	For example, to display information about skipped iterations for all jobs on the agent computer named `shasta`:
	
	`jobrsc shasta NetIQmc`
	
	Output example
	
	`----------------------`
	`Running Jobs`
	`----------------------`
	`Job [62]  =>`
	`KS Name : NT_CpuResource`
	`QDB Site : OLYMPUS904955486_904955487`
	`QDB Site UpdTime : 904955487`
	`MS machine : OLYMPUS <10.1.10.65>`
	`Job Status : worker running`
	`RSC ID : 0`
	`# Skipped Iter : 0`
	`Iter Skip Time :`
	`Job [66]  =>`
	`KS Name : SQL_DataSpace`
	`QDB Site : OLYMPUS904955486_904955487`
	`QDB Site UpdTime : 904955487`
	`MS machine : OLYMPUS <10.1.10.65>`
	`Job Status : Inactive`
	`RSC ID : 1`
	`# Skipped Iter : 18`
	`Iter Skip Time : Tue Sep 10 14:46:40 2002`
	`+++++++  total 2 running jobs`

jobsched	Displays scheduling information for a specific job. Basic syntax:
	
	`jobsched` *`mc_hostname`* `NetIQmc` *`ms_hostname job_id`*
	
	For example, to see scheduling information for job ID 68, which is running on the agent computer named `shasta` and was scheduled by the management server named `olympus`, use:
	
	`jobsched shasta netiqmc olympus 68`
	
	Output example
	
	`-------------------------`
	`Job Scheduling Info`
	`-------------------------`
	`Job [68__OLYMPUS904955486_904955487]  ==>`
	`Type: on demand`
	`Recur Type: INTERVAL ITERATION`
	`Interval: 5 min`

Command	Description

jobstat	Displays statistics for a specific job running on an agent computer. For example, `jobstat` displays the number of events, data headers, data points, and agent computer actions generated by the job. The basic syntax is:
	`jobstat` *`mc_hostname`* `NetIQmc` *`ms_hostname job_id`*
	For example, to display information for job number `62`, running on the agent computer `shasta` and was scheduled by the management server `olympus`, use:
	`jobstat shasta netiqmc olympus 62`
	Output example
	`Job [62_OLYMPUS904955486_904955487]  =>`
	
	`            #-Generated  #-Failed`
	`----------  -----------  ----------`
	`Event       4            0`
	`CtrlEvt     1344         0`
	`DataHead    1            0`
	`DataLog     22           0`
	`MCAction    0            0`

ks	Displays the Knowledge Script of a specified job that is currently running on an agent computer. The basic syntax is: `ks` *`mc_hostname`* `NetIQmc` *`ms_hostname job_id`* **Note** This command is not supported for version 4.0 and later Knowledge Scripts.

listfc	Displays the current network configuration and flow control information between an agent computer and the management server that it is communicating with. The basic syntax is:
	`listfc` *`mc_hostname`* `NetIQccm site|ms` *`ms_hostname`*
	• Use the `site` keyword to extract information from the Client Communication Manager using the repository name.
	• Use the `ms` keyword to extract information using the management server hostname.
	The information returned should be the same, whether you use the `site` or `ms` keyword.
	For example, to see network configuration and flow control information between the agent computer named `shasta` and the management server named `olympus`, use:
	`listfc shasta netiqccm site olympus`
	Output example
	`-------------------------------------------`
	`      List CCM Current Site Network Info`
	`-------------------------------------------`
	`   netsrv thread id   2460`
	
	`   MS hostname                OLYMPUS <10.5.111.139>`
	`   MS network status          up`
	`   Communication mode         uploadInProcess`
	`   Current high watermark     100 kb`
	`   Current low watmermark     0 kb`
	`   Current batching interval  1000 sec`
	`   Last net transaction Time  Thu Aug 15 12:23:20 2002`

Command	Description

listms	Displays a list of all management servers that are communicating with a specified agent computer. The list indicates the number of running jobs that were started from each management server and the state of the connection between the agent computer and each management server. The basic syntax is:
	`listms` *`mc_hostname`* `NetIQmc`
	For example, to see information about the management servers that communicate with the agent computer named `paris`, use:
	`listms paris netiqmc`
	Output example
	`----------------------------------------------`
	`List MS`
	`----------------------------------------------`
	`OLYMPUS1029277802_1029277802`
	`Current MS => 10.5.11.39 OLYMPUS`
	`Job Refcnt => 3`
	`Use MS IP => 1`
	`MS status => Up`
	`Send via => ccm`
	`AJAX1029363296_1029363296`
	`Current MS => 10.5.10.92 AJAX`
	`Job Refcnt => 2`
	`Use MS IP => 1`
	`MS status => Down`
	`Send via => ccm`
	This example indicates that the agent computer `paris` communicates with two management servers, `olympus` and `ajax`. Three jobs were submitted from the management server `olympus` and two jobs were submitted by the management server `ajax`.
	Because communication between the agent computer and the management server `ajax` is down, data and events for the two jobs submitted by `ajax` are being stored locally on the agent computer. Data and events from the three jobs submitted by `olympus` are being uploaded via the Client Communication Manager because communication between that management server and agent computer is working correctly.
	The `Use MS IP` parameter indicates whether the communication between the agent computer and management server is established using the management server's IP address or hostname. A value of `1` indicates the communication uses the management server's IP address. A value of 0 indicates communication is established using the hostname.

Command	Description

listrsc	Displays a list of all categories of jobs running on a agent computer and indicates which categories are affected by scheduled maintenance periods. Each job category is identified by a resource ID. The resource ID is specific to each agent computer. For example, if two SQL Knowledge Scripts are running on an agent computer, both jobs will have the same resource ID. However, that resource ID for SQL Server jobs on that agent computer will not necessarily match the resource ID for SQL Server jobs on other agent computers.
	For each category of Knowledge Script affected by a scheduled maintenance, the `jobrsc` command provides additional detail about the maintenance period.
	The basic syntax is:
	`listrsc` *`mc_hostname`* `NetIQmc`
	For example, to display a list of all categories of Knowledge Scripts running on the agent computer named `shasta`:
	`listrsc shasta netiqmc`
	Output example
	`----------------------`
	`RSC Info`
	`-----------------------`
	`<RSC #0> :`
	`Name => NT`
	`JobCnt => 2`
	`Status => NotScheduled`
	`ResDepend =>`
	`SvcDepend =>`
	`<RSC #1> :`
	`Name => SQL`
	`JobCnt => 1`
	`Status => Inactive`
	`ResDepend =>`
	`SvcDepend =>`
	`#-KPC-#`
	`Type: scheduled`
	`Recur Type: DAILY`
	`Recur Freq: every <1> day(s)`
	`Start Time: 14:00:00`
	`End Time: 17:59:00`
	`Interval: 5 min`

Command	Description

listsite	Displays a list of all QDBs monitoring an agent computer. The basic syntax is:
	`listsite` *`mc_hostname`* `NetIQccm`
	For example, to list all the QDBs that are monitoring the agent computer named `paris`, use:
	`listsite paris netiqccm`
	Output example
	`----------`
	`List CCM Site Info`
	`----------`
	`[SITE #1]`
	`QDB Name VENICE1029277802_1029277802`
	`QDB Key 32c368bf-36a0-401f-b64c-5ee5547cc689`
	`QDB Updtime 1029277802`
	`MS hostname VENICE <10.5.11.52>`
	`CCM comm id 3`
	`Server thread id 1060`
	`Startup Time Wed Aug 14 14:19:07 2002`
	`Reference Count 1`
	`[SITE #2]`
	`QDB Name LONDON1028151606_1028151607`
	`QDB Key ae215a98-dbdb-4d9f-a13e-22b8a33a083d`
	`QDB Updtime 1028151607`
	`MS hostname LONDON <10.5.18.42>`
	`CCM comm id 2`
	`Server thread id 1560`
	`Startup Time Fri Aug 02 11:30:52 2002`
	`Reference Count 1`
	`[SITE #3]`
	`QDB Name MILAN1027112432_1027112433`
	`QDB Key dfdb0224-fef3-4e51-b05d-5454ae4034f6`
	`QDB Updtime 1027112433`
	`MS hostname MILAN <10.5.11.61>`
	`CCM comm id 1`
	`Server thread id 1012`
	`Startup Time Thu Jul 25 17:09:28 2002`
	`Reference Count 1`

Command	Description

listupload	Lists the scheduled upload status that a QDB has registered with the Client Communication Manager service on an agent computer. This command lists the upload status of both events and data. The basic syntax is:
	`listupload` *`mc_hostname`* `NetIQccm site|ms` *`ms_hostname`*
	• Use the `site` keyword to extract information using the repository name.
	• Use the `ms` keyword to extract information using the management server hostname.
	The information returned should be the same, whether you use the `site` or `ms` keyword.
	For example, to list upload information for the agent computer named `paris` and the management server on the computer named `olympus`, use:
	`listupload paris netiqccm site olympus`
	Output example
	`----------`
	`List CCM Site Upload Configuration`
	`----------`
	`QDB Site Name OLYMPUS1029277802_1029277802`
	`QDB Site Updtime 1029277802`
	`MS hostname OLYMPUS <10.5.11.39>`
	`Event Upload active`
	`Data Upload active`

Command	Description

machine	Displays computer- or monitoring-related information that a management server has collected for all agent computers or a specified agent computer. The information displayed includes:
	• Remote Procedure Call communication version and the time the agent computer was last pinged.
	• The time of the last job status check between the management server and the agent computer, the number of start and stop job requests submitted, the current outstanding job (if any), the time the current job was submitted, and the number of jobs that may have been skipped because of this job.
	• Time zone information for the agent computer.
	The basic syntax is:
	`machine` *`ms_hostname`* `NetIQms [`*`mc_hostname`*`]`
	If you do not specify a specific agent computer, the `machine` command displays cached information for all agent computers that the management server is currently aware of. Because this operation is more expensive in terms of system resources, it is usually better to specify a specific agent computer.
	For example, to display the information that the management server named `olympus` has cached at the agent computer named `paris`, use:
	`machine olympus netiqms paris`
	Output example:
	`----------`
	`MS Cached Machine Info`
	`----------`
	`PARIS:`
	`** connectivity`
	`rpc comm version : 2.4.0`
	`last ping success : Thu Aug 15 11:47:39 2002`
	`last ping fail : Tue Aug 13 15:47:31 2002`
	`** jobs`
	`last job status success : Tue Aug 13 16:02:31 2002`
	`last job status fail : none`
	`start requests submitted : 9`
	`stop requests submitted : 1`
	`current job request : 0`
	`time job submitted : Wed Aug 14 15:41:03 2002`
	`jobs skipped : 0`
	`** time zone`
	`name : Pacific Standard Time`
	`daylight savings name : Pacific Daylight Time`
	`bias : 28800`
	`active bias : 25200`
	`daylight savings : applicable`

Command	Description

netconf	Lists the user-specified network configuration that a specific QDB or management server has registered with the Client Communication Manager (`NetIQccm`) service on an agent computer. The information includes the flow control configuration and the communication type.
	The basic syntax is:
	`netconf` *`mc_hostname`* `NetIQccm site|ms` *`ms_hostname`*
	• Use the `site` keyword to extract information by specifying the repository name.
	• Use the `ms` keyword to extract information by specifying the management server hostname.
	For example, to display the network configuration information that the management server named `olympus` has registered with the agent computer named `paris`, use:
	`netconf paris netiqccm site olympus`
	Output example:
	`----------`
	`List CCM Site Network Configuration`
	`----------`
	`MS hostname OLYMPUS <10.5.11.39>`
	`RPC connection caching disabled`
	`Communication type ipaddr`
	`Communication encryption Off`
	`Communication mode uploadScheduled`
	`Net high watermark 100 kb`
	`Net low watermark 0 kb`
	`Net batching interval 1000 sec`
	`Dynamic Flow Control disabled`

output	Specifies where to send the debug tracing output collected from a agent computer or management server. The basic syntax is:
	`output mc_hostname NetIQmc [option]`
	`output ms_hostname NetIQmc [option]`
	Optional parameters turn tracing off or direct output:
	• `off` turns off tracing output.
	• `here` displays the output in the NetIQctrl window.
	• `console` displays the output in the console window.
	• `log` displays the output in the Windows event log.
	• `file file_name` stores the output in a file.
	For example, to direct tracing for the AppManager agent on the agent computer named `shasta` to the file `trace.txt`, use:
	`output shasta NetIQmc file trace.txt`
	To view tracing for the AppManager agent on the agent computer named `shasta` in the NetIQctrl window, use:
	`output shasta NetIQmc here`
	Output example:
	`NetIQctrl> 10.1.1.163: [484] MCRunJob: finished processing job <45_SHASTA880588204>, ms=<10.1.1.163>`
	Note The debugging output will be intermixed with your commands.

Command	Description

ping	Checks connectivity and displays version and configuration information for an agent computer or a management server. If a connection is made, the `ping` command displays the version number and startup information for the AppManager service running on the specified computer.
	The basic syntax is:
	`ping` *`mc_hostname`* `NetIQmc [ext]`
	`ping` *`ms_hostname`* `NetIQms`
	The optional `ext` keyword displays extended configuration for the agent computer. The additional information indicates whether the agent is running in autonomous mode, whether data persistence is enabled, the communication protocol for the local repository and for the management server, and how tracing is configured.
	For example, to check whether the AppManager agent is running on the agent computer named `shasta`, use:
	`ping shasta NetIQmc`
	Output example:
	`version 4.6.31.0`
	`start_mode Service`
	`start_time Wed Aug 14 15:25:16 2002`
	`trace_level 1`
	`rpc version 4.0`
	`rpc port # 9998`
	If the AppManager agent isn't running on `shasta`, you see:
	`Failed to connect to mc`
	Note Generally, the AppManager agent uses port 9998, and the management server uses port 9999.

probe	Sends a probe request to the computer where the management server service (`NetIQms`) or agent service (`NetIQmc`) is located. If the service is running, the issue, arrival, and return times are displayed. The basic syntax is:
	`probe` *`mc_hostname`* `NetIQmc`
	`probe` *`ms_hostname`* `NetIQms`
	For example, to test the communication between the computer `paris` and the computer where the AppManager agent is running, use:
	`probe paris NetIQmc`
	Output example:
	`1029437586 - ctrl`
	`1029437586 - paris <NetIQmc>`
	`1029437586 - ctrl`
	Note Inconsistent arrival times are generally caused by different internal clock settings on the two computers.

Command	Description

profevt	Displays detailed event collapsing information for all jobs or a specific job on an agent computer. For a summary of event collapsing information, use the `jobevt` command. The basic syntax is:
	`profevt` *`mc_hostname`* `NetIQmc [`*`ms_hostname job_id`*`]`
	For example, to see detailed event collapsing information for job `6` on the agent computer `paris` and scheduled by the management server named `olympus`, use:
	`profevt paris netiqmc olympus 6`
	Output example:
	`Job [6] =>`
	`KS Name : NT_CpuResource`
	`QDB Site : OLYMPUS1029363296_1029363296`
	`QDB Site UpdTime : 1029363296`
	`MS machine : OLYMPUS <10.5.10.92>`
	`Collapse Intv : 1200 sec`
	`Occur Interval : 1`
	`---`
	`Event Id : 1029363916_9`
	`Severity : 5`
	`Object list :`
	`Event Message : Number of Processes High`
	`Occurrence Count : 1`
	`CurOccur Count : 0`
	`Collapse Intv : 1200 sec`
	`Total Collapse : 3`
	`Total NonCollap : 1`
	`Current Collapse : 3`
	`First Occurrence : Thu Aug 15 14:49:34 2002`
	`Last Occurrence : Thu Aug 15 14:52:33 2002`
	`---`
	`Event Id : 1029363916_10`
	`Severity : 5`
	`Object list :`
	`Event Message : Number of Threads High`
	`Occurrence Count : 1`
	`CurOccur Count : 0`
	`Collapse Intv : 1200 sec`
	`Total Collapse : 3`
	`Total NonCollap : 1`
	`Current Collapse : 3`
	`First Occurrence : Thu Aug 15 14:49:34 2002`
	`Last Occurrence : Thu Aug 15 14:52:34 2002`

Command	Description

profile	Displays detailed information about all jobs or a specific job running on an agent computer. For a summary of job information, use the `job` command. The basic syntax is:
	`profile` *`mc_hostname`* `NetIQmc [`*`option`*`]`
	The optional parameters specify the type of job for which you want to display information:
	• `run` for jobs that are currently running, including jobs that are in an active, inactive, and scheduled state.
	• `done` for jobs that are recently stopped.
	• `active` for jobs that are currently running and active.
	• `inactive` for jobs that are currently running but are inactive.
	• `sched` for jobs that are running and are waiting for the start of their next scheduled running period.
	For example, to display profile information for all jobs on the agent computer `paris`, use:
	`profile paris netiqmc`
	Output example:
	`----------------------`
	`Running-Active Jobs`
	`----------------------`
	`Job [4]  =>`
	`KS Name : NT_CpuLoaded`
	`QDB Site : OLYMPUS1029363296_1029363296`
	`QDB Site UpdTime : 1029363296`
	`MS machine : OLYMPUS <10.5.10.92>`
	`Thread ID : 2736 (0xab0)`
	`Local Action : <>`
	`Job Status : worker running`
	`Detailed State : <end one iteration>`
	`Submit Time : Thu Aug 15 10:27:06 2002`
	`Start Time : Thu Aug 15 10:27:06 2002`
	`StopPending Time :`
	`Iteration Intv : 300 sec`
	`Iteration Count : 18`
	`Iter Start Time : Thu Aug 15 11:52:07 2002`
	`Iter Stop Time : Thu Aug 15 11:52:08 2002`
	`Iter Run Time : 491 msec`
	`Avg Iter Time : 539 msec`
	`+++++++  total 1 job`

Command	Description

profrsc	Displays detailed status information about jobs that are currently inactive on an agent computer because of a maintenance period.
	For a summary of status information for inactive jobs, use the `jobrsc` command.
	Each job is associated with a resource ID that identifies the Knowledge Script category for that job. The resource ID is specific to each agent computer. For example, if two SQL Knowledge Scripts are running on an agent computer, both jobs will have the same resource ID. However, that resource ID for SQL Server jobs on that agent computer will not necessarily match the resource ID for SQL Server jobs on other agent computers.
	The basic syntax is:
	`profrsc` *`mc_hostname`* `netiqmc [`*`ms_hostname job_id`*`]`
	For example, to get detailed information about job number `66`, which is running on the agent computer named `polar` and was scheduled by the management server named `olympus`, use:
	`profrsc polar netiqmc olympus 66`
	Output example:
	`Job [66]  =>`
	`KS Name : SQL_DataSpace`
	`QDB Site : OLYMPUS904955486_904955487`
	`QDB Site UpdTime : 904955487`
	`MS machine : OLYMPUS <10.1.10.65>`
	`Job Status : Inactive`
	`RSC ID : 2`
	`# Skipped Iter : 22`
	`Iter Skip Time : Tue Sep 10 14:48:40 2002`

Command	Description

rpstat	Displays the statistics stored in the local repository of an agent computer. The list includes the number of events, data logs, and messages stored in the local repository. The messages may be waiting for a scheduled upload or they may be stored locally because the agent computer cannot communicate with the management server.
	Basic syntax:
	`rpstat mc_hostname NetIQccm site|ms ms_hostname`
	• Use the `site` keyword to extract information by specifying the repository name.
	• Use the `ms` keyword to extract information by specifying the management server hostname.
	For example, to list statistics for the agent computer named `paris`, which is managed by the management server named `olympus`, use:
	`rpstat paris netiqccm ms olympus`
	Output example:
	`----------`
	`List CCM Site Local RP Stat`
	`----------`
	`QDB Site Name OLYMPUST011029277802_1029277802`
	`QDB Site Updtime 1029277802`
	`MS hostname OLYMPUS <10.5.11.39>`
	`Type #-RP #-Cache LastDur AvgDur`
	`Event 0 0 0 0.00`
	`CtrlEvent 0 4 0 0.00`
	`DataHeader 0 0 0 0.00`
	`DataLog 0 0 0 0.00`
	`Exception 0 0 0 0.00`
	`JobComplete 0 1 0 0.00`
	• The RP column indicates the number of transactions held in the local repository.
	• The Cache column indicates the number of transactions held in memory.
	• The LastDur column indicates the duration of the last transaction with the local repository in seconds.

Command	Description

script	Saves NetIQctrl input and output in a text file. The basic syntax is:
	`script <filename>`
	To stop capturing input and output, use:
	`script done`
	For example:
	`NetIQctrl> script c:\temp\netiqctrl.log`
	`Script <c:\temp\netiqctrl.log> started`
	
	`NetIQctrl> ping mojo netiqmc`
	
	`version 2.0.261.5`
	`start_mode Service`
	`start_time Wed Nov 26 15:52:04 1997`
	`trace_level 1`
	`rpc version 2.0`
	`rpc port # 9998`
	
	`NetIQctrl> script done`
	`Script <c:\temp\netiqctrl.log> done`

site	Displays information about the repository that a specified management server is communicating with. The basic syntax is:
	`site` *`ms_hostname`* `NetIQms`
	For example, to see the repository information for the management server on the computer named `paris`, use:
	`site paris NetIQms`
	Output example:
	`site name PARIS880588204`
	`site time 880588206`

Command	Description

stat	Displays statistical information collected by a specific agent computer or management server. The output for this command varies depending on the service and options you specify in the command line. For example, if you run this command to display statistics for the Client Resource Monitor, `NetIQmc`, the output lists the number of events, data streams, actions, and jobs completed by the agent computer. By default, the statistics reflect the total for the agent computer regardless of which QDB or management server scheduled its jobs. The basic syntax is:
	`stat` *`mc_hostname`* `NetIQmc [option]`
	`stat` *`mc_hostname`* `NetIQccm [option]`
	`stat` *`ms_hostname`* `NetIQms`
	For `NetIQmc`, use the following keywords:
	• `site` displays both the total numbers for an agent computer, and a breakdown by site for the number of events, data streams, data points, and actions. You can also use the keyword `site` and a specific management server hostname to retrieve information associated with a specific AppManager repository.
	• `rpc` provides detailed information about RPC activity on the agent computer.
	• `sec` provides information about any agent computer activity that caused RPC requests to be rejected.
	• `jobpoll` indicates the last management server to which the agent computer sent a job status report.
	For `NetIQccm`, use the keyword `site` or `ms` and a management server hostname to retrieve information associated with a specific AppManager repository or management server.
	For `NetIQms`, this command provides detailed statistics covering thread activity and successful and failed communication.
	For example, to see summary information for the management client named `paris`, use:
	`stat paris NetIQmc`
	Output example:
	`----------`
	`List Summary Stat`
	`----------`
	`[Total] :`
	`Type #-Done #-Fail #-Skip`
	`---------- -------- -------- --------`
	`Event 7 0 0`
	`CtrlEvt 8 0 0`
	`DataHeader 2 0 0`
	`DataLog 19 0 0`
	`Exception 0 0 0`
	`JobComplete 7 0 0`

Command	Description

thread	Displays the status of all threads maintained by a management server. Basic syntax:
	`thread` *`ms_hostname`* `NetIQms [option]`
	Use the optional parameter to select what types of thread information you want to see:
	• `jobpoll` displays information for the job polling thread.
	• `jobstat` displays information for the job status check thread.
	• `machpoll` displays information for the machine polling thread.
	• `machping` displays information for the machine status check thread.
	If you don't specify an option, the command returns information for all management server threads.
	For example, to get detailed information about the job polling thread maintained by the management server named `olympus`, use:
	`thread olympus netiqms jobpoll`
	`----------`
	`Job Poll Thread Info`
	`----------`
	`** thread`
	`state : sleeping`
	`last completed : Thu Aug 15 11:56:35 2002`
	`last start time : Thu Aug 15 11:56:35 2002`
	`current machine : none`
	`sleep interval : 5`
	`** job polling`
	`last mod time : Thu Aug 15 11:49:32 2002`
	`run pending jobs : 0`
	`stop pending jobs: 0`
	`** run pending job submission`
	`jobs cached : 0`
	`jobs skipped : 0`
	`jobs submitted : 0`
	`** stop pending job submission`
	`jobs cached : 0`
	`jobs skipped : 0`
	`jobs submitted : 0`

trip	Sends a path request to the agent computer (`NetIQmc`) and a management server (`NetIQms`) to test the round-trip connectivity between them. The basic syntax is:
	`trip` *`mc_hostname`* `NetIQmc` *`ms_hostname`*
	For example, to test the round-trip connectivity between the agent computer named `shasta` and the management server named `paris`, use:
	`trip shasta NetIQmc paris`
	Output example:
	`1066766782 - ctrl`
	`1066766831 - mc <shasta>`
	`1066766782 - ms <paris>`
	`1066766831 - mc <shasta>`
	`1066766782 - ctrl`
	Note The response time for each computer is based on its internal clock settings. If the two computers have different clock settings, the UTC timestamps may appear to be incorrect. As long as the round trip is successful, however, you can verify the connectivity between the computers.

Command	Description

uploadsched	Lists the user-specified upload schedule that has been registered on an agent computer for a specific QDB. The output includes the site identifier for the repository and the separate schedules for uploading events and uploading data. The basic syntax is:
	`uploadsched mc_hostname NetIQmc [ms_hostname]`
	For example, to list the upload schedule that is registered on the agent computer named `shasta`, use:
	`uploadsched shasta netiqmc`
	Output example:
	`[PARIS1029363296_1029363296]`
	`#-EventUpload-#`
	`#-DataUpload-#`
	`Type: scheduled`
	`Recur Type: DAILY`
	`Recur Freq: every <1> day(s)`
	`Start Time: 13:00:00`
	`End Time: 02:00:00`
	`Interval: 1 hour`

unixjob	Displays a summary of the jobs on UNIX and Linux agent computers. The output includes a list of the job IDs for the selected computer, the management site ID, and the status of each job.
	The basic syntax is:
	`UnixJob ms_hostname NetIQms [UNIX_agent_IP]`
	Use the optional *UNIX_agent_IP* parameter to indicate a specific UNIX or Linux computer for which you want to list jobs. To display the job summary for a specific UNIX or Linux computer, include the computer's IP address in the command line (you must use the IP address and not the hostname). For example:
	`UnixJob rainier netiqms 64.220.132.10`
	`Unix Agent: 64.220.132.10`
	`Job: 26 Unknown`
	`Job: 28 Unknown`

Command	Description

unixmachine	Displays configuration and monitoring information for UNIX and Linux agent computers. The output for this command includes:
	• Agent version, platform, start time, and time zone
	• Queue configuration for the agent's data, event, job, and exception queues
	• Jobs that are running, pending, and pending a restart for the agent
	The basic syntax is:
	`UnixMachine` *`ms_hostname`* `NetIQms [`*`UNIX_agent_IP`*`]`
	Use the optional *`UNIX_agent_IP`* parameter to indicate a specific UNIX or Linux computer for which you want to list information. To display information for a specific UNIX or LInux computer, include the computer's IP address in the command line (you must use the IP address and not the hostname).
	For example:
	`unixmachine rainier netiqms 64.220.132.10`
	
	`Unix Agent: 64.220.132.10 (obj id: 149)`
	`Version: 2.0.137.0`
	`Platform OS: SunOS/sparc/5.8`
	`Startup Time: 1028323718`
	`Time Zone Bias: -480+1`
	`Socket Port: 9001`
	`# of Running Jobs: 2`
	`# of Pending Jobs: 0`
	`# of Pending Restart Jobs: 0`
	`Communication Queue Configuration:`
	`CurBrk/MaxBrk/MinBrk/Adj%/MaxSize/IncVal/MaxBlks>`
	`Data 1000/1000/10000/20/65536/10000/20`
	`Event 1000/1000/10000/20/65536/5000/20`
	`JobStat 1000/1000/10000/20/65536/0/20`
	`Exception 1000/1000/10000/20/65536/0/20`

Command	Description

unixmms	Displays the primary and secondary management servers for a UNIX or Linux agent computer. The output for this command displays the primary and secondary management server hostname for a specified UNIX agent or all UNIX agents.
	The basic syntax is:
	`UnixMMS ms_hostname NetIQms [UNIX_agent_IP]`
	Use the optional *UNIX_agent_IP* parameter to indicate a specific UNIX computer for which you want to list information. To display communication details for a specific UNIX computer, include its IP address in the command line (not the UNIX hostname). For example:
	`unixmms rainier netiqms 64.220.132.10`
	`Unix Agent: 64.220.132.10 (obj id: 149)`
	`Primary MS: RAINIER`
	`Secondary MS: SANDMAN`

unixtime	Displays information about the communication between the management server and UNIX and Linux agent computers. The output for this command includes:
	• The time the last heartbeat was received by the management server
	• The time the agent last requested data, events, jobs, and exception information
	• The time the management server last received the agent requests for data, events, jobs, or exceptions
	• The time of the last request rejected by the agent and the management server.
	The basic syntax is:
	`UnixTime ms_hostname NetIQms [UNIX_agent_IP]`
	Use the optional *UNIX_agent_IP* parameter to indicate a specific UNIX computer for which you want to list information. To display communication details for a specific UNIX or Linux computer, include the computer's IP address in the command line (you must use the IP address and not the hostname). For example:
	`unixtime rainier netiqms 64.220.132.10`
	`Unix Agent: 64.220.132.10 (obj id: 149)`
	`Last MSU Heartbeat Received Time: 1028324108`
	`Last Agent Request Reject Time: 0`
	`Last MSU Request Reject Time: 0`
	`Last Agent Data Request Time: 0`
	`Last MSU Data Request Received Time: 0`
	`Last Agent Event Request Time: 1028323797`
	`Last MSU Event Request Received Time: 1028323815`
	`Last Agent Job Status Request Time: 1028323878`
	`Last MSU Job Status Request Received Time: 1028323896`
	`Last Agent Exception Request Time: 1028323722`
	`Last MSU Exception Request Received Time: 0`
	The `UnixTime` command is useful for troubleshooting communication or network problems between the management server and UNIX agent. With this command, you can trace the heartbeat and task requests from the UNIX agent and compare them to the requests received by the management server to determine whether there are connection or communication problems.

Using the NetIQ Diagnostics Utility

NetIQ Diagnostics is a utility found in the `bin` folder for your AppManager installation (for example, in a default location such as `C:\Program Files\NetIQ\AppManager\bin`). NetIQ Diagnostics is used to collect information from the agent computer log files and from the registry. Although the information may help you to troubleshoot your environment on your own, typically this information is sent directly to NetIQ Corporation Solutions Support to help them analyze and diagnose problems in your deployment of AppManager components.

The NetIQ Diagnostics utility must be run locally on the computer you are attempting to diagnose.

To run the NetIQ Diagnostics utility:

1. Double-click the **NetIQDiag.exe** program in the `NetIQ\AppManager\bin` folder.
2. In the first page of the Diagnostics wizard:
   - Verify that **Set maximum trace level** is selected.
   - Check both **AppManager Agent** and **AppManager Management Server** unless instructed otherwise by NetIQ Technical Support.
   - Click **Set** to set the tracing levels for the agent and management server to the maximum level, then click **Next**.
3. Check the specific components you want to diagnose from the AppManager Component Options list.
4. Click **Diagnose**. The NetIQ Corporation Diagnostics utility begins collecting information about your environment. When the diagnosis is complete, click **Next**.
5. If diagnosing an Analysis Center report repository, type a SQL Server login account and password for a SQL Server account with permissions associated with the System Administrators role, and, if appropriate, check **Use NT Authentication**.
6. If diagnosing an AppManager repository, type a user name and password that has permission to access the AppManager database, if appropriate, check **Use NT Authentication**, then click **Next**.
7. Check any external sources of information you want included with the diagnostic package, then click **Next**. For example, if you are seeing unusual Windows events or Dr. Watson errors, you may want to include those log files.
8. Click **Next** to bypass the AppAnalyzer and XMP diagnostic pages.
9. Click compress to collect all of the log files and other information into a CAB file in the `NetIQ\diagnostics` directory, for example, `C:\Program Files\NetIQ\diagnostics`.

   The name of the diagnostic file is:

   `computer name_MM.DD.YY_HH.MM.SS``_diag.cab`

   For example: `Detroit_08.16.02_15.09.20_diag.cab`

Viewing NetIQ Diagnostics Output

All log files that NetIQ Diagnostics generates are collected in a compressed CAB file each time you run the Diagnostics utility. To view any of the log files, you first need to extract them from the compressed file. You can then view the information from the text-based logs in any text editor.

Enabling Tracing and Viewing Log Files

Most AppManager components include registry keys for fine-tuning trace logging. By default, most components are set to do little or no logging for performance reasons. If you are troubleshooting your AppManager environment, however, it may be useful to change the tracing level to provide more detailed information in log files.

Note
NetIQ Corporation recommends you use the Diagnostics utility to set the tracing level rather than edit registry keys directly because using the Registry Editor incorrectly can cause serious problems that may require you to reinstall your operating system. In general, you should only edit the registry directly if instructed to do so by NetIQ Technical Support or if you have a thorough understanding of the key values and the implications of making changes. NetIQ cannot guarantee that problems resulting from the incorrect use of the Registry Editor can be resolved.

All AppManager log files for the components installed on a computer are stored in the location defined in the `HKEY_LOCAL_MACHINE` registry under `\Software\NetIQ\Generic\Tracing`. For example, the default path is typically:

```
TraceLogPath: c:\program files\netiq\Temp\NetIQ_Debug
```

For information about the specific registry keys that enable or configure logging for each component, see "Registry Keys" on page 221. The following table provides a summary of the information recorded in the log files and how you can use this information to troubleshoot your AppManager environment.

Log	Information Recorded
`ccmtrace.log`	If you enable `TraceCCM`, the NetIQ Corporation AppManager Client Communication Manager (`NetIQccm`) records information about its activity in the `ccmtrace.log` file. The information recorded in this file includes the connection status for communication with the management server, and the processing status for events and data stored in the local repository.
`mctrace.log`	If you enable `TraceMC`, the NetIQ Corporation AppManager Client Resource Monitor (`NetIQmc`) records information about its activity in the `mctrace.log` file. The information recorded in this file includes the status of polling threads, job requests, and job execution. If you also enable Knowledge Script tracing with `TraceKS`, this log also includes line-by-line trace entries for each job running on the agent. This allows you to step through the Knowledge Script as it is executed to locate the point of failure. **Note** Knowledge Script tracing creates an ASCII copy of the compiled Knowledge Script with debugging line numbers in the *`TraceLogPath`*`\mc` subdirectory. A separate file is created for each job and action executed on the agent and the name of each file includes the JobID and the SiteID.
`mo.log`	If you enable tracing for any managed object, the `mo.log` records information about monitoring activity for that managed object. The information recorded includes all function calls made from the managed object during job execution. By default all managed object tracing flags are set to `0x10`, which provides a minimal level of tracing. You can log additional detail by increasing the value for the relevant `TraceMO`*`component`* registry key. To enable full tracing for a desired MO, set the appropriate registry value to `0xFF`.

Log	Information Recorded
Ckcomponent.log	Pre-installation information for each managed object selected during a pre-installation check is recorded in a separate log file. For example, if you run the pre-installation check for Exchange 2000, the setup program records information about the checks performed in the CkExch2.log. These log files can help you determine why a particular managed object cannot be installed or discovered on a computer.
loc.log	The loc.log traces internal pipe communication between the NetIQ Client Resource Monitor and NetIQ Client Communication Manager services.
Nqioc_err.log	The Nqioc_err.log records information about errors generated by the IO Completion port on the agent. The IO Completion port is used to verify the delivery of information to the management server. If errors are generated on this port, it may suggest delivery failures.
Ms.log	The NetIQ Corporation AppManager Management Service (NetIQms) records information about all of its activity in the ms.log. For example, NetIQ Corporation AppManager Management Service writes information in the ms.log when it delivers jobs to agents, receives events and data, or detects changes to job or machine status.
msaction.log	If you enable Action Log tracing, the NetIQ Corporation AppManager Management Service (NetIQms) records detailed information about management server or proxy action processing errors.
msqdb.log	If you enable QDB Log tracing, the NetIQ Corporation AppManager Management Service (NetIQms) records detailed information about errors encountered when the management server connects to the repository.
Rplib.log	The rplib.log records information about the connection and activity between the NetIQ Corporation AppManager Management Service (NetIQms) and the repository or between the Operator Console and the repository. This file logs any ODBC errors encountered in the connection between the management server and the repository or between the Operator Console and the repository. The file also includes information about all requests to fetch, update, refresh, or delete information in the repository.
KSCheckin.log	The kscheckin.log records information about all attempts to check Knowledge Scripts into the repository, including attempts to check in scripts from a local Operator Console, using the Kscheckin.exe utility, or through an installation or upgrade of the repository.
QDBInstall.log	The QDBInstall.log stores a complete record of the repository installation process.
QDBUpgradenn.log	The QDBUpgrade*nn*.log stores a complete record of the repository upgrade process. The log filename indicates the version of AppManager to which you upgraded. For example, if you are upgrading to AppManager 6.7, the log filename is QDBUpgrade67.log.
KSCustomnn.log	The KSCustom.log stores information about Knowledge Scripts that have had properties customized and records whether the customized version of the Knowledge Script has been successfully checked into the QDB. The log filename indicates the version of AppManager to which you upgraded. For example, if you are upgrading to AppManager 6.7, the log filename is KSCustom67.log.
appmgr.log	The appmgr.log records debugging information for an AppManager agent installation. Unlike other log files, this file is located in the Windows system folder. For example: `%SYSTEMROOT%\appmgr.log`

Log	Information Recorded
`Maint.log`	The `maint.log` records information about any patches or hot-fixes installed for any AppManager component on a local computer.
`Msadapt.log`	The `msadapt.log` records information about all activity processed by an AppManager Connector.

Using the Log Analysis Tool

The Log Analysis Tool lets you parse the UNIX agent log files to consolidate information about default threads executed by the agent, and information about threads executed in conjunction with jobs.

During normal operation, the UNIX agent records information about its activity in log files. Because the UNIX agent makes entries in the log file at each execution of a thread, the various thread entries become interspersed with each other. The Log Analysis Tool helps you analyze the information recorded in the log file by consolidating entries by thread and extracting the consolidated information in a more readable format. The output from the Log Analysis Tool makes it easier to troubleshoot agent operation and identify any agent problems.

Note
For information about configuring logging for the UNIX agent, see the *AppManager for UNIX Servers Management Guide*.

The Log Analysis Tool is located in `$NQMAGT_HOME/bin/logparser/fileParse.sh`.

You can use the following arguments and options with the Log Analysis Tool:

Option	Description

`-v` *`level`*	Set the verbosity level. Valid levels are:
	`1 Displays only the start and end steps of each task thread (default).`
	`2 Displays start and end steps, as well as all intermediate steps of each task thread.`

`-a`	Display the average time for each iteration.
`-i` *`interval`*	Set the iteration interval to any whole number you specify for the *`interval`*. For example, if you specify 3, the Log Analysis Tool returns every 3rd iteration of a thread. If you do not specify an interval, the Log Analysis Tool returns information for all intervals.
`-x` *`type`*	Specify the type of information you want to exclude from the output. The valid types of output you can exclude are: • `threads` to exclude the default agent threads and display only job threads. • `jobs` to exclude job threads and display only the default agent threads. **Note** The default agent threads are the Heartbeat, Event Queue, Job Status Queue, Exception Queue, Job Sync, and Thread Monitoring threads. If you do not specify a type of output to exclude, the Log Analysis Tool returns information for all threads.

Option	Description

`-d` *`date`*	Specify a date range for the information returned. You can identify a specific date or a range of dates using the format:
	`mm/dd/yy`
	For example, if you want information for a specific date, you can enter that date:
	`-d 03/01/04`
	To specify a date range, enter the start and end dates. For example:
	`-d 03/01/04-03/05/04`
	If you do not specify a date or date range, the Log Analysis Tool returns information for all dates in the log file.

`-j` *`jobID`*	Identify a specific job for which you want to return information. You can separate multiple job IDs by using commas. For example, to specify you want information for Job IDs 1, 12, and 23: `-j 1,12,23` If you do not specify a job ID, the Log Analysis Tool returns information for all jobs.
`-q` *`siteID`*	Identify a specific QDB for which you want to return information. **Note** This option is not supported in this version of the Log Analysis Tool.
`-h or ?`	Display usage Help for the Log Analysis Tool.
`-f` *`logfile`*	Specify the filename for the log file to parse. You should use this option if you are running the Log Analysis Tool in the log file directory. Use `nqmlog` to parse the most recent log file. The `nqmlog` is a hard link to the most recent log file. To parse an earlier file, enter the exact filename. You can parse multiple files by entering the filenames, separated by commas. For example: `-f log20030412180531,log20030412180531` If you do not specify a filename, the Log Analysis Tool parses all UNIX agent log files.
`-l` *`path`*	Specify the path to the log file directory. You should use this option if you are running the Log Analysis Tool from a directory other than the log file directory. For example: `$NQMAGT_HOME/log` If you use this option *without* specifying the `-f` option, the Log Analysis Tool parses all log files in the directory. If you use this option *with* the `-f` option, the Log Analysis Tool parses only the files specified by the `-f` option.
`-e` *`messageoption`*	Indicate whether you want to write error messages to file or standard output. The valid *`messageoptions`* are: • `yes` to write error messages in the log file to an `Errors.txt` file in the current directory. • `no` to write error messages to standard output.

Appendix A
Additional Site Administration Utilities

This appendix provides reference information for using AppManager utilities that perform specialized tasks. All of these utilities are command-line programs. The following utilities are discussed:

- Key file utility for Windows agents
- Key file utility for UNIX agents
- MAPI mail utility
- Synchronization utilities
- Time conversion utility

Key File Utility for Windows Agents

The NetIQ Corporation key file generation program, `NQKeyGenWindows.exe`, is a command-line program used to set the security level for an AppManager management site and to generate and manage public/private encryption keys for secure communication between the management server and Windows managed computers. This utility is installed in the `NetIQ\AppManager\bin` folder.

The basic syntax for the `NQKeyGenWindows.exe` program is:

```
NQKeyGenWindows -option value
```

Note
Type `NQKeyGenWindows` without specifying any options to see usage information.

The program supports the following command-line options:

Option	Description

`-db`	Specifies the login information for connecting to the repository using the following format:
	`NQKeyGenWindows -db database_name:user_name:sql_server`
	For example:
	`NQKeyGenWindows -db qdb:smithj:nyc2003`
	If you are using Windows authentication to connect to the repository, leave the username blank. If you are using SQL Server authentication, type a SQL Server username for connecting to the repository. The program prompts for the password to use for the SQL Server account.
	Note Most of the other options require you to specify the connection information. If you use this option without specifying any additional options, the command displays the current security level setting.

`-new`	Creates a new record in the repository for the key information used to encrypt communication and authenticate the management server to the agents. For example:
	`NQKeyGenWindows -db db:user:sqlsvr -new`
	To create a new key file to share across multiple repositories on a computer other than the repository, use the command:
	`NQKeyGenWindows -new filelocation`
	This option creates a new key with password protection in the specified file location without checking it into the repository.
	Note When you use the `-new` option, you'll be prompted to provide a password for the key stored in the repository. You must specify a password to create the key.

`-change`	Changes the key information stored in the repository to use the new key file you specify. You must specify the key file password you used to create the key and the location of the key file to use. For example: `NQKeyGenWindows -db db:user:sqlsvr -change filelocation` This option enables you to check an existing key from a key file into a new repository when you want to share a key file across multiple repositories and management servers. **Note** When you use this option, you'll be prompted to provide the password you specified when you created the key.
`-ckey`	Extracts only the agent portion of the key stored in the repository. You must specify a location for the agent key file. For example: `NQKeyGenWindows -db db:user:sqlsvr -ckey filelocation` To extract the agent portion of the key, you must run `NQKeyGenWindows` on a management server. **Note** When you use this option, you'll be prompted to provide the password you specified when you created the key in the repository. Once you extract the agent portion of the key, you can copy the file and distribute it to the agents for encryption or authentication and encryption.
`-info`	Displays the current security level setting stored in repository. You are then prompted for the repository key password to display the checksum for verifying the encryption key and authentication key for an agent. For example: NQKeyGenWindows -db *db*:*user*:*sqlsvr* -info You can compare the checksum from the repository with the checksum returned by the `-agentinfo` option to verify whether an agent is using the correct key file for a specific repository. You can only use this option if you run `NQKeyGenWindows` on a management server computer.

Option	Description
`-skey`	Extracts the key information stored in the repository. You must specify a location for the key file. For example: `NQKeyGenWindows -db db:user:sqlsvr -skey filelocation` **Note** When you use this option, you'll be prompted to provide the password you specified when you created the key in the repository. This option checks out the current key into a password-protected file format. This file then can be checked into a different repository using the `-change` option. You can only use this option if you run `NQKeyGenWindows` on a management server computer.
`-seclev`	Sets the security level in the repository for communication between the management server and agents. Valid security levels are: • **0** for no security • **1** for encryption only security • **2** for authentication of the management server and encryption **Note** If you change the security level, the change takes effect when the management server is restarted. For example, to set the security level to use authentication of the management server: `NQKeyGenWindows -db db:user:sqlsvr -seclev 2`
`-agentchange`	Changes the agent key file for a managed computer to a key file you specify. The file location must be a local path. For example: `NQKeyGenWindows -agentchange filelocation` This option enables you to update the agent key file for a managed computer.
`-agentinfo`	Displays the checksum for verifying the encryption key and authentication key for an agent. For example: `NQKeyGenWindows -agentinfo` This option is useful for comparing the key information stored in the repository with the agent key information recorded for a managed computer to verify whether the correct key is being used.
`-agentseclev`	Sets the security level in the managed computer registry for communication between the management server and the agent. The valid security levels are: • **0** for no security • **1** for encryption only security • **2** for authentication of the management server and encryption For example, to set the security level to use authentication of the management server: `NQKeyGenWindows -agentseclev 2`
`-remoteseclev`	Sets the security level for a remote managed computer registry. You must specify the hostname of the remote computer for which you want to set a security level. For example to set the security to authentication and encryption for the remote computer `AJAX`: `NQKeyGenWindows -remoteseclev ajax 2` The valid security levels are: • **0** for no security • **1** for encryption only security • **2** for authentication of the management server and encryption Requires a user account with permission to access the remote computer's registry.

Option	Description
`-convert`	Converts an old key file from a previous release to the new key file format. For example: `NQKeyGenWindows -convert` *`oldkeylocation -newkeylocation`* Enables you to check an older key file generated using the NetIQ Encryption Utility (`rpckey.exe`) in AppManager 5.0.1 and earlier into the repository and continue using it for all of your agents. After converting an old key file, use the `-change` option to check the key information into the repository, set the security level to 1 with the `-seclev` option, and restart your management servers. For more information about updating an older key file after upgrading to AppManager, see the *Upgrade and Migration Guide for AppManager.*
`-verify`	Verifies the password and encrypted key file location are correct and can be imported into the repository. To use this option, you must specify the password used to create the public/private key and the location of the key file extracted from the repository. For example: `NQKeyGenWindows -verify` *`filelocation`* **Note** You are prompted to provide the password you specified when you created the key.

Key File Utility for UNIX Agents

The NetIQ Corporation key file generation program, `NQKeyGenUNIX.exe`, is a command-line program used to set the security level for a site and to generate and manage public/private keys for secure communication between the management server and UNIX managed computers. This utility is installed in the `NetIQ\AppManager\bin` folder when you run the AppManager setup program.

The basic syntax for the `NQKeyGenUnix.exe` program is:

`NQKeyGenUnix -`*`option value`*

Note
If you type `NQKeyGenUnix` without specifying any options, the program displays usage information.

The program supports the following command-line options.

Option	Description

`-db`	Specifies the login information for connecting to the repository using the following format:
	`NQKeyGenUnix -db database_name:user_name:sql_server`
	For example:
	`NQKeyGenUnix -db qdb:smithj:nyc2003`
	If you are using Windows authentication to connect to the repository, leave the username blank. If you are using SQL Server authentication, type a SQL Server username for connecting to the repository. The program prompts for the password to use for the SQL Server account.
	Note Most other options require you to specify connection information.

`-new`	Creates a record in the repository for the public/private key pair used to authenticate the management server to your UNIX agents. You must specify a password to create the key. For example:
	`NQKeyGenUnix -db db:user:sqlsvr -new`
	To create a new key file to share across multiple repositories on a computer other than the repository, you can use the command:
	`NQKeyGenUnix -new filelocation`
	This option creates a new private/public key pair with password protection in the specified file location without checking the new key into the repository.
	Note When you use the `-new` option, the `NQKeyGenUnix` utility prompts you to provide a key pair password.

`-change`	Changes the public/private key stored in the repository to use the new key file you specify. You must specify the key file password you used to create the key pair and the location of the key file to use. For example: `NQKeyGenUnix -db db:user:sqlsvr -change filelocation` This option enables you to check an existing key from a key file into a new repository when you want to share a key file across multiple repositories and management servers. **Note** When you use this option, you are prompted for the password you specified when you created the key pair.
`-ckey`	Extracts just the public key portion of the key file stored in the repository. You must specify a location for the public key file. For example: `NQKeyGenUnix -db db:user:sqlsvr -ckey filelocation` Once you extract the public portion of the key, you can copy the file and distribute it to your UNIX agents for authentication purposes.
`-skey`	Extracts the public and private key stored in the repository. You must specify a location for the key file. For example: `NQKeyGenUnix -db db:user:sqlsvr -skey filelocation` This option is used to check out the current key pair into a password-protected file. This file then can be checked into a different repository using the `-change` option.

Option	Description
`-seclev`	Sets the security level in the repository for communication between the management server and UNIX agents. The valid security levels are: • **0** for no security • **1** for encryption only security • **2** for authentication of the management server • **9** to remove all historical key-pairs while maintaining the current security level Removing historical key pairs enables you to manually expire older keys, as needed. **Note** If you change the security level, the change takes effect when the management server is restarted. For example, to set the security level to use authentication of the management server: `NQKeyGenUnix -db db:user:sqlsvr -seclev 2`
`-verify`	Verifies the password and encrypted key file location are correct and can be imported into the repository. To use this option, you must specify the password used to create the public/private key and the location of the key file extracted from the repository. For example: `NQKeyGenUnix -verify filelocation` **Note** When you use this option, you are prompted for the password you specified when you created the key pair.
`-ckeyinfo`	Display the public portion of the key as it is stored in the repository. For example: `NQKeyGenUnix -db db:user:sqlsvr -ckeyinfo` This option is useful for comparing the public key information stored in the repository with the public key information recorded in the UNIX agent log file to verify whether the correct key is being used.

Persistent Data Utility

When a management server receives events, data, and job status from its agents, it inserts the information into persistent IOC queue files for later processing. The IOC queue files help to ensure information is not lost if the management server stops before it can finish processing all of the incoming messages. These persistent IOC queues files are located in the `NetIQ\AppManager\dat\pioc` folder by default.

The persistent data program, `mspioc.exe`, is a command-line program for displaying either summary or detailed information about each PIOC queue. For example, you can use the `mspioc.exe` program to determine the total number of records received of the size in bytes of the queues. You can find the `mspioc.exe` program in the `\Extras` folder in the AppManager installation kit.

The syntax for the `mspioc.exe` program is:

`mspioc.exe` *`ms_pioc_file_path`* [*`detailed_report_path`*]

Where:

- *`ms_pioc_file_path`* is a required argument that indicates the path to the queue file for which you want information.
- *`detailed_report_path`* is an optional argument that specifies the location and filename for writing detailed information about the queue file.

If you do not specify the *detailed_report_path* argument, the `mspioc.exe` program displays summary information on the screen but does not create the detailed report. For example, to only display the summary information, enter a command similar to the following:

```
mspioc.exe "C:\Program Files\NetIQ\AppManager\dat\pioc\PiocData"
```

To create a detailed report about a persistent queue file, you must specify a location for the report. For example:

```
mspioc.exe "C:\Program Files\NetIQ\AppManager
\dat\pioc\PiocEvent" "C:\Temp\EventRpt.txt"
```

MAPI Mail Utility

The NetIQ Corporation MAPI mailer program, `NetIQMapiMail.exe`, is a command-line program used to send MAPI mail messages.

The basic syntax for the `NetIQMapiMail.exe` program is:

`NetIQMapiMail -t`*`Recipients`* `[-s`*`Subject`*`] [-m`*`Message`*`]`

`[-p`*`Profile`*`] [-w`*`Password`*`] [-f`*`AttachmentPath`*`]`

Use quotation marks around the entire parameter string if any information you specify contains blank spaces. For example, if you are specifying the subject line and using spaces between words, you would enclose the entire string in quotation marks similar to this:

```
NetIQMapiMail -tsmith@xyz.com -s"This is a test mail message"
```

The program supports the following options:

Option	Description
`-t`*`Recipients`*	Specifies the e-mail address of each individual you want to receive the report, using the format in the address book. To specify more than one address, separate each name with a semicolon (;). For example: `-t"Chris Lin;pat_conner@bigcorp.com"`
`-s`*`Subject`*	Specifies the text to use in the Subject line of the mail message.
`-m`*`Message`*	Specifies the body of the mail message.
`-p`*`Profile`*	Specifies the MAPI client profile name to use for sending the message.
`-w`*`Password`*	Specifies the password for the client profile you are using.
`-f`*`AttachmentPath`*	Specifies the full path to the file you want attached to the mail message.

SMTP Mail Utility

The NetIQ Corporation SMTP mailer program, `NetIQSMTPMail.exe`, is a command-line program used to send SMTP mail messages.

The basic syntax for the `NetIQSMTPMail.exe` program is:

`NetIQSMTPMail -t`*`Recipients`* `[-s`*`Subject`*`] [-f`*`From`*`] [-m`*`Message`*`]`

`[-h`*`Host:Port`*`] [-r`*`Filename`*`]`

Use quotation marks around the entire parameter string if any information you specify contains blank spaces. For example, if you are specifying the subject line and using spaces between words, you would enclose the entire string in quotation marks similar to this:

```
NetIQSMTPMail -tsmith@xyz.com -s"This is a test mail message" -fjones@abc.com -
hmailcenter:800 -rC:\Temp\NewsReport.txt
```

The program supports the following options:

Option	Description
`-t`*`Recipients`*	Specifies the e-mail address of each individual you want to receive the report, using the format: *`recipient@domain`* For example: `-tIT_Admin@ajuba.com` Separate names of multiple recipients by commas.
`-s`*`Subject`*	Specifies the text to use in the Subject line of the message.
`-f`*`From`*	Specifies the sender identified in the From line of the message.
`-m`*`Message`*	Specifies the body of the mail message.
`-h`*`Host:Port`*	Specifies the SMTP mail server hostname and, optionally, the port number on the server to use.
`-r`*`Filename`*	Specifies the full path to a file to attach to the message.

SNMP Trap Utility

The NetIQ Corporation SNMP trap program, `NetIQSNMPTrap.exe`, is a command-line program used to generate and send enterprise-specific SNMP traps. This program provides backend support for AppManager actions that send SNMP traps in response to events.

The basic syntax for the `NetIQSNMPTrap.exe` program is:

```
NetIQSNMPTrap [-a agent] [-c community_name]
[-d destination] [-f filename] [-i] [-m message]
[-o trap_oid] [-p trap_port] [-s specific_number]
[-v agent_varbind_oid]
```

The program supports the following options:

Option	Description
`-a` *agent*	Specifies the name of the computer originating the trap.
`-c` *community_name*	Specifies the community name to use.
`-d` *destination*	Specifies the SNMP destination or trap sink manager computer.
`-f` *filename*	Specifies the full path to the file containing a custom trap message.
`-i`	Installs or reset the registry entries under `HKEY_LOCAL_MACHINE\Software\NetIQ\AppManager\4.0\NetIQmc\SNMPTRAP\Config` to their default values.
`-m` *message*	Specifies the message associated with the trap.
`-o` *trap_oid*	Specifies the enterprise-specific Object Identifier.
`-p` *trap_port*	Specifies the port that receives SNMP traps.
`-s` *specific_number*	Specifies the enterprise-specific trap number.
`-v` *agent_varbind_oid*	Specifies one `varbind` object identifier for all SNMP traps containing the default AppManager event information or your custom message.

Synchronization Utilities

Two synchronization utilities are available to help you troubleshoot inconsistencies between AppManager agents and the repository.

The `mssync.exe` program checks for differences between the management server designation on the AppManager agent and the management server designation stored in the repository. Optionally, this program can also be used to correct the management server designation information in the repository so that it matches the designation information on the agent. For help on using this program, open a Command Prompt window and type `mssync -h`.

The `netiqsync.exe` program checks for differences in job status between AppManager agents and the repository, and optionally stops orphaned jobs. You must run this program on the management server computer. For help on using this program, open a Command Prompt window and type `netiqsync -h`. Contact NetIQ Technical Support for more information about using this program.

Time Conversion Utility

The time conversion program, `prttime`.exe, is used to convert a UTC time value to a comparable data and time string in plain text. You can find the `prttime.exe` program in the `Extras` folder in the AppManager installation kit.

The syntax for the `prttime.exe` program is:

```
prttime
```

You are then prompted to provide the UTC time you want to convert. For example, if you type the UTC time `1055439148`, the return value is:

```
Thu Jun 12 10:32:28 2003
```

Type Ctrl-C to exit the program.

Note
This program always returns the UTC time in its corresponding Pacific Standard Time, regardless of the time zone the local computer uses. By definition, UTC format is the number of seconds since January 1, 1970, 12:00am GMT. Therefore, if you type 0, the prttime.exe program returns Wed Dec 31 16:00:00 1969.

Appendix B
Registry Keys

Each AppManager component uses registry keys to control a range of operations. For most organizations, it is rare to need to modify any of these key values or edit the registry directly. In some cases, however, it is useful to understand how these keys are used. This appendix provides an overview of the common AppManager registry keys and the registry keys associated with each AppManager component.

For each key value, the following information is provided:

- Value name
- Description of what the value represents and the default value for the key or sample values to illustrate the entry format

All of the keys described in this appendix are under the **HKEY_LOCAL_MACHINE on Local Machine** registry entry. Depending on the configuration of your installation and the version of AppManager you are using, you may see registry keys not included in this appendix, different values, or keys displayed in a different format.

Modifying the Registry

In most cases, you set or modify registry key values from within AppManager components by making selections in the user interface or during installation. You can also use the **NTAdmin_RegistrySet** Knowledge Script or the Registry Editor to modify the registry keys.

Before running **NTAdmin_RegistrySet**, check that the AppManager agent services (`NetIQmc` and `NetIQccm`) are running on the target computer as `LocalSystem` or as an account with local Administrator privileges.

Note
You should always back up the registry before you modify any key values. You should also update your Emergency Repair Disk (ERD) before making any changes to the registry.

Basic NetIQ Registry Folder

All AppManager registry keys and folders are located in `HKEY_LOCAL_MACHINE` under `SOFTWARE\NetIQ`. This top-level folder is created when you install any NetIQ Corporation product on a computer.

Depending on the products and components you have installed on the local computer, the `NetIQ` folder may include some combination of the following folders for AppManager-related registry keys:

Folder	Content
`AppManager`	Key values for all of the AppManager components installed on the computer you are viewing.
`Common`	The path to the `NetIQ Corporation\Common` directory where files that can be used by multiple NetIQ Corporation products have been installed.
`Diagnostic Console`	Key values for the Diagnostic Console components installed on the computer you are viewing.
`Generic`	Keys used to customize the maximum log size of the trace log and the path where the log file is saved.
`Report Sharing Components`	The path to the directory where shared report components have been installed.
`Response Time`	Key values for the location of the schema file and tracing log used by AppManager Response Time modules.

Main AppManager Keys and Folders

The AppManager registry keys and folders located in `SOFTWARE\NetIQ\AppManager\4.0` contain the most important registry key information for AppManager Version 4.0 and later.

- The key values in `SOFTWARE\NetIQ\AppManager\4.0` provide version and build information for the AppManager executables (`.exe`) and libraries (`.dll`) installed on a computer. All of the values are recorded during installation and provide useful information to verify the compatibility of components installed on a computer. You should not manually change any of these values. The format for the version and build number is `n.n.nnnnn.n` (for example, `6.0.56615.0`) and is the same for all components.
- The registry folders in `SOFTWARE\NetIQ\AppManager\4.0` contain keys and subfolder keys that control the behavior of AppManager components installed on the computer and the communication between local AppManager components and AppManager components installed on other computers.

For example, you may see some or all of the following folders:

Folder	Associated Component	Contents
`AgtShared`	AppManager agent	Key values that define the characteristics of the managed computer's local repository.
`AMDevCon`	AppManager Developer's Console	Key values used to configure and store information about the Developer Console.
`IconEdit`	AppManager Developer's Console	Key values that indicate the path to any custom icon files you have added using the Icon Editor.
`Install`	AppManager Response Time module or agent	Key values that indicate the path to the uninstaller for modules.

Folder	Associated Component	Contents
NetIQccm	AppManager agent	Key values that configure communication with the management server. For more information about this key, see “NetIQccm Folder” on page 224.
NetIQmc	AppManager agent	Key values that configure the AppManager agent’s operation. For more information about this key, see “NetIQmc Folder” on page 227.
NetIQms	AppManager management server	Key values that configure communication with the repository and managed computers. For more information about this key, see “NetIQms Folder” on page 232.
QDB	AppManager repository	Key values that configure the data device, name, path, size and the log name, path, size. It also contains keys that shows the encrypted password and the path of the database installation. For more information about this key, see “QDB Folder” on page 238.
Repository Browser	AppManager Operator Console	Key values that define the saved queries available in the Repository Browser.
Security Manager	AppManager Security Manager	Key values that define the default security role and default SQL Server group for new users.
Web	Web management server	Key values that describe the path to the web management server. For more information about this key, see “Web Folder” on page 239.
WebRecorder	Web Transaction Recorder	Key values used to configure and store information about the Web Transaction Recorder.

AgtShared Folder

The SOFTWARE\NetIQ\AppManager\4.0\AgtShared folder stores keys that define the characteristics of the managed computer’s local repository.

Key	Description
DataCacheQueSize	Defines the maximum queue size for the agent services. The default is 5MB.
RpAccessMode	Defines the connection method for accessing the local repository. Currently, only ODBC is supported.
RPC Authentication	Indicates whether the agent service should authenticate the management server before sending encrypted data. • A value of 0 indicates no authentication is required. • A value of 1indicates authentication is required.
RPC Encryption	Indicates whether the RPC communication between the agent services and the management server should be encrypted. • A value of 0 indicates no encryption. • A value of 1indicates all communication is encrypted.
RpMaxCacheSize	Defines the maximum cache size for the local repository. A value of 0 indicates unlimited cache size.
RpPath	Identifies the path to the local repository. For example: C:\Program Files\NetIQ\AppManager\db

AMDevCon Folder

The `SOFTWARE\NetIQ\AppManager\4.0\AMDevCon` folder stores keys that define characteristics for the Developer Console.

Key	Description
`EbsDebugger`	Specifies the full path to the debugger for scripts written in Summit BasicScript (`.ebs`). The Developer Console checks this key, and if the path is specified, it starts that debugger when you click **Project > Debug**.
`KsCheckin`	Identifies the path to the Knowledge Script checkin program (`kscheckin.exe`). For example: `C:\Program Files\NetIQ\AppManager\bin`
`KsTemp`	Identifies the path to the folder where log files generated by `kscheckin.exe` are saved. For example: `C:\NetIQ\Temp\NetIQ_Debug\`*`server`
`NewKsPath`	Identifies the default path for checking in new Knowledge Scripts. For example: `C:\Program Files\NetIQ\AppManager\qdb`
`QBDName`	Identifies the name of the repository database.
`SQLServer`	Identifies the name of the repository server.
`SQLUser`	Identifies the name of a SQL Server user for logging in to the repository.
`VbsDebugger`	Specifies the full path to the debugger for scripts written in VBScript (`.vbs`). The Developer Console checks this key, and if the path is specified, it starts that debugger when you click **Project > Debug**.
`VbsProgArgs`	Specifies the program arguments to use with the VBScript debugger.

NetIQccm Folder

The `SOFTWARE\NetIQ\AppManager\4.0\NetIQccm` folder stores keys that define characteristics of the NetIQ Corporation AppManager Client Communication Manager service and how that service communicates with the management server. The keys are organized in the following subfolders:

- Admin
- Config
- Tracing

Admin

The `NetIQccm\Admin` folder contains keys that are used in agent self-monitoring.

Key	Description
`AdminEvtSev`	Defines the default AppManager event severity level for agent self-monitoring events. The default event severity level is 40.
`DisableAdminEvt`	Sets the flag that indicates whether an event should be raised if an agent needs to be restarted.
`IOCEventLogCommInt`	Specifies the communication interval in seconds for the agent to use in checking the Windows event log for new self-monitoring events. The default value is 60 seconds.
`MCFreezeThreshold`	Specifies the maximum amount of time, in seconds, that can elapse between timestamps to determine whether an agent should be restarted. Set by the AMAdmin_AgentSelfMon Knowledge Script.

Config

The `NetIQccm\Config` contains keys that control agent autonomy and communication with the management server.

Key	Description
`BatchLoad`	Specifies the maximum number of records (events and data) for the NetIQ Corporation AppManager Client Communication Manager to read from the local repository and send to the management server in a single batch. If communication with the management server fails, records are saved in the local repository. When communication with the management server is restored, this batch load value provides guidance for how much data the Client Communication Manager should attempt to send at one time. If the local repository has few records, the Client Communication Manager may upload all of the records at once for efficiency. If the communication between the Client Communication Manager and the management server is slow, the Client Communication Manager may need to transfer the data in a series of batches. The Client Communication Manager can adjust the number of records uploaded dynamically, decreasing the load when the management server is busy or communication is slow and increasing the load when the management server is free or communication improves.
`CacheRpcConn`	Sets the flag to control RPC connections. Zero disables caching, whereas a non-zero value enables caching. The default is zero.
`DataTableSize`	Defines the maximum size (number of records) for the local repository's Data table. This value is configured using the AMAdmin_SetLocalRPSize Knowledge Script. The default value is zero (0), which indicates no limit.
`EventLogLvl`	Defines the level of the event log messages in the Windows Event Log. The valid values for this key include: **0x0** - Don't log any events **0x1** - Log Info events **0x2** - Log Warning events **0x4** - Log Error events Values are added to combine the events logged. For example, **0x5** logs Info and Error events. The default is 0xF (Log everything).

Key	Description
EventTableSize	Defines the maximum size (number of records) for the local repository's Event table. This value is configured using the AMAdmin_SetLocalRPSize Knowledge Script. The key value is zero (0), which indicates no limit.
MonitorInterval	Defines the monitoring interval (in seconds) to check whether the NetIQmc service is running. If you set an interval, the NetIQ Corporation AppManager Client Communication Manager automatically restarts the NetIQ Corporation AppManager Client Resource Monitor service if it is detected down. To turn off self-monitoring, set this value to 0. The default value is 1800 seconds (30 min).
PingMSInterval	Defines the interval (in seconds) to check the status of the NetIQ Corporation AppManager Management Service. The default is 30 seconds, for example, 0x1e. **Note** The NetIQ Corporation AppManager Client Communication Manager service checks availability of the NetIQ Corporation AppManager Management Service to determine whether to send data and events to the management server or log them in the local repository. This key is used when managed computers run in **Autonomous** mode.
PollMCInterval	Defines the interval (in seconds) that the NetIQ Corporation AppManager Client Communication Manager uses to poll the NetIQ Corporation AppManager Client Resource Monitor for new data and events to be sent to the management server. The default is 5 seconds. This key is used when managed computers run in **Autonomous** mode.
RpcBatchHighWm	Defines the high watermark for tuning the flow of network communication from the NetIQ Corporation AppManager Client Communication Manager service to the management server. Set when you configure network flow with the AMAdmin_ConfigSiteNetFlowCtrl Knowledge Script.
RpcBatchIntv	Defines the communication interval for dynamically adjusting the batch size when the NetIQ Corporation AppManager Client Communication Manager service transfers data to the management server. Set when you configure network flow with the AMAdmin_ConfigSiteNetFlowCtrl Knowledge Script.
RpcBatchLowWm	Defines the low watermark for tuning the flow of network communication from the NetIQ Corporation AppManager Client Communication Manager service to the management server. Set when you configure network flow with the AMAdmin_ConfigSiteNetFlowCtrl Knowledge Script.
RpcBatchDynamicFlow	Sets the flag that enables or disables dynamic flow control tuning. Zero enables dynamic flow control. A non-zero value disables dynamic flow control. The default is zero.

Tracing

The NetIQccm\Tracing folder stores the key for tracing the activity of the NetIQ Corporation AppManager Client Communication Manager service.

Key	Description
TraceCCM	Sets the flag that enables or disables tracing for the AppManager Client Communication Manager service. A value of zero (0) turns tracing off and a non-zero value turns tracing on. Setting a higher value for this key generates more verbose tracing output. If set to 1 or higher, the Client Communication Manager logs information about its activity to the ccmtrace.log file.

NetIQmc Folder

The `SOFTWARE\NetIQ\AppManager\4.0\NetIQmc` folder stores keys that define characteristics of the NetIQ Corporation AppManager Client Resource Monitor service and control many aspects of the managed computer's behavior and communication with other services. There are several keys directly under the `NetIQmc` folder. Additional keys are organized in the following subfolders:

- Admin
- Config
- Security
- Tracing

Registry Keys in the NetIQmc Folder

The `NetIQmc` folder contains general-purpose keys for the NetIQ Corporation AppManager Client Resource Monitor.

Key	Description
`Exchange Mailbox`	Specifies the Exchange mailbox alias name to use if the managed computer can send MAPI mail as an action or if the agent is set to monitor Exchange Server on this computer. Set during installation when you enable MAPI mail as an action or install the Exchange managed object.
`Exchange Profile`	Specifies the Exchange profile name to use if the managed computer can send MAPI mail as an action or if the agent is set to monitor Exchange Server on this computer. Set during installation when you enable MAPI mail as an action or install the Exchange managed object.
`Exchange Server`	Specifies the Exchange Server name to use if the managed computer can send MAPI mail as an action or if the agent is set to monitor Exchange Server on this computer. Set during installation when you enable MAPI mail as an action or install the Exchange managed object.
`Local repository`	Specifies the path to the local repository. Set during installation. For example: `C:\Program Files\NetIQ\AppManager\db`
`MS Backup`	Specifies the management server you have identified as the backup management server for this managed computer. You designate the primary and backup management server by running the SetPrimaryMS Knowledge Script.
`MS Primary`	Specifies the management server you have identified as the primary management server for this managed computer. You designate the primary and backup management server by running the SetPrimaryMS Knowledge Script.
`NetIQms Port`	Specifies the RPC port number where the management server listens for communication from the NetIQ Corporation AppManager Client Resource Monitor service. The default is 9999 (0x270f).
`Port`	Specifies the RPC port where the AppManager Client Resource Monitor service listens for communications from the management server. The default is 9998 (0x270e).
`ServiceDependency`	A comma-separated list of services on which the managed computer is dependent. The Client Resource Monitor service checks for the dependent services before starting up. The value is dependent on the managed objects installed. For example, if the managed object for monitoring SQL Server is installed, this key may contain a list similar to this: `MSSQLSever,SQLExecutive,msftpsvc`

Key	Description
User Domain	Specifies the domain for a Windows user account if a Windows user account is being used for the agent services to run under. Set during installation if you: • enable MAPI mail as an action • install the Exchange managed object • install a report-enabled agent • choose to run the agent with a Windows account • use the agent to perform remote agent installation
User Name	Specifies the username for the Windows user account for the agent services. Set during installation under the same circumstances as the User Domain key.
User Password	Specifies the password for the Windows user account for the agent services. Set during installation under the same circumstances as the User Domain key.

Admin

The NetIQmc\Admin folder contains keys that are used in agent self-monitoring.

Key	Description
AdminEvtSev	Defines the default AppManager event severity level for agent self-monitoring events. Default event severity is 40.
DisableAdminEvtSev	Sets the flag to enable or disable the event severity level for agent self-monitoring events. Default event severity is 0.
DisableJobAbort	Sets the flag that enables or disables the ability of a script to abort a job. If this key is set to 0, the agent will allow a script to abort a job. If this key is set to 1, the agent will not allow a script to abort a job. Default is 0.
Knowledge Script Failure Events	Contains internal Knowledge Script abort and exception codes. This is not the same as the event severity value you can set on the Values tab of a Knowledge Script Properties dialog box.
LastMCCheck	Stores the timestamp of the last self-monitoring check in UTC format (seconds since January 1, 1970, 12:00 am GMT). If the timestamp is older than the MCFreezeThreshold, the NetIQ Corporation AppManager Client Communication Manager attempts to restart the NetIQ Corporation AppManager Client Resource Monitor service.

Config

The `NetIQmc\Config` folder contains keys that control agent autonomy, data persistence, event handling, service availability, and failover support.

Key	Description
`AppDetectionPollInterval`	The interval at which the agent scans local resources for new software for which there are modules available. This information is provided to the deployment services to install modules according to deployment services rules.
`Autonomy`	Sets the flag that enables or disables agent autonomy. If set to 1, the agent runs in Autonomous mode. If set to 0, autonomy is disabled. The default is 1.
`AutoUpdateMS`	Sets the flag that allows the managed computer to update its management server information automatically. If set to 1, automatic updates are allowed. If set to 0, the agent is prevented from automatically updating its management server information. If set to 1 and the management server is moved to another computer or the name of the management server computer changes, the managed computer updates its internal information about the management server to reflect the new information. The default is 1 to enable automatic updates.
`ConcurrentRptJob`	Specifies the maximum number of concurrent reports that can run on the agent. The default is 3.
`Disable64BitPerProvider`	Determines whether AppManager uses both the 32-bit and 64-bit performance providers or uses only the 32-bit performance provider.
`EventLogLvl`	Defines the level of the event log messages in the Windows log files. Valid values for this key include: **0x0** - Don't log any event **0x1** - Log Info events **0x2** - Log Warning events **0x4** - Log Error events Values are added together to combine the events logged. For example, **0x5** logs Info and Error events. The default value is 0xF (Log everything).
`JobSpacingInterval`	Controls how long an agent should pause for each subsequent job to run. This registry key allows you to set a time delay between the jobs to prevent overloading the system. The default delay is 3 seconds.
`JobStatusPollInt`	Sets the interval for performing a health check of the communication between the agent and the management server. At each job polling interval, the agent collects data about the list of jobs that are running, the version ID for each job, and the version number of the agent. The version number of the agent is useful information during upgrades. The default interval is 300 seconds.
`MonitorInterval`	Defines the monitoring interval (in seconds) to check whether the `NetIQccm` service is running. If you set an interval, the Client Resource Monitor service automatically attempts to restart the Client Communication Manager service if the service is detected down. To turn off self-monitoring, set this value to 0. The default value is 1800 seconds (30 min).

Key	Description
NoEventSev	Defines a severity level that does not raise an event. For example, you can set a value for this key to trigger an action when a condition is met but not raise an event in the Operator Console. The default value, 0, disables the key.
NoMSEvent	Specifies a comma-separated list of management server names that should not receive event information. The managed computer does not send event information to the computers you specify. For example, to restrict the management servers MARS and AJAX from receiving events from the AppManager agent on the local computer: `NoMSEvent:REG_SZ:MARS,AJAX` By default, no value is set for this key, indicating that no management servers are prevented from receiving event information.
PDH threshold	Threshold at which legacy performance providers are abandoned for PDH information.
Persistent	Sets the flag that enables or disables data persistence. If this key is set to 1, persistence is enabled and events and data are written to the local repository when communication with the management server fails. If this key is set to 0, events and data are only transferred to the management server. If communication with the management server is interrupted, any event or data collected during the interruption is lost. The default is 1.
PrimaryMSFailOverCtrlTimes	Specifies the threshold for the number of times the agent should send ping requests to the primary management server before failing over to the secondary management server. If the ping request fails the number of times specified, the agent identifies the primary management server as unavailable and transfers all events and data to the backup management server, if you have designated one. If you have not designated a secondary management server, events and data are written to the local repository until communication with the primary management server is restored. The default value for this key is 3 ping attempts.
PrimaryMSFailOverInterval	Specifies the interval in seconds to ping the primary management server. The default is 60 seconds.
StartUpDelay	Controls how long the agent should pause after the Client Communication manager service starts before starting an iteration of any job on the agent. The default is 15 seconds.
SvcWaitInterval	Specifies the time, in seconds, for the Client Resource Monitor service to wait before attempting to restart dependent services. The default is 5 seconds.
vbStringSpaceSize	Specifies the string size, in bytes, allocated for the Summit scripting engine to use. The default value is 1048576 characters (0x100000).
VMRPCNoOfRetry	Sets the number of retry attempts for making RPC calls from the agent to the management server to send software inventory and application detection data. The default is 3.
VMRPCTimeout	The time (in seconds) the agent will wait to receive an RPC response from the management server before producing an error. The default is 60.

Security

The `NetIQmc\Security` folder contains keys that control the management servers authorized to communicate with the managed computer and the operations that the local managed computer is authorized to perform.

Key	Description
`AllowDosCmd`	Specifies the management servers that are allowed to run DOS commands on the local computer. The default value, *, allows all management servers to initiate DOS commands. To create a restricted list of management servers that can run DOS commands, set this key to a comma-separated list of computer names. For example, if only `SHASTA` and `DYNAMO` are allowed to run DOS commands: `AllowDosCmd:REG_SZ:shasta,dynamo` This registry key value controls whether the General_RunDOS and Action_DosCommand Knowledge Scripts can run commands on this managed computer. **Note** Checking is based on the management server that initiates the job, not the user account that starts it. For example, if the management server `TANGO` starts a RunDOS job on `SHASTA`, but was not included in the key, the job on `SHASTA` will abort with an error.
`AllowMS`	Specifies the list of management servers that can communicate with the Client Resource Monitor. An asterisk (*) authorizes all management servers to communicate with the local computer. **Note** You should not use this registry key to enforce security or control communication between the management server and the managed computer within a single management site. If you have more than one management server in a site, use the SetPrimaryMS Knowledge Script to identify the primary and secondary management server for each managed computer. Use the Windows and UNIX key file utilities to manage security for the site.
`AllowReboot`	Specifies the management servers that can request the local managed computer to reboot. The default value, 0, prevents **all** management servers from rebooting managed computers. To restrict reboot operations to specific computers, enter a comma-separated list of computer names. For example: `AllowReboot:REG_SZ:NYC001,190.12.1.28` You can use the asterisk (*) wild card to permit all management servers to reboot the local managed computer. **Note** You must specify at least one computer if you want to use the Action_RebootSystem Knowledge Script.
`RemoveAllowMSStar`	Indicates whether to remove the anonymous authorization that allows all management servers to communicate with the agent (AllowMS set to *). When set to 1, this key updates the AllowMS key with current information when you change the agent's designated primary or secondary management server.

Tracing

The `NetIQmc\Tracing` folder stores keys for tracing the activity of the NetIQ Corporation AppManager Client Resource Monitor service and managed objects.

Key	Description
`TraceKS`	Specifies the tracing level for Knowledge Scripts. The higher the value, the more verbose the tracing output. Specifying 0 turns tracing off. The default is 1 (0x1). Enabling Knowledge Script tracing creates an ASCII copy of the compiled Knowledge Script with debugging line numbers in the *`TraceLogPath`*`\mc` subdirectory and logs entries for each job running on the agent in the `mctrace.log` file. A separate file is created for each job and action executed on the agent. The name of each file includes the related JobID and the SiteID.
`TraceMC`	Specifies the tracing level for the NetIQ Corporation AppManager Client Resource Monitor service. The higher the value, the more verbose the tracing output. Specifying 0 turns tracing off. The default is 1 (0x1). If you enable `TraceMC`, the NetIQ Corporation AppManager Client Resource Monitor (`NetIQmc`) records information about its activity in the `mctrace.log` file. The information recorded in this file includes the status of polling threads, job requests, and job execution. If you also enable `TraceKS`, the `mctrace.log` also includes line-by-line trace entries for each job running on the agent to help you step through the Knowledge Script as it is executed to locate a point of failure.
`TraceMOcomponent`	Specifies the tracing level for specific application managed objects (for example, use `TraceMOactiveds` to enable tracing for Active Directory managed objects). The higher the value, the more verbose the tracing output. Specifying a value of zero (0) turns tracing off. The default is 16 (0x10). If you enable tracing for any managed object, the `mo.log` records information about monitoring activity for that managed object. The information recorded includes all function calls made from the managed object during job execution.

NetIQms Folder

The `SOFTWARE\NetIQ\AppManager\4.0\NetIQms` folder stores keys that define characteristics of the NetIQ Corporation AppManager Management Service. These keys are the registry keys that you are most likely to be interested in or need to modify. They control many important characteristics of a management server's behavior and communication with other components. Several keys are directly under the `NetIQms` folder. Additional keys are organized in the following subfolders:

- Admin
- Config
- RP
- Tracing

Note
The Integration folder contains keys used by AppManager connectors.

Registry Keys in the NetIQms Folder

The `NetIQms` folder contains general-purpose keys for the NetIQ Corporation AppManager Management Service.

Key	Description
`NetIQmc Port`	Specifies the RPC port that the AppManager Client Resource Monitor service listens on for communication from the management server. The default is 9998 (0x270e).
`Port`	Specifies the RPC port number that the management server listens on for communication from the Client Resource Monitor service. The default is 9999 (0x270f).
`RP database`	Specifies the name of the repository database. Set during installation. For example: `QDB`
`RP DSN`	Specifies Data Source Name for the ODBC connection to the repository. Set during installation. For example: `QDBms`
`RP Logon Timeout`	Specifies the maximum period of time, in seconds, that the management server should wait for a successful connection to SQL Server. The management server uses this value to determine whether SQL Server is down when the management server attempts the connection. The default is 600 seconds.
`RP password`	Stores the encrypted password for the management server to use in logging on to the repository.
`RP server`	Stores the name of the computer where the repository is located.
`RP username`	Stores the username the management server uses to connect to the QDB. The default is `netiq`.
`Unix Port`	Specifies the port number that the UNIX agent uses to communicate with the management server.
`UUID`	Stores a unique identifier for the management server.

Admin

The `NetIQms\Admin` folder contains keys that are used in management server self-monitoring.

Key	Description
`AdminEvtSev`	Defines the default AppManager event severity level for management server self-monitoring events. The default event severity level is 40.
`DisableAdminEvt`	Sets the flag indicating whether an event should be raised if an agent needs to be restarted.
`EnableMachineGrayOut`	Controls the display of computers in the TreeView pane of the Operator Console. If set to 1, computers that are unavailable to the management server are displayed as grayed-out in the TreeView pane of the Operator Console. You cannot start or stop AppManager jobs inactive computers. If set to 0, computers that are unavailable to the management server are not grayed out in the TreeView pane of the Operator Console. The default value is 1.
`General failure Events`	Used internally by management servers for specific error conditions.
`General Success Events`	Used internally by management servers for specific success conditions.
`GenKeyMismatchEvents`	Specifies whether a key mismatch between the agent and the management server should generate an Application Log event. The default value is 1 to generate an event is there is a mismatch in the key information.
`LogOptionalEvt`	Set the flag to enable or disable logging of optional events. The default is 0.
`MinKeyMismatchEventPeriod`	Specifies the number of seconds to wait before logging another Application Log event if a key mismatch is detected.

Config

The `NetIQms\Config` folder contains keys that control event and data handling and communication with managed computers.

Key	Description
`Allow Agent Install`	Identifies the management server to use for performing remote agent installation. Set to 1 to all the local management server to be used for remote installation jobs. Set to 0 to prevent a management server from attempting to start installation jobs. The default value is 1. **Note** If you have more than one management server in your environment, you should select a single management server for performing installation-related tasks and manually set this registry key to 0 on all other management servers.
`Comm Timeout`	Specifies the time in milliseconds for the management server to wait before sending a message to a managed computer if there is an outstanding call. The default is 5000 milliseconds.
`Data Thread`	Specifies the number of data worker threads. Increasing this value increases the number of connections to the database, thereby increasing overhead. The default is 2 threads.
`Enable Flow Control`	Sets the flag that grants control over network traffic to the Client Communication Manager service on the management server. Enabling flow control allows the agent on the management server to use a high watermark, low watermark, and a transfer interval to dynamically adjust the flow of data. Set this key to 0 to disable flow control. Set this key to 1 to enable flow control. The default is 1.
`Event Thread`	Specifies the number of event worker threads. Increasing this value increases the number of connections to the database, thereby increasing overhead. The default is 1 thread.
`Job Poll Interval`	Sets the interval, in seconds, for the job polling thread to check for outstanding (pending) job requests. The default is 5 seconds.
`Job Resend Frequency`	Specifies the maximum number of synchronization updates to ignore before checking for orphan or error jobs. The default is 1 update.
`Job Status Change Batch`	Determines whether the management server updates the QDB in batches or sequentially. If set to 1, the management server updates the QDB in a single batch when it receives updates from the agent. If set to 0, the management server updates the QDB separately for each job. The default is 0.
`Job Status Change Thread`	Specifies the number of job status change worker threads. Increasing this value increases the number of connections to the database, thereby increasing overhead. The default is 1.
`Job Status Check Interval`	Set the interval, in seconds, for the job status check thread. The job status check thread performs a “job handshake” with each managed computer. The default is 300 seconds.
`Job Status IP Refresh Interval`	Sets the minimum interval, in seconds, before an IP name resolution refresh is performed by the job status check thread. The default is 1000 seconds.
`Machine Poll Interval`	Sets the interval, in seconds, for the machine polling thread.The default is 500 seconds.
`MaxQueueItem`	Sets the maximum size of this request queue for UNIX and Linux communications. When a request comes from a UNIX agent, it is placed in a request queue for processing. Whenever a thread from the thread pool becomes available (idle), it picks up an item in this request queue to process. The default maximum size is 1000 items.

Key	Description
MaxSocketThread	Sets the maximum number of threads in the thread pool for processing requests.
MC Job Abort Event Sev	Sets the severity level for an event caused by job aborting. The default severity level is 10.
Number of Update Retry	Specifies the number of times a management server should attempt an update. The default is 1.
OptimisticSend	Instructs the management server to clear out the list of pending requests after responding to the heartbeat from a UNIX agent.
Orphan Job Event Sev	Sets the severity level for an event caused orphaned jobs. The default is 5.
Persistent IOC	Sets the flag to enable or disable the persistent IOC data and event mapping file. The default is 1.
Ping Machine Poll Interval	Sets the interval for the ping machine thread. The default is 5 seconds.
PIOC Data Map File Size MB	Defines the size of the persistent IOC file for data. The default is 25 MB.
PIOC Event Map File Size MB	Defines the size of the persistent IOC file for event information. The default is 5 MB.
PIOC JobStat Map File Size MB	Defines the size of the persistent IOC file for job information. The default is 5 MB.
PIOC Map File Path	Specifies the path to where the persistent IOC file is stored. For example: `C:\Program Files\NetIQ\AppManager\dat\pioc`
RecordEvents	Sets the flag that controls how the management server handles events. The valid values for this key are: • **0** - Don't record events in the QDB • **1** - Record events in the QDB (normal operation) • **2 -** Record and automatically acknowledge events in the QDB The default is 1.
RPC Encryption	Specifies whether the management server uses encrypted communication. If set to 1, RPC encryption is enabled. If set to 0, encryption is not enabled. Set during installation.
SocketRetryInterval	Defines the retry interval (in milliseconds) for sending or receiving data between a UNIX agent and the management server. If the management server fails to send data to the agent, it will wait for the retry interval and then try again. The default is 500. The number of retry attempts is defined by the MaxSocketRetryCount value, which has a default value of 10.
StateChangeClassic	Defines the content of short and detailed event messages for state change events when the **Generate a new event when original event condition no longer exists** option is selected on the Advanced tab of a Knowledge Script Properties dialog box. When set to 0, the short event message contains “State Change” and the detailed message contains the original message for the event that was closed. When set to 1, the short event message contains the original event ID and the string “State Change” and the detailed message contains the original event ID for the event that was closed. The default is 0.

Key	Description
Unix Machine Check Interval	Sets the interval, in seconds, to control how often the management server checks the time of the last heartbeat signal from each of its UNIX agents. Used in conjunction with the Unix Machine Timeout value to determine whether a UNIX or Linux computer is available. If the management server hasn't received a heartbeat signal within the period specified as the Unix Machine Timeout value, the UNIX agent is considered unavailable. The default is 300 seconds.
Unix Machine Timeout	Sets the interval, in seconds, that identifies a UNIX agent as unavailable. If the UNIX agent does not send a heartbeat signal within this period of time, it is considered unavailable. The default is 300 seconds.

RP

The NetIQms\RP folder contains keys that control connections to the repository.

Key	Description
RP Connection Refresh Wait	Sets the interval between the management server's attempts to reconnect to the repository when SQL Server goes down.The default is 300 seconds.
RP RPC Shutdown Threshold	Specifies the number of times the management server tries to connect to SQL Server before shutting down the RPC Server thread when SQL Server goes down. The default is 0.
Stop Rpc On SQL Failure	Sets a flag to stop the RPC service when the management server detects that the SQL Server is down. Setting this key to 1 stops the RPC service and 0 turns this feature off. The default is 1.

Tracing

The NetIQms\Tracing folder stores keys for tracing the activity of the NetIQ Corporation AppManager Management Service and connections to the repository.

Key	Description
Action Log	Sets the flag to enable or disable tracing for management server actions and proxy actions. Set to 1 to enable action debugging for management server actions. Information about action processing errors are written to the msaction.log file. The default is 0 (off).
NT Event Log Tracing Threshold	Sets the flag to enable or disable tracing for the Windows event logs. Set to 1 to enable tracing for the Windows logs. The default is 1 (on).
Qdb Log	Sets the flag to enable or disable tracing for repository communication errors. Set to 1 to enable repository connection tracing. Errors are written to msqdb.log file. The default is 0 (off).
Rpc Log	Sets the flag to enable or disable tracing for RPC. Set to 1 enable tracing for RPC communication between the management server and the agents. The default is 0 (off).
TraceCache	Sets the flag to enable or disable tracing for the cache. Set to 1 to enable tracing for the IOC cache processing on the management server. The default is 0 (off).

Key	Description
TraceData	Sets the flag to enable or disable tracing for data processing. Set to 1 to enable data point tracing on the management server. The default is 0 (off).
TraceEvent	Sets the flag to enable or disable tracing for event processing. Set to 1 to enable event tracing on the management server. The default is 0 (off).
TraceJob	Sets the flag to enable or disable tracing for job processing. Set to 1 to enable job tracing on the management server. The default is 0 (off).
TraceOther	Sets the flag to enable or disable tracing for other processing. Set to 1 to enable other process tracing. The default is 1 (on).
TraceQueueSize	Sets the size of the temporary message queue. The management server writes log messages to a queue, which is cached in memory temporarily. When the queue reaches the size set by this registry key, a background thread is activated to flush the messages from the temporary queue to the log file.The default is 100.
TraceRepository	Sets the flag to enable or disable tracing for communication with the repository. Set to 1 to enable tracing. The default is 1 (on).
TraceRPC	Sets the flag to enable or disable tracing for RPC communication. Set to 1 to enable RPC tracing between the management server and the repository. The default is 1 (on).
TraceSockets	Sets the flag to enable or disable tracing for socket communication. Set to 1 to turn on socket tracing. The default is 0 (off).
TraceTimeOut	Sets the interval for activating the background thread that flushes the messages from the temporary queue to the log file. In addition to the TraceQueueSize, this key controls when the background thread should flush messages to the log file.The default is 60 seconds.
TraceXml	Sets the flag to enable or disable tracing for XML communication. Set to 1 to turn on XML tracing. The default is 0 (off).

QDB Folder

The SOFTWARE\NetIQ\AppManager\4.0\QDB*servername*\QDB folder contains keys that store information about the configuration of the repository database.

Note
This folder is only available in upgraded environments. New AppManager 8.0 installations do not have this folder.

Key	Description
Data Device Name	Specifies the name of the repository data device specified during installation. Set during installation.
Data Device Path	Specifies the path for the repository data device specified during installation. Set during installation.
Data Device Size	Specifies the size of the repository data device specified during installation. Set during installation.
Database Name	Specifies the name for the repository database specified during installation. Set during installation.

Key	Description
Log Device Name	Specifies the name for the repository log device specified during installation. Set during installation.
LogDevice Path	Specifies the path for the repository log device specified during installation. Set during installation.
LogDevice Size	Specifies the size of the repository log device specified during installation. Set during installation.
sa Password	Specifies the encrypted password for the system administrator. Set during installation.
SQL Path	Specifies the base path for SQL Server. For example: C:\Program Files\Microsoft SQL Server\MSSQL
SQL Server Name	Name and path of the SQL Server computer.

Repository Browser

The SOFTWARE\NetIQ\AppManager\4.0\Repository Browser folder contains keys that define the default and custom queries that are available in the Repository Browser.

Security Manager

The SOFTWARE\NetIQ\AppManager\4.0\Security Manager folder contains keys that record the objects you have expanded in the Security pane, the identifier and name of the role you are using as the default role, and the default SQL Server group for new SQL users.

Note
This folder is only available in upgraded environments. New AppManager 8.0 installations do not have this folder.

Web Folder

The SOFTWARE\NetIQ\AppManager\4.0\Web folder contains keys that define the path to the AppManager Web management server and the Active Server Pages the Web management server uses.

Note
This folder is only available in upgraded environments. New AppManager 8.0 installations do not have this folder.

Other Folders

The SOFTWARE\NetIQ\Common\AMReports folder contains keys used in configuring and viewing AppManager reports. The SOFTWARE\NetIQ\Generic folder contains keys that define the maximum size for tracing files and the directory to use for tracing output.

Appendix C
AM Health Knowledge Scripts

The AppManager Self Monitoring module, also known as AM Health, provides Knowledge Scripts for monitoring the health and availability of AppManager components. From the Knowledge Script view of Control Center, you can access more information about any NetIQ-supported Knowledge Script by selecting it and clicking **Help**. In the Operator Console, click any Knowledge Script in the Knowledge Script pane and press **F1**.

Knowledge Script	What It Does
CCComponentsHealth	Monitors the health and availability of SQL Server resources associated with Control Center components.
HeartbeatUNIX	Monitors the AppManager agent heartbeat in a UNIX environment.
HeartbeatWin	Monitors the AppManager agent heartbeat in a Microsoft Windows environment.
QDBComponentsHealth	Monitors the health and availability of SQL Server resources associated with the AppManager repository (QDB) and the Management Server.
Recommended Knowledge Script Groups	Perform essential monitoring of AppManager components.

Monitoring the Health of a Management Server in an Untrusted Domain

If you want to monitor the health of a Management Server that is in an untrusted domain for your AppManager installation, you need to enable that Management Server to use a SQL Server user to authenticate with the QDB. You will also need to use a SQL Server user to allow the AM Health Knowledge Scripts on the Management Server to communicate with the QDB on the SQL Server.

You cannot use Windows authentication because the SQL Server will not be aware of any users that belong to the untrusted domain. Instead, set up SQL Authentication with AppManager Security Manager to allow the Management Server to connect to the QDB.

To monitor the health of a Management Server in an untrusted domain:

1. On the Extensions menu in the Operator Console, click **Security Manager**.
2. Connect to the QDB communicating with the Management Server in the untrusted domain.
3. Expand **Computers** and select the Management Server from the list of computers.
4. On the Custom tab, click **Add**.
5. In the Label field, type `sql$<management server name>`. For example, if your Management Server name is SERVER1TEST, you would type `sql$SERVER1TEST`.
6. In the Sub-Label field, type the SQL user name that exists in the QDB and will be used by the AMHealth_QDBComponentsHealth Knowledge Script running on the Management Server agent to communicate with the QDB.
7. In the Value 1 field, type the password for the user entered in the Sub-Label field.
8. Select **Extended application support** to encrypt the password when it is stored in the repository.
9. Click **OK**.
10. Click **Apply** to save the changes.
11. When you want to discover the Management Server, type the SQL user name from Step 6 of this procedure into the *SQL Server Login* parameter of the Discovery_AMHealth Knowledge Script.
12. When you want to run the AMHealth_QDBComponentsHealth Knowledge Script on the Management Server agent, type the SQL user name from Step 6 of this procedure into the *SQL username* parameter for the AMHealth_QDBComponentsHealth script.

Note

If you want to use the AMHealth_CCComponentsHealth Knowledge Script to monitor a command queue service (CQS) that is not in the same domain as the Control Center repository (NQCCDB), set up the user account using the above process for the command queue service instead of a Management Server.

CCComponentsHealth

Use this Knowledge Script to monitor the health and availability of Microsoft SQL Server resources associated with AppManager Control Center components. These components include the Cache Manager, the command queue service (CQS), Deployment services, and the Control Center repository (NQCCDB).

This script monitors the percentage of database space and log space used, the amount of time required for a SQL command or query to execute, and the status of AppManager scheduled tasks, which are SQL Server agent jobs operating on the NQCCDB. This script can restart a service or job that is down.

Note
If the NQCCDB does not have an AppManager agent installed on it, Discovery_AMHealth discovers the NQCCDB components on the server running the command queue service. As a result, the service and database monitoring parameters for this script run remotely. In this situation, you must have sufficient privileges on the service account for the NetIQMC service on the server running the command queue service so that the service account can remotely access the SQL Server service on the NQCCDB to obtain its status. If the account does not have proper privileges, the script will be unable to access the service status and will report that the SQL Server service is down even when it is not. If you do not have sufficient access for the service account for the NetIQMC service, deselect the *Raise an event if SQL Server services are down* and *Restart SQL Server services that stop unexpectedly* parameters in this script to avoid raising unnecessary events.

If you want to use CCComponentsHealth to monitor a command queue service that is not in the same domain or trusted domain as the NQCCDB, see "Monitoring the Health of a Management Server in an Untrusted Domain" on page 242.

This script raises events for the following situations:

- SQL Server services are down or have been restarted
- SQL Server agent jobs are disabled, missing, or have failed
- The command queue service is down or is not connected to the Control Center repository
- Control Center Cache Manager errors occur
- SQL Server queries against the NQCCDB take too long to process
- Deployment services are down
- Database or log space is low, and there is insufficient disk space for further growth

If you do not have an agent installed on the NQCCDB server, the repository component gets discovered on the command queue service. If you try to remotely monitor the NQCCDB from the command queue service using the CCComponentsHealth script, the script will not be able to obtain the disk information remotely.

As a result, if the repository component is monitored remotely from the command queue service by the CCComponentsHealth script, the following NQCCDB component monitoring parameters under the *SQL Server File Size and Growth Settings Monitoring* Event Notification Knowledge Script section will not be available:

- Raise an event if SQL Server maximum file size exceeds available disk space?
- Raise an event if insufficient space available for further file growth?

Resource Objects

- Cache Manager
- Command queue service
- Deployment services
- Control Center repository

Default Schedule

The default interval for this script is every 30 minutes.

Setting Parameter Values

Set the following parameters as needed:

Parameter	How to Set It
Event Notification	
Raise an event if SQL Server services are down?	Select **Yes** to raise an event if Control Center SQL Server services are down. The default is Yes. **Tip** Use Action Script for notification, as the Control Center repository will not be available.
Event severity when SQL Server services are down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which SQL Server services are down. The default is 10.
Restart SQL Server services that stop unexpectedly?	Select **Yes** to restart SQL Server services that stop unexpectedly, such as when not as part of scheduled maintenance. The default is Yes.
Raise an event if SQL Server services cannot be restarted?	Select **Yes** to raise an event if SQL Server services cannot be restarted. The default is Yes. **Tip** Use Action Script for notification, as the Control Center repository will not be available.
Event severity when SQL Server services cannot be restarted	Set the event severity, from 1 to 40, to indicate the importance of an event in which SQL Server services cannot be restarted. The default is 5.
Raise an event if SQL Server services are restarted?	Select **Yes** to raise an event if SQL Server services are successfully restarted. The default is Yes.
Event severity when SQL Server services are restarted	Set the event severity level, from 1 to 40, to indicate the importance of an event in which SQL Server services are successfully restarted. The default is 25.

Parameter	How to Set It
Raise an event if Control Center Cache Manager errors occur?	Select **Yes** to raise an event if the Control Center Cache Manager experiences one of the following: • An error related to Microsoft Distributed Transaction Control (MSDTC) • An error with the synchronization of data between Control Center and the QDB repositories that are synchronized to Control Center • An error from a Control Center SQL Server agent job that has failed The default is Yes. **Note** Not all Control Center SQL Server Agent jobs are monitored. Only those that are related to Cache Manager are monitored. The following jobs are monitored for errors that affect Cache Manager: • NetIQ CC Manage SQL Jobs NQCCDB • NetIQ CC Hourly Task NQCCDB • NetIQ CC Half-Hourly Task NQCCDB • NetIQ CC Daily Task NQCCDB Cache Manager is a child process of the command queue service. Cache Manager runs the Control Center queries on AppManager repository computers to retrieve view information for Control Center management groups. After the first iteration of the Knowledge Script completes, only new errors generated since the previous iteration will be raised as events.
Event severity when Cache Manager errors occur	Set the event severity, from 1 to 40, to indicate the importance of an event in which Cache Manager experiences errors. The default is 10.
Raise an event if Control Center SQL Server jobs are missing?	Select **Yes** to raise an event if SQL Server jobs are missing. The default is Yes. A missing job is one that may have been deleted or renamed.
Event severity when Control Center SQL Server jobs are missing	Set the event severity, from 1 to 40, to indicate the importance of an event in which SQL Server jobs are missing. The default is 15.
Raise an event if Control Center SQL Server jobs are disabled?	Select **Yes** to raise an event if SQL Server jobs are disabled. The default is Yes.
Event severity when Control Center SQL Server jobs are disabled	Set the event severity, from 1 to 40, to indicate the importance of an event in which SQL Server jobs are disabled. The default is 15.
Enable SQL Server jobs that are disabled?	Select **Yes** to enable, or start, SQL Server jobs that are disabled. The default is No.
Event severity when SQL Server jobs cannot be enabled	Set the event severity, from 1 to 40, to indicate the importance of an event in which disabled SQL Server jobs cannot be enabled, or started. The default is 20.
Raise an event if SQL Server jobs are successfully enabled?	Select **Yes** to raise an event if disabled SQL Server jobs are successfully enabled, or started. The default is No.
Event severity when SQL Server jobs are successfully enabled	Set the event severity level, from 1 to 40, to indicate the importance of an event in which disabled SQL Server jobs are successfully enabled, or started. The default is 20.
Raise an event if a Control Center SQL Server job fails?	Select **Yes** to raise an event if any step in a SQL Server job fails. The default is Yes.
Event severity when a Control Center SQL Server job fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a SQL Server job fails. The default is 20.

Parameter	How to Set It
Raise an event if query process time exceeds threshold?	Select **Yes** to raise an event if the amount of time it takes to process a SQL query exceeds the threshold you set. The default is Yes.
Threshold -- Maximum process run time	Specify the maximum length of time it can take SQL Server processes to run before an event is raised. The default is 300 seconds.
Event severity when query process time exceeds threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which SQL query processing time exceeds the threshold you set. The default is 10.
Event severity when query process time cannot be retrieved	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the script cannot determine the processing time for SQL queries. Processing time could be prevented from being retrieved if the query fails due to blocking, or if a connection cannot be made to the Control Center QDB SQL Server, which would cause the query to fail. If the query fails, the event will be raised without exception, which would inform users if there was an issue with the Control Center QDB SQL Server. The default is 10.
Raise an event if Deployment services are down?	Select **Yes to** raise an event if Control Center Deployment services are down. The default is Yes. The following situations can cause Deployment services to be down: • The Control Center server is restarted. • Deployment services took too long to restart. • The connection to the Control Center repository is unavailable. • The proxy server is unavailable.
Event severity when Deployment services are down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which Deployment services are down. The default is 10.
Raise an event if unable to retrieve Control Center component information?	Select **Yes** to raise an event if the script does not have the discovery details for the Control Center components, or if the discovery details are empty. The default is Yes.
Event severity when unable to retrieve Control Center component information	Set the event severity level, from 1 to 40, to indicate the importance of an event in which AppManager cannot retrieve Control Center Component information. The default is 10.
NetIQ Control Center Command Queue Service Monitoring	
Raise an event if Command Queue Service is not connected to the Control Center database?	Select **Yes** to raise an event if the command queue service (CQS) is not connected to the Control Center database (`NQCCDB`). The default is Yes. The command queue service polls the Command Queue table for queries to run. The Command Queue table stores queries that collect view information based on the criteria defined in Control Center.
Event severity when Command Queue Service is not connected to the Control Center database	Set the event severity, from 1 to 40, to indicate the importance of an event in which the command queue service is not connected to the Control Center database (`NQCCDB`). The default is 10.
Raise an event if Command Queue Service is down?	Select **Yes** to raise an event if the command queue service is down. The default is Yes.
Event severity when Command Queue Service is down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which command queue service is down. The default is 10.
Restart Command Queue Service if the service is down?	Select **Yes** to restart the command queue service if it is down. The default is Yes.
Raise an event if Command Queue Service is restarted?	Select **Yes** to raise an event if the command queue service is successfully restarted. The default is Yes.

Parameter	How to Set It
Event severity when Command Queue Service is restarted	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the command queue service is successfully restarted. The default is 25.
Raise an event if Command Queue Service cannot be restarted?	Select **Yes** to raise an event if the command queue service cannot be restarted. The default is Yes.
Event severity when Command Queue Service cannot be restarted	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the command queue service cannot be restarted. The default is 5.
SQL Server File Size and Growth Settings Monitoring	
Raise an event if insufficient space available for further file growth?	Select **Yes** to raise an event if the amount of available disk space is not enough to allow the file to continue to grow. SQL Server has growth settings on the Database Data and Log files. If the amount of free space on the disk is lower than this growth setting, an event will be raised to keep the files from attempting to grow and causing the SQL Server databases to become corrupted. The default is Yes.
Event severity when available space is insufficient	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the available server space is insufficient. The default is 10.
Raise an event if file growth rate falls below the threshold?	Select **Yes** to raise an event if the growth rate of the database data and log files falls below the threshold you set. If the growth rate is low, the data and log files become fragmented, which negatively impacts performance. **Note** The two threshold parameters (MB and percentage) are evaluated individually depending on whether the database file growth setting is in MB or a percentage. The default is Yes.
Threshold -- Minimum growth rate in MB	Specify in megabytes the minimum growth rate of SQL Server files before an event is raised. The default is 256 MB.
Threshold -- Minimum growth rate in percentage	Specify as a percentage the minimum growth rate of SQL Server files before an event is raised. The default is 9%.
Event severity when file growth rate falls below the threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the rate of growth of SQL Server files falls below the threshold you set. The default is 10.
Raise an event if Autogrowth is not enabled and file usage exceeds the threshold?	Select **Yes** to raise an event if the file usage exceeds the threshold and Autogrowth is not enabled. The default is Yes. The Autogrowth feature allows a database to grow by the amount of space required by a file or a transaction.
Threshold -- Maximum file usage with Autogrowth disabled	Specify the maximum amount of file usage with Autogrowth disabled that can occur before an event is raised. The default is 90%.
Event severity when Autogrowth disabled and usage exceeds threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the file usage exceeds the threshold and Autogrowth is not enabled. The default is 10.
Raise an event if SQL Server maximum file size exceeds available disk space?	Select **Yes** to raise an event if the maximum SQL file size (`MAXSIZE`) is set to a value greater than the amount of available disk space. The default is No. The `MAXSIZE` value identifies the maximum size to which a SQL Server database can grow.
Event severity when SQL maximum file size exceeds disk space	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the maximum SQL file size is set to a value greater than the amount of available disk space. The default is 15.

Parameter	How to Set It
Data Collection	
Collect data for database space utilization?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns the percentage of database space used on the Control Center server. The default is No.
Collect data for log space utilization?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns the percentage of log space used on the Control Center server. The default is No.
Collect data for Command Queue Server connection status?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns **100** if the command queue service is up and **0** if the command queue service is down. The default is No. **Note** This script only collects data for the command queue service connection status when you run the script on the Control Center database (`NQCCDB`). The script does *not* collect data when you run it on the command queue service.
Collect data for SQL Server service status?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns **100** if SQL Server services are up and **0** if SQL Server services are down. The default is No.
Monitoring	
SQL username	Specify the username required to access SQL Server on the Control Center server.
Event severity for a Knowledge Script error	Set the event severity level, from 1 to 40, to indicate the importance of an event in which an unexpected Knowledge Script error occurs. The default is 10.

HeartbeatUNIX

Use this Knowledge Script to test the heartbeat of the AppManager agent computer running UNIX. A heartbeat is a periodic signal generated by an AppManager agent computer to indicate that it is still running.

You can set this script to raise an event for the following conditions:

- Heartbeat fails
- An agent is healthy, such as when the heartbeat returns after failing
- The agent heartbeat fails a user-specified number of times

Note
If you use this Knowledge Script with the AppManager Operator Console, you can access the Actions and Advanced tab, but the options on those two tabs will not function.

Resource Objects

UNIX servers

Default Schedule

The default interval for this script is every five minutes.

Setting Parameter Values

Set the following parameters as needed:

Parameter	How to Set It
Heartbeat Options	
Raise an event if the agent heartbeat fails?	Select **Yes** to raise an event if the heartbeat for the AppManager agent server stops. The default is Yes.
Event severity when the agent heartbeat fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the heartbeat stops. The default is 5.
Raise an event when agent heartbeat restarts?	Select **Yes** to raise an event if the heartbeat starts again after stopping. The default is Yes.
Event severity when the agent heartbeat restarts	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the heartbeat starts again after stopping. The default is 25.
Number of consecutive heartbeat failures before raising an event	Specify the number of times the heartbeat must fail before raising an event. The default is 1.
Generate heartbeat data?	Select **Yes** to enable the heartbeat check. If you select Yes and the data point from this job is missing, AppManager raises an event. If you select No, the heartbeat check will not look for the data point from this job. The default is Yes.
Heartbeat Investigation Steps (Used by Management Server)	AppManager performs the following steps only if the heartbeat event or the heartbeat data is missing.
Attempt to contact agent computer by ICMP ping?	Select **Yes** to send an ICMP ping request to the agent computer. The default is Yes.
Perform tracert diagnostic if ICMP ping fails?	Select **Yes** to run a tracert (traceroute) diagnostic test if the ping request fails. The default is Yes. A traceroute test helps you troubleshoot network routing problems that can block ICMP traffic. This script raises an event if the tracert fails.

HeartbeatWin

Use this Knowledge Script to test the heartbeat of the AppManager agent computer. A heartbeat is a periodic signal generated by an AppManager agent computer to indicate that it is still running.

This script raises events if the heartbeat for the agent computer stops or restarts, and it generates a data point about the heartbeat events. You can also use this script to track whether jobs finish in the expected time frame, or if they exceed the maximum run time.

You can set this script to raise an event for the following conditions:

- Heartbeat fails
- An agent is healthy, such as when the heartbeat returns after failing
- The agent heartbeat fails a user-specified number of times
- Jobs take longer than expected to execute
- Jobs exceed maximum run time
- No jobs are found

Note
If you use this Knowledge Script with the AppManager Operator Console, you can access the Actions and Advanced tab, but the options on those two tabs will not function.

Resource Objects

Windows servers

Default Schedule

The default interval for this script is every five minutes.

Setting Parameter Values

Set the following parameters as needed:

Parameter	How to Set It
Heartbeat Options	
Raise an event if the agent heartbeat fails?	Select **Yes** to raise an event if the heartbeat for the AppManager agent server stops. The default is Yes.
Event severity when the agent heartbeat fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the heartbeat for the AppManager agent server stops. The default is 5.
Raise an event when agent heartbeat restarts?	Select **Yes** to raise an event if the heartbeat starts again after stopping. The default is Yes.
Event severity when the agent heartbeat restarts	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the heartbeat starts again after stopping. The default is 25.
Number of consecutive heartbeat failures before raising an event	Specify the number of times the heartbeat must fail before raising an event. The default is 2.
Generate heartbeat data?	Select **Yes** to enable the heartbeat check. If you select Yes and the data point from this job is missing, AppManager raises an event. If you select No, the heartbeat check will not look for the data point from this job. The default is Yes.

Parameter	How to Set It
Job Monitoring Options	
Monitor individual jobs?	Select **Yes** to monitor *all* jobs running on the agent that came from the same QDB as the heartbeat job. If you select No, the heartbeat job simply sends the heartbeat event and data according to the heartbeat-related parameters you set for this Knowledge Script. The default is Yes.
Raise an event if jobs take longer than average to execute?	Select **Yes** to raise an event that lists all jobs that are taking longer to execute than their average execution time. The agent stores a list of the average times jobs take to execute. The default is Yes.
Ignore jobs running for less than this amount of time	If you want to ignore jobs that are running for certain length of time, specify the running time for jobs that will be ignored. The default is 30 seconds.
Grace period	Specify a number to represent the grace period for job execution. The grace period is a multiple of the average time a job takes to execute. The agent stores a list of the average times jobs take to execute. For example, if you specified a grace period of 5, this script would take that value and multiply it by the average time a job takes to execute. If a job took one second to execute on average, the grace period would be 5 seconds. If the job takes longer than 5 seconds, the script raises an event. The default grace period is 5.
Event severity when jobs take longer than average to execute	Specify the event severity, from 1 to 40, to indicate the importance of an event in which the execution time for Knowledge Script jobs is longer than the average execution time for that job. The default is 5.
Raise an event if jobs take longer than their schedule to execute?	Select **Yes** to raise an event that lists all jobs that are taking longer to execute than their scheduled time to execute. The scheduled time is how often the job is set to run, such as every five minutes. The default is Yes.
Event severity when jobs take longer than their schedule to execute	Specify the event severity, from 1 to 40, to indicate the importance of an event in which the execution time for a Knowledge Script job is longer than the script's schedule. The default is 5.
Raise an event if job exceeds maximum job run time?	Select **Yes** to raise an event if the run time for a Knowledge Script job exceeds the *Maximum job run time* threshold. The default is Yes.
List of Knowledge Scripts to skip "Maximum job run time" check	Provide a comma-separated list of the Knowledge Scripts that you do not want to compare to the *Maximum job run time* threshold.
Maximum job run time	Specify the maximum number of seconds a Knowledge Script job can run before an event is raised. The default is 180 seconds.
Event severity when job exceeds maximum job run time	Set the event severity, from 1 to 40, to indicate the importance of an event in which the run time for a Knowledge Script job exceeds the *Maximum job run time* threshold. The default is 5.
Raise an event if no jobs found?	Select **Yes** to raise an event if no Knowledge Script jobs are running. The default is No.
Event severity when no jobs found	Set the event severity level, from 1 to 40, to indicate the importance of an event in which no Knowledge Script jobs are running. The default is 35.
Timeout when processing jobs	Specify how long AppManager should wait for the agent to process jobs before assuming the agent either will not respond or has timed out. Use this parameter to monitor agents that are consistently taking longer than expected to respond. If the agent doesn't respond before your specified timeout value, AppManager raises an event stating that it was unable to process this command and suggesting you increase the timeout value. The event might also include data about a Windows error, if one was generated. The default timeout is 10 seconds.

Parameter	How to Set It
Use XML formatting in event message?	Select **Yes** to format event messages in XML. The default is Yes. Leave this parameter No to format event detail messages in plain text. Events formatted in XML display results in tables. Events in plain text display results in rows of unformatted text.
Knowledge Script Options	
Event severity when unexpected error occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which this Knowledge Script job fails or any other unexpected event occurs. The default is 35.
Heartbeat Investigation Steps (Used by Management Server)	AppManager performs the following steps only if the heartbeat event or the heartbeat data is missing.
Attempt to contact agent computer by ICMP ping?	Select **Yes** to send an ICMP ping request to the agent computer. The default is Yes.
Perform tracert diagnostic if ICMP ping fails?	Select **Yes** to run a tracert (traceroute) diagnostic test if the ping request fails. The default is Yes. A traceroute test helps you troubleshoot network routing problems that can block ICMP traffic. This script raises an event if the tracert fails.
Connect to agent NetIQmc port if ICMP ping succeeds?	Select **Yes** to attempt a connection to the `NetIQmc` port on the agent computer. The default is Yes. The connection is attempted only if the ping attempt succeeds. This script raises an event if the ping fails.
Use RPC to probe agent if port check succeeds?	Select **Yes** to send a Remote Procedure Call (RPC) to the agent computer. The default is Yes. The RPC is sent only if the port connection succeeds. This script raises an event if the RPC probe fails.
Test agent computer registry if RPC probe succeeds?	Select **Yes** to allow the management server to attempt to use the Remote Registry Service to connect to the Windows Registry on the agent computer. The connection is attempted only if the RPC probe succeeds. The management server must have sufficient privileges to connect to the Registry. The default is No. This script raises an event if the management server cannot connect to the Registry.
Check status of agent services if registry test succeeds?	Select **Yes** to allow the management server to verify whether the NetIQ agent services, `NetIQccm` and `NetIQmc`, are running. This test is attempted only if the registry test succeeds. The management server must have sufficient privileges to access the agent services. The default is No. This script raises an event if the agent services are up or down.

QDBComponentsHealth

Use this Knowledge Script to monitor the health of NetIQ AppManager repository (QDB) and Management Server components.

This script monitors SQL Server resources associated with the QDB, including the percentage of database space and log space used, the time taken for a SQL command or query to execute, and the status of AppManager scheduled tasks. If a service or job is down, this script can restart it

Note
If the QDB does not have an AppManager agent installed on it, Discovery_AMHealth discovers the QDB components on the Management Server. As a result, the service and database monitoring parameters for this script run remotely. In this situation, you must have sufficient privileges on the service account for the NetIQMC service on the Management Server so that the service account can remotely access the SQL Server service on the QDB to obtain its status. If the account does not have proper privileges, the script will be unable to access the service status and will report that the SQL Server service is down even when it is not. If you do not have sufficient access for the service account for the NetIQMC service, deselect the *Raise an event if SQL Server services are down* and *Restart SQL Server services that stop unexpectedly* parameters in this script to avoid raising unnecessary events.

If you want to use QDBComponentsHealth to monitor a Management Server that is not in the same domain or trusted domain as the QDB, see "Monitoring the Health of a Management Server in an Untrusted Domain" on page 242.

You can set this script to raise an event for the following conditions:

- SQL Server services are down or have been restarted
- SQL Server agent jobs are disabled, missing, or have failed
- Database or log space is low
- The Management Server isn't connected to the QDB database
- The Management Server service is down
- SQL Server queries against the QDB are taking too long to process
- Database or log space is low, and there is insufficient disk space for further growth

When monitoring Management Server performance, QDBComponentsHealth does not use the standard method of creating events through the Management Server, because the problem being detected might prevent events from being generated through the Management Server. As a result, this script generates events for these conditions directly in the QDB, and Action scripts will not operate with certain QDBComponentsHealth parameters to generate actions when the conditions occur. Because these events are generated directly in the QDB, event collapsing is always enabled for these events, and it cannot be turned off.

The following QDBComponentsHealth parameters related to Management Server monitoring will not generate actions:

- Raise an event if the Management Server service is down?
- Raise an event if the Management Server service fails to restart?
- Raise an event if the Management Server service restarts successfully?
- Raise an event if the Management Server service is not connected to the QDB?
- Raise an event if data map file usage exceeds threshold?
- Raise an event if event map file usage exceeds threshold?
- Raise an event if job map file usage exceeds threshold?

If you do not have an agent installed on the QDB server, the repository component gets discovered on the Management Server. If you try to remotely monitor the QDB from the Management Server using the QDBComponentsHealth script, the script will not be able to obtain the disk information remotely.

As a result, if the repository component is monitored remotely from the Management Server by the QDBComponentsHealth script, the following QDB component monitoring parameters under the *SQL Server File Size and Growth Settings Monitoring* Event Notification Knowledge Script section will not be available:

- Raise an event if insufficient space available for further file growth?
- Raise an event if SQL Server maximum file size exceeds available disk space?

Resource Objects

- AppManager repository (QDB) server
- Management Server

Default Schedule

The default interval for this script is every thirty minutes.

Setting Parameter Values

Set the following parameters as needed:

Parameter	How to Set It
Event Notification	
Raise an event if SQL Server services are down?	Select **Yes** to raise an event if SQL Server services are down. The default is Yes. **Tip** Use Action Script for notification, as the QDB will not be available.
Event severity when a SQL Server service is down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a SQL Server service is down. The default is 5.
Restart SQL Server services if the services are stopped unexpectedly?	Select **Yes** to restart the SQL Server services if the services stop unexpectedly. The default is Yes.
Raise an event if a SQL Server service restart fails?	Select **Yes** to raise an event if a SQL Server service fails to restart. The default is Yes. **Tip** Use Action Script for notification, as the QDB will not be available.

Parameter	How to Set It
Event severity if a SQL Server service fails to restart	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a SQL Server service fails to restart. The default is 5.
Raise an event if a SQL Server service restart succeeds?	Select **Yes** to raise an event if a SQL Server service restart succeeds. The default is Yes.
Event severity if a SQL Server service restarts successfully	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a SQL Server service restarts successfully. The default is 30.
Raise an event if the Management Server service is not connected to the QDB?	Select **Yes** to raise an event if the Management Server service is not connected to the QDB. The default is Yes.
Event severity when the Management Server service is not connected	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the Management Server service is not connected. The default is 5.
Raise an event if the QDB SQL Server agent jobs are missing?	Select **Yes** to raise an event if the QDB SQL Server agent jobs are missing. The default is Yes.
Event severity for missing QDB SQL Server agent jobs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which QDB SQL Server agent jobs are missing. The default is 15.
Raise an event if QDB SQL Server agent jobs are disabled?	Select **Yes** to raise an event if QDB SQL Server agent jobs are disabled. The default is Yes.
Event severity for disabled QDB SQL Server agent jobs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which QDB SQL Server agent jobs are disabled. The default is 15.
Enable QDB SQL Server agent jobs if they are disabled?	Select **Yes** to enable QDB SQL Server agent jobs if they are disabled The default is No.
Event severity when the attempt to enable QDB SQL Server agent job fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which an attempt to enable QDB SQL Server agent job fails. The default is 15.
Raise an event if attempt to enable QDB SQL Server agent job succeeds?	Select **Yes** to raise an event if the attempt to enable the QDB SQL Server agent job succeeds. The default is No.
Event severity when attempt to enable QDB SQL Server agent job succeeds	Set the event severity level, from 1 to 40, to indicate the importance of an event in which an attempt to enable the QDB SQL Server agent job succeeds. The default is 20.
Raise an event if a QDB SQL Server agent job fails?	Select **Yes** to raise an event if a QDB SQL Server agent job fails**.** The default is Yes.
Event severity when a QDB SQL Server job fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a QDB SQL Server job fails. The default is 20.
Raise an event if the query process time exceeds threshold?	Select **Yes** to raise an event if the query process time exceeds threshold. The default is Yes.
Threshold -- Maximum process run time	Specify the longest process run time allowed before an event is raised. The default is 300 seconds.
Event severity when query process time exceeds threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the query process time exceeds the threshold you set. The default is 5.
Event severity when attempt to retrieve query process time fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the attempt to retrieve query process time fails. The default is 10.

Parameter	How to Set It
Raise an event if unable to retrieve QDB component information?	Select **Yes** to raise an event if AppManager is unable to retrieve QDB component information. The default is Yes.
Event severity when unable to connect to retrieve QDB component information	Set the event severity level, from 1 to 40, to indicate the importance of an event in which AppManager is unable to connect to retrieve QDB component information. The default is 10.
Management Server Performance Monitoring	
Raise an event if the Management Server map file is not enabled?	Select **Yes** to raise an event if the Management Server map file is not enabled. The default is Yes.
Threshold -- Current map file size	Specify the current map file size. If the map file size is too small, the Management Server performance will not be optimal in larger environments. The default is 5 MB.
Event severity if map file is not enabled	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the map file is not enabled. The default is 15.
Raise an event if data map file usage exceeds threshold?	Select **Yes** to raise an event if the data map file usage exceeds the threshold you set. The default is Yes.
Threshold -- Maximum data map file usage	Specify the highest level of data map file usage that can occur before an event is raised. The default is 80%.
Event severity when data map file usage exceeds the threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the data map file usage exceeds the threshold you set. The default is 15.
Raise an event if event map file usage exceeds threshold?	Select **Yes** to raise an event if the event map file usage exceeds the threshold you set. The default is Yes.
Threshold -- Maximum event map file utilization	Specify the highest level of event map file utilization that can occur before an event is raised. The default is 80%.
Event severity when event map file utilization exceeds threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the event map file utilization exceeds the threshold you set. The default is 15.
Raise an event if job map file usage exceeds threshold?	Select **Yes** to raise an event if the job map file usage exceeds the threshold you set. The default is Yes.
Threshold -- Maximum job map file utilization	Specify the highest level of job map file utilization before an event is raised. The default is 80%.
Event severity when job map file usage exceeds threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the job map file usage exceeds the threshold you set. The default is 15.
Monitoring SQL Server File Size and Growth Settings	
Raise an event if insufficient space available for further file growth?	Select **Yes** to raise an event if there is not enough space for additional file growth on the SQL Server. The default is Yes.
Event severity for insufficient space	Set the event severity level, from 1 to 40, to indicate the importance of an event in which there is not enough space for additional file growth on the SQL Server. The default is 10.
Raise an event if SQL Server file growth rate is lower than the threshold?	Select **Yes** to raise an event if the SQL Server file growth rate is lower than the threshold you set. The default is Yes.
Threshold -- Minimum growth rate in MB	Specify the lowest possible growth rate for the SQL Server in MB. The default is 256 MB.
Threshold -- Minimum growth rate as a percentage	Specify the lowest possible growth rate for the SQL Server as a percentage. The default is 9%.

Parameter	How to Set It
Event severity when file growth rate is lower than the threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the file growth rate is lower than the threshold you set. The default is 10.
Raise an event if Autogrowth is not enabled and file utilization exceeds the threshold?	Select **Yes** to raise an event if Autogrowth is not enabled, and file utilization exceeds the threshold you set. The default is Yes.
Threshold -- Maximum file utilization with Autogrowth disabled	Specify the highest percentage of file utilization with Autogrowth disabled before an event is raised. The default is 90%.
Event severity if Autogrowth disabled and usage exceeds threshold	Set the event severity level, from 1 to 40, to indicate the importance of an event in which usage exceeds the threshold and Autogrowth is disabled. The default is 10.
Raise an event if SQL Server maximum file size exceeds available disk space?	Select **Yes** to raise an event if the SQL Server maximum file size exceeds the available disk space. The default is Yes.
Event severity when SQL Server file size exceeds disk space	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the SQL Server file size exceeds the available disk space. The default is 15.
Monitoring the Management Server Service	
Raise an event if the Management Server service is down?	Select **Yes** to raise an event if the Management Server service is down. The default is Yes. **Note** AppManager cannot create an event for this if the Management Server is down.
Event severity if the Management Server service is down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the Management Server service is down The default is 10.
Restart the Management Server service if the service is down?	Select **Yes** if you want to restart the Management Server service if the service is down. The default is Yes.
Raise an event if the Management Server service restarts successfully?	Select **Yes** to raise an event if the Management Server service restarts successfully. The default is Yes. **Note** AppManager cannot create an event for this if the Management Server is down.
Event severity if the Management Server service restarts successfully	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the Management Server service restarts successfully. The default is 25.
Raise an event if the Management Server service fails to restart?	Select **Yes** to raise an event if the Management Server service fails to restart. The default is Yes. **Note** AppManager creates an event directly in the QDB if the Management Server fails to restart.
Event severity if the Management Server service fails to restart	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the Management Server service fails to restart. The default is 5.
Raise an event if agents are not FIPS- compliant?	Select **Yes** to raise an event if the agents are not FIPS-compliant. The default is Yes.
Event severity when agents are not FIPS-compliant	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the agents are not FIPS-compliant. The default is 10.
Data Collection	
Collect data for database space utilization?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns the percentage of the database data file currently being used. The default is No.

Parameter	How to Set It
Collect data for log space utilization?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns the percentage of the database log file currently being used. The default is No.
Collect data for Management Server connection status?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns 100 if the Management Server is connected to the repository and 0 if the Management Server is not connected. The default is No.
Collect data for SQL Server service status?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns 100 if the SQL Server service is up and 0 if the service is down. The default is No.
Collect data for Management Server data map file utilization?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns the percentage of the Management Server data map file currently being utilized. The default is No.
Collect data for Management Server event map file utilization?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns the percentage of the Management Server event map file currently being utilized. The default is No.
Collect data for Management Server job map file utilization?	Select **Yes** to collect data for charts and reports. If enabled, data collection returns the percentage of the Management Server job map file currently being utilized. The default is No.
Monitoring	
SQL username	Specify the user name required to access SQL Server on the Control Center server. **Note** If you are running this script on a Management Server in an untrusted domain, type the SQL user name you used in Step 6 of the procedure for Monitoring the Health of a Management Server in an Untrusted Domain.
Event severity for a Knowledge Script error	Set the event severity level, from 1 to 40, to indicate the importance of an event in which an unexpected Knowledge Script error occurs. The default is 10.

Recommended Knowledge Script Groups

You can find the AMHealth Knowledge Script Groups (KSGs) on the RECOMMENDED tab of the Knowledge Script pane in the Operator Console.

All the scripts in the KSGs have their parameters set to recommended values. To run all of the recommended scripts in a KSG at one time, click the RECOMMENDED tab, and then run the KSG on an AppManager resource.

The AMHealth KSGs enable a "best practices" usage for monitoring your AppManager environment. You can use these KSGs with AppManager monitoring policies. A monitoring policy lets you efficiently and consistently monitor all the resources in your environment using a set of pre-configured Knowledge Scripts. For more information, see "About Policy-Based Monitoring" in the AppManager Help.

A KSG is composed of a subset of a module's Knowledge Scripts. The script that belongs to a KSG is a different copy of the original script you access from the AMHealth tab. If you modify a script that belongs to a KSG, the parameter settings of the original script in the AMHealth tab are not affected.

In some cases, default script parameter settings are different when the script is deployed as part of a KSG, as opposed to when it is deployed alone. The default settings of a script within a group depend on its monitoring purpose within the larger group, and on the intended monitoring scope of that group.

If you modify or remove a script associated with the AMHealth KSG and want to restore it to its original form, you can reinstall the AppManager Self Monitoring module on the repository computer or check in the appropriate script from the `AppManager\qdb\kp\AMHealth` directory.

HealthCheckAMAgentComponents Recommended KSG

The following Knowledge Scripts are members of the HealthCheckAMAgentComponents recommended KSG:

- HeartbeatUNIX
- HeartbeatWin

HealthCheckAMCoreComponents Recommended KSG

The following Knowledge Scripts are members of the HealthCheckAMCoreComponents recommended KSG:

- CCComponentsHealth
- QDBComponentsHealth