Planning and Configuration Guide for AppManager

A NetIQ Work Smarter Guide

NetIQ Corporation
5410 Betsy Ross Drive
Santa Clara ■ CA ■ 95054
www.netiq.com



IMPORTANT NOTICE

DISCLAIMER OF WARRANTY

NetIQ Corporation makes no representation or warranties, either express or implied, by or with respect to anything in this manual, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special, or consequential damages.

COPYRIGHT NOTICE

This document and the software described in this document is furnished under a license agreement or a non-disclosure agreement and may be used only in accordance with the terms of the agreement. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical or otherwise, without the prior written consent of NetIQ Corporation. This publication and the features described herein are subject to change without notice. Companies, names and data used in this publication are fictitious unless otherwise noted.

Copyright © 1995-2000 NetIQ Corporation. All rights reserved.

Portions of the software described in this document copyright © 1992-1998 Summit Software Company. All rights reserved.

U. S. Government Restricted Rights: Use, duplication, or disclosure by the Government is subject to the restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause of the DFARs 252.227-7013 and FAR 52.227-29(c) and any successor rules or regulations.

TRADEMARKS

NetIQ, AppManager, AppManager Suite, Knowledge Scripts, and associated logos are registered trademarks or trademarks of NetIQ Corporation in the United States and/or other countries.

Microsoft, Windows, Windows NT, and BackOffice are registered trademarks of Microsoft Corporation. All other brands, products, or services mentioned are trademarks, registered trademarks, service marks, or product names as designated by their respective companies.

Trend Micro and ScanMail are Trend trademarks.

Contents

About this guide

Intended audience 9
What version of the software do you use? 10
Using this guide 10
Conventions used in this guide. 11
Where to go for more information. 11
Questions or suggestions? Contact us.... 12

Chapter 1 Introduction

What are the AppManager components? 14
Communication between AppManager components 17
What happens on the managed client. 17
How the management server and repository are used 19
What is the impact on planning and configuration? . 19
Developing a project plan 20
Assembling a project team 20
Defining project phases. 22
Establishing a time line 23

Chapter 2 Planning deployment

Evaluating the environment to be monitored 27
Reviewing your network configuration 29
Network connectivity 31
Planning management sites 32

Determining where to locate the management server 34
Checking system requirements 35
Planning system resource usage for AppManager . 36
Resources for a managed client 36
Resources for a management server 37
Resources for an Operator Console 38
Sizing the AppManager repository 38
Estimating with a management scenario 39
Estimating without a management scenario . . 41
Adjusting the size of other databases 41
Planning security 42
Planning your management strategy 43
Identifying management goals 44
Selecting Knowledge Scripts and setting thresholds 44
Planning event handling and response 45
Planning data handling and reporting 46
Planning for customization 47
Documenting your decisions and policies 48

Chapter 3 Staging the deployment

Installing in a lab environment 49
Preparing to install the pilot group 50
Running the recommended core Knowledge Scripts 52
Setting thresholds for the recommended scripts 56
What you should see 57
Using the data collected 58
Adjusting thresholds and intervals 59
Moving to the next stage of deployment 60

Expanding the scope of your deployment 60
Deploying additional Knowledge Scripts 61
Identifying your reporting requirements. 62
Expanding the use of actions and notification policies 63
Reviewing and refining the deployment 63
Moving beyond deployment to extend AppManager 64
Roadmap for a staged deployment 65

Chapter 4 Managing events

Setting consecutive event intervals 67
Collapsing duplicate events 71
Setting the event collapsing interval 71
Setting consecutive occurences to ignore duplicate events 72
Event collapsing scenarios 72
Determining the event collapsing rules to use 76
Selectively disabling events for a managed client . . 77
Disabling events raised by specific Knowledge Scripts 78
Using a custom severity value 79
How to set the registry key value 80
Automating event notification 81
Sending an email message. 81
Forwarding an event to a paging system 83
Forwarding an event to another console. 86
Automating corrective action 87

Chapter 5 Managing data collection

Deciding when to collect data. 91
Setting the maximum number of data points 92

Purging older data points 94
Editing the stored procedure parameters for PurgeData 95
Archiving data for historical reporting 97
Should you archive data? 99

Chapter 6 Configuring site communication

Authorizing a management server to communicate with a managed client 101
Changing the authorization list 102
Changing the name of a management server 103
Configuring network communication 103
Using an IP address or hostname 103
Controlling the flow of network traffic 104
Configuring a local repository 105
Configuring and scheduling data transfers 105
Scheduling maintenance periods 107
Temporarily blocking jobs 107
Setting a resource dependency 108

Chapter 7 Managing security

Understanding the layers of security 109
Understanding user access to AppManager 112
Setting up AppManager users 113
Adding existing SQL Servers users to AppManager 113
Adding new SQL Server users from the Security Manager 114
Adding SQL Server users when using Integrated security 115
Understanding AppManager roles 115

Special administrative privileges 118
Storing security information for Knowledge Scripts 119
Setting the SNMP community string 121
Setting Exchange profile and mailbox names 122
Setting Oracle user names and passwords . 122
Setting a configuration name for SAP Knowledge Scripts 122
Setting the SMS SQL Server login 122
Setting SQL Server logins and passwords . . 123

Chapter 8 Maintaining the repository

How AppManager uses SQL Server 125
What are the key tables 125
What stored procedures do 126
What scheduled tasks do 127
Checking your SQL Server configuration 128
Tuning configuration options 128
Configuring network communication for SQL Server 131
Creating data devices for expanding the database 134
Backing up the repository 135
Backing up the repository using SQL Enterprise Manager 135
Backing up the repository with SQL_RunSql 136
Backing up tasks and stored procedures . . . 137
Backing up the master database 137
Scheduling backups 138
Restoring a repository from a backup 138
Removing archived data and old events 141
Expanding the size of an existing repository 142

Expanding the current database device 143
Expanding into a new database device 144
Verifying the size of the AppManager repository 144
Migrating the repository to another SQL Server. . . 145
Restarting the management server 147
Restoring database triggers after migration . 147
Restoring SQL users after migration 147
Checking the integrity of the repository 148
Setting permissions for stored procedures. 149

Chapter 9 Samples and worksheets

Simplified sample project plan 151
Sample inventory worksheet 153
Required permissions worksheet. 153
Sample management worksheet 156

Appendix A Choosing Knowledge Scripts

Windows NT operating system 160
NetServices 161
Windows Internet Name Service (WINS) . . . 161
Dynamic Host Configuration Protocol (DHCP) 161
Remote Access Service (RAS). 162
Domain Name Service (DNS). 162
Microsoft Exchange Server 162
Microsoft SQL Server. 163
Microsoft Systems Management Server (SMS) . . . 164
AppManager Admin (AMAdmin) 165

Appendix B **Registry keys**

Basic NetIQ keys 168
AppManager subkey 169
Generic\Tracing subkey 172
Managed client registry keys 172
AgtLocalRP subkey 173
NetIQccm subkeys 173
NetIQmc subkeys 175
Management server registry keys 181
AppManager repository registry keys 185
Web management server registry keys 186

Index

About this guide

The NetIQ® AppManager Suite (AppManager®) is a comprehensive solution for managing and monitoring the performance and availability of distributed Microsoft Windows NT and Windows 2000 servers and server applications.

AppManager enables system administrators to view all of their Windows NT and Windows 2000 systems from a central, easy-to-use console and to monitor computer resources and gather performance data for real-time and historical reporting.

Intended audience

The *Planning and Configuration Guide for AppManager* is intended for system managers, network administrators, and implementation team members preparing to install, configure, and deploy AppManager.

This guide provides recommendations and guidelines for planning a successful AppManager deployment, and suggests strategies for tuning AppManager to your organization's needs. However, because each organization's requirements and environment may vary, these suggestions and guidelines should be viewed as examples to help you make decisions appropriate to your specific implementation.

This guide assumes that you are familiar with your operating system and the applications you intend to monitor. This guide also assumes you have a working knowledge of or access to other documentation for performing basic Windows NT and AppManager activities (for example, you should be familiar with AppManager components and terminology, starting AppManager jobs, and using the Windows User Manager for Domains and Control Panel programs).

What version of the software do you use?

This guide assumes a 3.4 version of AppManager. Although the general recommendations and planning activities may apply to other versions of the software, specific tuning and sizing recommendations are based on version 3.4. If you have a different version, contact NetIQ Technical Support to see if updated information is available. For version-specific information, see the README web page.

Using this guide

Depending on the applications and resources you want to monitor and your environment, you may want to read portions of this guide selectively.

- Chapter 1, "Introduction," reviews the AppManager architecture and the interaction between AppManager components.
- Chapter 2, "Planning deployment," guides you through the process of developing a project plan and and what you need to consider in planning your implementation.
- Chapter 3, "Staging the deployment," describes what to expect in a staged roll-out of AppManager, including a brief highlight of the key tasks undertaken in each stage.
- Chapter 4, "Managing events," reviews some of the options for configuring event handling and notification.
- Chapter 5, "Managing data collection," discusses the options for data handling and archiving.
- Chapter 6, "Configuring site communication," illustrates some of the options for configuring the communication between AppManager components.
- Chapter 7, "Managing security," describes the security mechanisms you should be familiar with to manage AppManager user access.

- Chapter 8, “Maintaining the repository,” provides guidelines for managing and performing ongoing maintenance for the AppManager repository.
- Chapter 9, “Samples and worksheets,” provides some sample resources you may find useful as models for planning the deployment and installing AppManager.
- Appendix A, “Choosing Knowledge Scripts,” provides a summary of the recommended and suggested Knowledge Scripts for various Knowledge Script categories.
- Appendix B, “Registry keys,” summarizes the registry keys used by AppManager components.

In addition to these chapters and appendices, an index is provided for your reference.

Conventions used in this guide

The following conventions are used in this guide:

- `Courier` font is used to highlight source code, program names or output, file names, and commands that you enter.
- Angle brackets < > are used to indicate variables.
- **Bold text** is used to emphasize commands, buttons, or user interface text, and to introduce new terms.
- SMALLCAPS are used for keyboard keys, for example RETURN and ESC.

Where to go for more information

The AppManager documentation set includes several sources of information. Depending on your interests, you may want to explore some or all of these sources further:

- *NetIQ AppManager Installation Guid*e for complete instructions on installing and configuring AppManager.

- *NetIQ AppManager User's Guide* for complete information about running jobs, responding to events, creating reports, and working with all of the AppManager consoles.
- *NetIQ AppManager Knowledge Scripts Reference Guide* for a description of what each Knowledge Script does and details about setting job parameters.
- *NetIQ AppManager Developer's Guide* for an introduction to writing your own custom Knowledge Scripts.
- *NetIQ AppManager Managed Objects Reference Guide* for technical information about the most commonly used AppManager managed objects. (This guide is provided in online format only, and does not document all AppManager managed objects.)

In addition to the AppManager documentation, you may want to consult Windows NT, BackOffice, Domino and other systems documentation for reference and conceptual information. This background information can help you get the most out of your AppManager installation.

Questions or suggestions? Contact us....

For technical support or suggestions for improving AppManager, visit the NetIQ web site **www.netiq.com**, contact NetIQ Technical Support directly at **support@netiq.com**, or send your comments to NetIQ at the address on the back of this guide.

For information about purchasing AppManager Suite products, contact NetIQ Sales at **info@netiq.com**.

For comments or suggestions regarding the documentation and online help, contact us at **documentation@netiq.com**.

Introduction

AppManager is a comprehensive solution for monitoring the performance and availability of distributed Windows NT systems and server applications. With AppManager, system administrators can:

- Set thresholds and event severity notification levels as needed.
- Continuously monitor computer resources with little or no manual intervention.
- View complete application configuration details.
- Stay informed about potential and current system problems.
- Initiate automated responsive or corrective actions.
- Gather and view performance data for real-time and historical reporting.
- Extend monitoring functionality with easy-to-use, Visual Basic for Applications (VBA) scripting tools.

To effectively plan the installation and configuration of AppManager for a production environment, you first need to understand how AppManager works and the key components you need to install and configure. This chapter provides an introduction to AppManager's architecture and an overview of how AppManager components work together—and independently of each other—to monitor your environment.

Note that only key concepts and the interaction between core components are covered here. For a more complete discussion of AppManager architecture, see the *NetIQ AppManager User's Guide*.

What are the AppManager components?

AppManager's flexible, multi-tier architecture consists of several required and optional components. The components can be installed together on a single computer or separately on multiple computers. Each component has unique requirements or configuration options.

The **required components** are:

- At least one **AppManager Operator Console**. An Operator Console is used to view and control the jobs that monitor and manage your computers and server applications. There are three Operator Console environments: the standard Windows interface *Operator Console*, the web-browser based *Operator Web Console*, and the MMC snap-in *Operator Console MMC Snap-in*.
- The **AppManager agent**. The AppManager agent actually consists of two Windows NT services (`netiqmc` and `netiqccm`), a local repository database, and at least one managed object that reside locally on each computer you are managing. This component is responsible for monitoring the managed client's system and application resources (such as MTA queue length, CPU utilization, and internal fan conditions).
- A **Management server**. The management server is a Windows NT service (`netiqms`) that manages the event-driven communication between the repository and the AppManager agents.
- The **AppManager repository**. The repository is a SQL Server database that stobres all of your management information.

The optional components are:

- The **AppManager Web management server**. This component is required if you want to use the Operator Web Console to check the status of jobs and events, create and

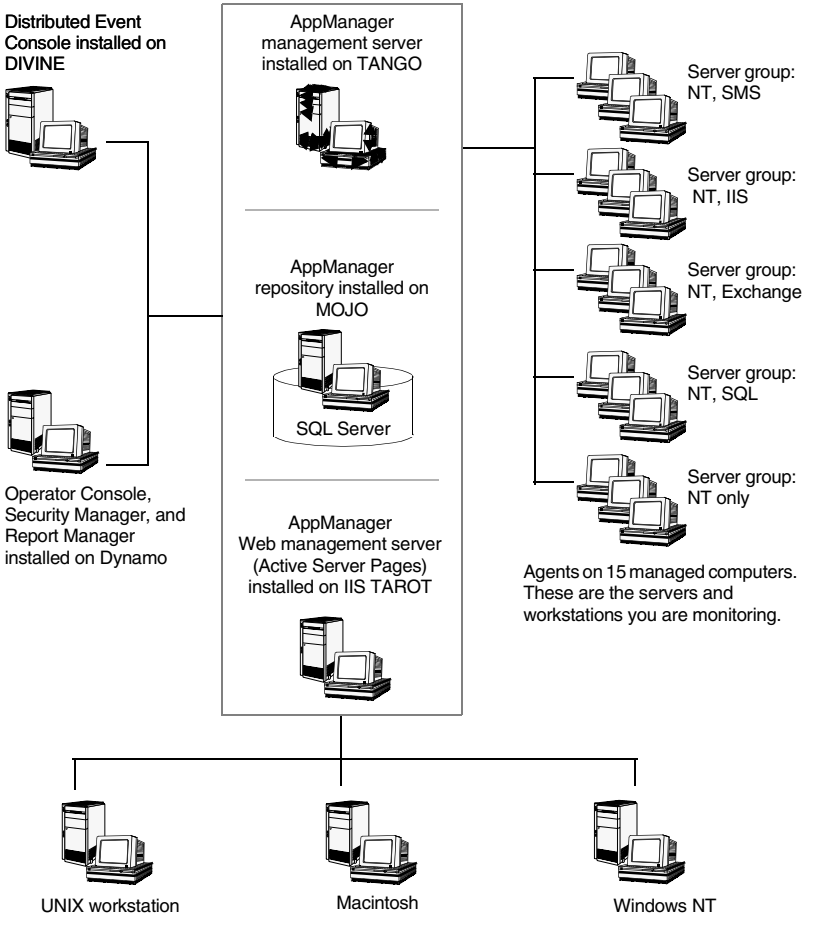
view graphs, run reports, run Knowledge Script jobs, and view details about the computers you are monitoring using a web browser.

- The **Security Manager** allows you to manage security for an AppManager site. Technically, the Security Manager is not a required component because the SQL Server `sa` user has access to the AppManager repository even if you don't install this component. However, the Security Manager is required to give any other SQL login accounts permission to use AppManager.
- The **Report Manager** provides pre-designed reports for accessing and analyzing data in the AppManager repository.
- The **Distributed Event Console** allows you to view event information on one or more AppManager repositories.
- The **Developer's Console Utilities** provide tools for developing custom Knowledge Scripts.

When you run the AppManager Setup program, you select which components you want to install. As part of the planning process, you need to decide where to install these different components and how many of each component you need to effectively manage your environment.

As an example, if the administrators who need to be able to view and respond to events are widely distributed, you may want to provide Web Console access for the remote administrators; if you have support or Help Desk operators who need a summary of open events across repositories but don't actually respond to the events or run or stop jobs, you may want to provide this access through a Distributed Event Console.

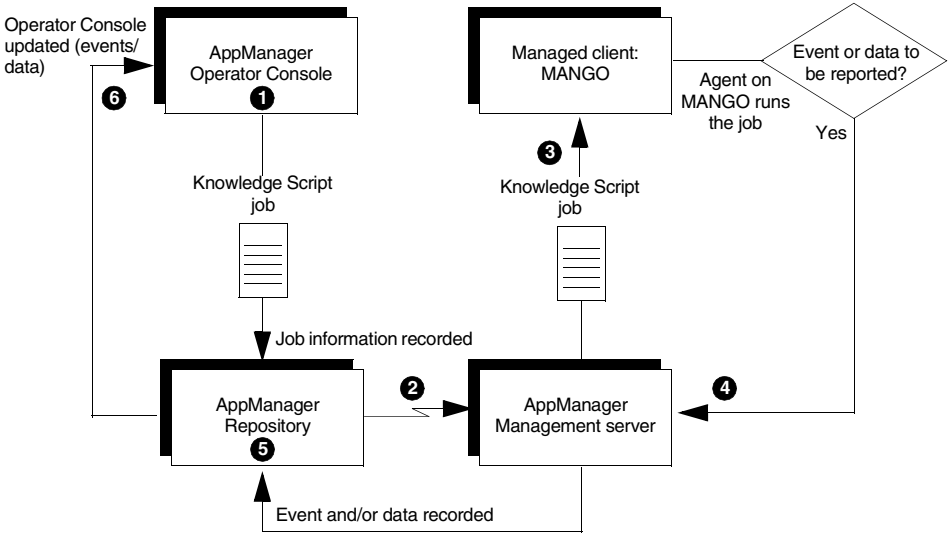
The following figure illustrates a simplified installation with AppManager components installed across 20 computers.



These computers access the ASP on TAROT and act as Operator Web Consoles. The HTML output is available on virtually any computer with Microsoft Internet Explorer or Netscape Navigator.

Communication between AppManager components

To help you plan the deployment of AppManager, you need to understand the communication between AppManager components and what occurs on a managed client. The basic flow is illustrated in the following diagram:



What happens on the managed client

Any Windows NT or Windows 2000 server or workstation on which you install an **AppManager agent** becomes a **managed client** that you can monitor. When you start a job, you indicate the managed clients on which you want the job to run. The AppManager agent service `NetIQmc` receives the request from the management server and runs the job locally.

When `NetIQmc` runs a job, it interprets the VBA code in the Knowledge Script. Most Knowledge Scripts include an OLE automation call (`CreateObject`) to call one or more of the AppManager **managed objects**.

A managed object is a collection of lower-level COM/OLE objects, packaged in a DLL. There is a DLL that contains all of the COM/OLE objects for managing Windows NT, another DLL that contains all of the COM/OLE objects for managing Exchange, and so on for each server or application you can monitor with AppManager. The appropriate DLLs are copied to the managed client during setup when you select what you want to monitor.

Within each DLL, the lower-level COM/OLE objects use counters, event logs, queries, and other sources to gather statistics and monitor specific elements and properties of each application.

When a Knowledge Script calls a managed object, the appropriate DLL is loaded into the system memory. The COM/OLE object collects whatever raw statistics or information it needs, such as the current CPU utilization or database lock activity, and passes that information to the Knowledge Script job. The Knowledge Script evaluates the raw information to determine if an event condition exists or what should be done with the data.

If the Knowledge Script logic evaluates the raw information and determines that an event has occurred or data needs to be inserted into the repository, the `NetIQmc` service notifies the `NetIQccm` service, and the `NetIQccm` service communicates with the management server to upload the relevant information. If the `NetIQccm` service can't communicate with the management server for any reason (for example, because the network connection is down or you have intentionally prevented communication), the `NetIQccm` service writes the data to the local managed client repository. When connectivity is reestablished, the `NetIQccm` service uploads any data stored locally to the management server.

How the management server and repository are used

The management server is a Windows NT service, `NetIQms`, that manages the event-driven communication between the repository and the managed clients.

The AppManager repository server is a central repository based on Microsoft SQL Server that stores management data for a site.

When you use the Operator Console or Operator Web Console to start a job, a record is inserted into the AppManager repository's database. An update of this nature is interpreted as an instruction for the management server to communicate the Knowledge Script and its properties to the appropriate managed clients.

The management server also receives communication back from the managed clients in the form of performance and event information returned by the Knowledge Scripts jobs. The management server inserts this information into the AppManager repository. The Operator Console polls the repository, notes the new information, and updates its display (for example, a real-time graph gets updated with new data, or a newly detected event begins to flash an icon in the TreeView).

What is the impact on planning and configuration?

Because AppManager uses a multi-tiered architecture, there are several characteristics of your environment to consider before you begin actively planning your implementation. For example, it's important to understand the communication between AppManager components because you need to plan network connectivity and bandwidth usage, the system resources you will need, where best to install different components, and how best to configure each component to suit your organization's requirements.

In addition to understanding how AppManager works, it is essential that you evaluate your current environment and the

scope of your monitoring requirements. This is especially true for larger scale deployments which may require additional research to get a complete picture of your current configuration. Although your initial estimates may need to be refined, it's important for you to have a basic management strategy and a thorough knowledge of your environment to ensure a successful deployment.

Developing a project plan

One of the first steps in a successful deployment of AppManager is developing a project plan that you can use as a roadmap to guide you during implementation. Although a formal project plan may not be required for smaller scale organizations or if you are monitoring a small number of servers in a local area network, most larger deployments with complex management needs benefit from putting a plan in place.

Assembling a project team

In developing your project plan, you need to assemble the project team and identify the key tasks and responsibilities of each member. The size and technical expertise of the team should reflect the size and management requirements of the deployment.

For example, if you are monitoring Exchange Servers, your team should include an Exchange Administrator or individuals with similar experience; if you are monitoring SQL Server, the team should include individuals with database administration experience. All team members should have fairly comprehensive knowledge of Windows NT Server and the network configuration of the organization.

The following table provides guidelines for the information your project team members need to possess:

Area of expertise	Information or access required
Windows NT	■ Domain structure and trust relationships. ■ Admin passwords for all managed clients and the computers where the management server, repository, and consoles are to be installed. ■ Ability to create and modify user accounts.
Network	■ Network bandwidth and topology. ■ Latency and the configuration of switches and routers. ■ Any DNS, WINS, or DHCP setup in use.
SQL Server	■ System administrator (sa) privileges for the repository server and for target managed clients during installation of the SQL managed object DLL, if applicable. ■ Ability to create and modify SQL Server users and logins or knowledge of SQL Server logins with permission to access system tables. ■ Experience with SQL Server security modes. ■ Ability to review and evaluate the hardware configuration for the computer that will serve as the AppManager repository server. ■ Ability to review and evaluate SQL Server configuration options, in particular the configuration of memory, RAM, locks, user connections, tempdb, and log size. ■ Ability to create and modify SQL Server scheduled tasks. ■ Ability to back up and restore a database. ■ Understanding of ongoing database maintenance, such as using dbcc for consistency checking.
Applications	Depending on the applications you monitor, the information needed will vary. Some common recommendations are: ■ For SMS: SMS SQL Server sa password. ■ For Exchange: Ability to create mailboxes, grant view-only access, and turn on tracking logs. ■ For Oracle: System-level (V$ tables) access, ability to create and modify users. ■ For Domino: Notes administrator experience.

Defining project phases

When you have identified the skills and strengths of the project team, you should outline the project phases, the responsibilities of each team member during each phase, and a time line for completing each phase. These phases help to stage the deployment to ensure the greatest success with the least impact on the production environment.

Typically, the project plan involves the following phases:

Planning and design phase: In this phase the project team collects all the necessary information about the environment, analyzes the data, reviews implementation options, identifies organization objectives for what needs to be monitored, and documents decisions.

Testing and evaluation phase: In this phase, the project team needs to create a testing laboratory that mimics (on a smaller scale) the overall environment to be monitored, install AppManager components on one or more computers, and run the jobs identified in the design phase on the computers in the testing lab.

Pilot deployment phase: In this phase, the project team moves the implementation out of the testing lab to a pilot group of servers, and evaluates deployment decisions to see if changes need to be made before a full scale deployment. This stage is important because it provides a realistic picture of how AppManager functions in the live environment and gives the team a chance to fine-tune configuration decisions.

Expanded or full-production deployment phase: In this phase, AppManager is rolled out to the larger environment. In addition, most organizations begin to refine their management and notification policies and tailor AppManager to suit their environment.

Post-deployment phase: In this phase, organizations focus on customizing, extending, and maintaining the AppManager environment.

These phases can serve as general guidelines. However, each deployment is different and your requirements may vary.

In addition, your project plan should identify specific tasks for each phase. See Chapter 9, "Samples and worksheets," for a sample project plan in simplified form.

Establishing a time line

Because most deployments are done through a progression of stages or phases, your project plan should include a time line for the implementation. Depending on the detail in your project plan, your time line might address the period for individual tasks or for completing a stage of the deployment. In most cases, it is easier to establish the schedule for completing each phase than addressing individual tasks; however, each member of the project team should review the tasks to be completed to set a realistic schedule.

To get you started, the sample project plan in Chapter 9, "Samples and worksheets," illustrates the time line for an implementation that has no gating factors (for example, the implementation does not have be coordinated with other departments and the project team has full capability to create any Windows NT or SQL Server accounts it needs).

The following table provides a quick summary of what to expect and some of the factors that may impact the time needed to complete each phase.

Project phase	Time:	Factors that may increase time needed:
Planning and design	3 to 5 days	■ Delays in collecting the necessary information. ■ Availability of key personnel or difficulties in the approval process. ■ Difficulty identifying who needs to be notified of events and how notification should be handled. ■ Difficulty producing required documentation (for example, the time it takes to document the project plan or the scope of the project).
Testing and evaluation	5 days	■ Access to a testing laboratory ■ Permission to create user or database accounts. ■ Training for team members. ■ Changes to management strategy (for example, jobs and event handling)
Pilot deployment	1 to 4 weeks	■ Access to the pilot group of servers. ■ Problems with domain and trust relationships. ■ Problems with network name resolution. ■ Refinement of management strategy. ■ AppManager configuration issues. For example, if an organization has grown dramatically or in unexpected ways, the project team may need to reevaluate decisions such as a centralized management site versus multiple decentralized sites.

Project phase	Time:	Factors that may increase time needed:
Full deployment	1 to 4 weeks	■ Physical location of servers. ■ Department policies and procedures. ■ Problems with domain and trust relationships. ■ Permission to create user or database accounts. ■ Bandwidth or latency issues to be resolved. ■ AppManager configuration issues. For example, if an organization has grown dramatically or in unexpected ways, the project team may need to reevaluate decisions such as a centralized management site versus multiple decentralized sites.
Post-deployment	ongoing	■ Development expertise or interest. ■ Database management expertise.

The length of time required to complete each stage will vary depending on a number of factors, such as the number of servers and applications to be monitored, the physical location of computers, and internal policies and procedures. For example, if you are deploying AppManager in several departments that are located in different countries, each location may need to go through its own design and testing phase or may need the production deployment scheduled with local requirements in mind.

Although this time line illustrates a basic deployment schedule, keep in mind that you continually review your assumptions and expectations throughout the implementation. Mostly likely, you will need to refine your management strategy over several weeks as more servers are monitored and events and data are generated. You may also need to plan for custom Knowledge Script development and testing, which may impact the schedule for implementation. We recommend you carefully evaluate the test installation and the pilot program to determine whether any custom development needs to be done before moving to a full deployment.

With a project plan and a deployment team prepared, you are ready to begin collecting information about your environment. The next chapter provides examples, guidelines, and recommendations to help you plan your implementation based on the specific requirements of your site.

Planning deployment

With a project plan in place and a deployment team prepared to take action, you are ready to begin collecting information about your environment. This chapter provides examples, guidelines, and recommendations to help you plan an implementation of AppManager based on the specific requirements of your site.

Evaluating the environment to be monitored

An essential part of the planning process is understanding the characteristics of your environment and the network you are going to monitor. Before you can decide where to install or how to configure AppManager components, you need to be able to answer several questions about the physical and logical make-up of your network.

To get started, you need to think about the computers you want to monitor and how you plan to use AppManager. The questions you need to answer include:

- Number of Windows NT/Windows 2000 servers to be monitored?
- Number of application servers (such as Exchange Server, Domino server, etc.) to be monitored?
- Number of hardware types to be monitored?

Note that a single computer may fall into more than one category. For example, you may have a Compaq server with Windows NT and Exchange Server on it. As part of your planning you need to decide, for example, if you will monitor NT, CIM, and Exchange on that computer or just NT because this may affect how you size the repository or how you decide to distribute AppManager components.

In addition to the computers you are going to monitor, you need to decide and document how you intend to use AppManager. The questions you need to answer include:

- Who needs to view events? Who will respond to events?
- How are users going to view events and other information? What type of consoles are needed? For example, will some users require web access? dial in to a terminal server? connect through RAS? Should specialized consoles, such as the Report Manager or Security Manager be placed on separate computers that are only accessed by a few users?
- What are your main monitoring objectives? Is your organization only concerned with event monitoring? with a small amount of data collecting? with collecting a great deal of data for reporting and graphing?
- How does the organization intend to handle event notification? Should events trigger a call to a paging system, send an email message, forward an SNMP trap to an external system, or perform some other action?

As with the number of servers and applications to be monitored, this information will affect how you size the repository and how you decide to distribute AppManager components. Your monitoring and event handling goals may also impact the configuration you need to do after you install AppManager.

Reviewing your network configuration

As you evaluate the computers you intend to monitor, you also need to be aware of where those computers are and how they are configured. This is a key part of the data collection you do in the planning and design phase of the project.

The questions you need to answer include:

- What is the physical location of the servers to be monitored?
- What are the bandwidth and latency characteristics of your network? Where are the broad band and narrow band connections and where are the routers and gateways through which network communication flows?
- What is your network load at different points in the day? What are the peak and off-peak times?
- How are server domains and trusts currently defined? (Although you can monitor across trusted and untrusted domains, you should collect information about domains and trusts for installation purposes.)
- Which accounts have Administrator privileges in which domains?

Information about your network topology and its limitations is most likely to affect how you distribute AppManager components. For example, you may decide, based on this information to create two regional management sites instead of one centralized site for network efficiency. This information is also useful for determining how best to group computers for remote agent installation.

To help you record network information in preparation for installation, you may find it useful to create a chart that illustrates

for each computer: what you plan to monitor, domain and trust information, and physical location. For example:

Computer name	What to monitor	Domain and trusts	Location	Disk
Porche	Windows NT Exchange Server	WEST1 domain Trusted by: ■ Enterprise00 Trusts: ■ CAL1 ■ CAL2 ■ WASH1	Portland, OR	
Gremlin	Windows NT Domino	WEST1 domain Trusted by: ■ Enterprise00	San Jose, CA	

Additional columns for disk space or configuration notes may be added (for example, when a server is part of a cluster). You can prepare a similar worksheet for the computers on which you are planning to install other AppManager components, such as the repository and management server.

Tip You may also want to add columns for other important information unrelated to configuration but useful for installation or administrative purposes. For example, you may want to add an "Owner" column to track the person or group who needs to provide you with permission to install, or a "Business-critical" column to highlight the servers that you can only access during non-production hours.

Creating a chart like this can help you see potential troublespots in domain and trust relationships or if you need to collect additional information. For example, if you are monitoring an Exchange Server, you may need to investigate creating a Windows NT user account for the AppManager agent to run as.

In addition, you may find it useful to do this project planning by filling out a worksheet similar to the sample provided in Chapter 9, "Samples and worksheets."

Network connectivity

Reviewing your network configuration should help you determine where to install AppManager components. However, you should also keep in mind that specific types of network connectivity must exist between the computers where specific AppManager components are installed.

In deciding where you are going to install components, check the following network connections (the following checks represent the typical network connections; if you have configured your network differently—for example with ODBC running over RPC—your connectivity points may be different):

- AppManager Operator Console computer and the repository server require TCP/IP and ODBC connectivity.
- AppManager repository and the management server, if installed on separate servers, require TCP/IP and ODBC connectivity.
- AppManager management server and each managed client on which AppManager agent has been installed require TCP/IP and RPC connectivity. In addition, the AppManager management server needs to be able resolve name or IP address for each managed client and each managed clients needs to be able to resolve the name/IP address of the management server.
- AppManager repository and the Web management server, if installed on separate servers, require TCP/IP and ODBC connectivity.
- AppManager Web management server and the computer with a web browser (that acts as the Operator Web Console) require TCP/IP connectivity.
- AppManager components expect to communicate with each other through specific ports. You should check for any port restrictions or firewall protections.

Typically, AppManager uses the following ports for communication between components:

Communication between:	Uses this port:	Protocol:
Operator Console - repository	1433 (if using TCP/IP) 135 (if using Named Pipes)	SQL ODBC
Management server -repository	1433	SQL ODBC
Web management server - repository	1433	SQL ODBC
Web Operator Console - Web management server	80	HTTP
NetIQms service to NetIQmc service	9999	TCP/IP
NetIQmc/NetIQccm to NetIQms	9998	TCP/IP
AMAdmin_AgentInstall (NQEXEC program) to remote directories	9979	TCP/IP
RPC - XCopy (for remote installation scripts, such as AgentInstall)	135	RPC

Planning management sites

As described in "What are the AppManager components?" on page 14, AppManager consists of required and optional components that can all be installed on a single computer or installed separately, in virtually any combination, on multiple computers.

Each component has specific system requirements, so in the planning process you should try to determine where you want to install which components. It is especially important to consider the number and placement of management sites.

A **management site** always consists of one AppManager repository and at least one AppManager management server, regardless of where these components are installed. Every management server has exactly one repository it communicates with. That one-to-one

or many-to-one link is the primary element that defines a management site.

Each management server has one or more managed client computers that run the AppManager agent. Each managed client has one primary management server you identify when you install the AppManager agent. Although managed clients always have one primary management server (and, by extension, one primary repository), they can be monitored independently by other management sites if your organization permits it.

In planning where to install components and how many management sites you'll need, there are several issues to consider:

- When you set AppManager repository preferences for event and data handling, these preferences serve as the defaults for the site (they can be overridden on a job-by-job basis).
- If you are monitoring a large number of managed clients and you have the capability to stagger or schedule communication between managed clients and the management server, you may want to use a single centralized site. If you have a widely distributed environment or have latency or bandwidth restrictions, you may want to set up regional or divisional management sites for increased network efficiency.
- If your organization's internal policies require more localized operational control over parts of the network and who has access, you may want to set up separate management sites to provide this control.
- If you have a large number of jobs and expect to generate a large number of events or collect a great deal of data, you may want to have multiple management sites to balance the load or improve performance.
- Smaller-scale management sites may offer you increased efficiency and performance, but multiple sites will also increase the complexity of administrating the system.

In general, if your organization has a large, widely distributed network, you may want to have different management sites focused on particular groups. In a smaller organization, you may prefer to centralize management in one site.

Determining where to locate the management server

In deciding where to locate the management server, you should consider several factors, such as:

- Who is responsible for the servers you are going to be managing in a given site? How many groups need to manage the servers and what level of authority or autonomy is needed for each group?
- What is the geographical distribution of the servers and management groups?
- What is the network bandwidth, latency, and normal load between the computer you want to use as the management server and the servers to be managed?
- What is the network bandwidth, latency, and normal load between the computer you want to use as the management server and the associated repository server?

Although managed clients need to communicate with the management server to send events and data, the most concentrated network traffic that the management server generates is between the management server and the repository database.

In sites supporting 200 or fewer servers, NetIQ recommends locating the repository and management server on the same computer for maximum efficiency and easier maintenance. For sites monitoring more than 200 servers, NetIQ recommends installing the repository and management server on separate computers if the computers are on a **100 Mbps or faster backbone on the same IP subnet**. In most cases, even organizations with 200 to 250 servers can use a single computer for both the repository and

management server (the preferred configuration), unless the computer is overloaded or experiencing performance problems.

If you monitor more than 250 servers, you may first want to consider creating multiple management sites with the repository and server installed on the same computer in each site. This model suits many organizations because it mirrors departmental or geographical divisions already in place and allows for distributed management by groups that may already be functionally separate.

In other cases, however, it may make sense to install the repository and management server on separate computers even if you are monitoring fewer servers. For example, you may want to install the repository on a computer that is managed through your database administration team to take advantage of the database expertise while installing the management server on a computer that is maintained by IT personnel.

In planning where to install components it is just as important to consider your organization's structure and policies as any physical system attributes or network requirements.

Checking system requirements

Each AppManager component and managed object has specific system requirements, such as RAM, disk space, or supported software versions.

The *NetIQ AppManager Installation Guide* describes the basic minimum requirements for AppManager components and how to adjust the requirements if you are installing multiple components on the same computer (for example, for disk space you need the sum of the disk space required for all components you are installing, but you only need the RAM required for the component requiring the most RAM).

In addition, the AppManager Setup program includes a pre-installation check to verify whether the computer on which you are about to install AppManager components passes or fails the

system requirements. For example, the AppManager repository requires that your SQL Server must be configured for at least 100 user connections. The pre-installation check program will tell you if your SQL Server is configured for enough connections.

The pre-installation check also verifies many server-specific requirements. For example, the pre-installation check verifies the version of Microsoft Exchange Server installed for the Exchange managed object or that the SNMP Service is available if you are installing Compaq Insight Manager managed object.

For a more complete review of AppManager requirements, see the *NetIQ AppManager Installation Guide*. Detailed requirements are listed in an appendix in the *NetIQ AppManager Installation Guide*.

Note In addition, to the requirements discussed in the *NetIQ AppManager Installation Guide*, you may want to review the *Tuning Tips for the AppManager Operator Console* Work Smarter Guide. *Tuning Tips for the AppManager Operator Console* provides many recommendations for optimizing a system for peak AppManager performance.

Planning system resource usage for AppManager

In your planning process, you also need to consider the system resources you can expect each AppManager components to use on your workstations and servers. The next sections estimate CPU, memory, and disk used by AppManager components in a few sample configurations.

Note You should also consult the *Tuning Tips for the AppManager Operator Console* Work Smarter Guide for recommendations based on the number of servers you are monitoring.

Resources for a managed client

Each managed client runs two services, `NetIQmc` and `NetIQccm`. Because the `NetIQmc` service receives job requests and runs jobs, the resources it requires will depend on several factors:

- Number of monitoring jobs running on the computer.
- Number of server applications you are monitoring.
- Interval at which the jobs run.
- Types of jobs you run.

Because all these factors can impact resource requirements and repository sizing, it's important to consider this kind of detail about the jobs you want to run as early in the planning process as possible.

As an example, consider a NT_LogicalDiskSpace job that requests two values for each logical disk and evaluates each value to determine whether to send an event. This job puts little demand on system resources. Now consider a job like NT_TopCpuProcesses that requests the CPU utilization value and the values for all processes with their share of the CPU utilization. It must then sort the processes by amount of CPU used, and select the top *n* number of processes, a more complex task that requires more system resources.

Resources for a management server

The management server runs one service, `NetIQms`. The `NetIQms` service serves as the communication broker between the AppManager repository and the managed clients. The resources used by the `NetIQms` service will depend on several factors:

- Number of computers you are monitoring.
- Number and frequency of events that occur in your environment.
- Number and frequency of jobs that are collecting data.
- Network bandwidth and latency.

The `NetIQms` service checks connectivity to managed clients every five to eight minutes. This mechanism is similar to an RPC `ping` command that sends a small data burst with an insignificant impact on network traffic.

The rate at which event or data point information is received will influence the system resources used.

Resources for an Operator Console

The resources used by the Operator Console will depend on several factors, including:

- The preferences and options you have set (for example, the Views, panes, and tabs you decide to display).
- Number of computers (and details) displayed in the TreeView.
- Number of events that occur in your environment and which you elect to display in the TreeView.
- Number of data streams being collected and displayed.
- Number of active, real-time graphs being updated.

Sizing the AppManager repository

One of the most important tasks in the planning process is determining how to size and tune your AppManager repository. There are many factors that influence the configuration of the database, but two of the most important factors are the number of events you expect and the number of data points you intend to collect. Therefore, before you can properly size the repository, you need to draft your management strategy and review the scope of your deployment.

To accomplish this, your project plan should include details about:

- Number of computers you plan to monitor
- The Knowledge Scripts you intend to run
- How frequently you will run them
- Whether you are triggering events, collecting data, or both
- The amount of history you need to keep for reporting

Once you have this information, the easiest way to see the impact of events and data on planning the repository size is to walk through an example.

Estimating with a management scenario

The fictitious company, BrainCandy wants to collect CPU, memory, and disk transfer information. The company identifies these Knowledge Scripts and properties in the project plan:

Knowledge Script	Interval	Collect data	Trigger events
CPULoaded	15 minutes	yes	yes
MemUtil	15 minutes	yes	yes
LogicalDiskIO	15 minutes	yes	yes
LogicalDiskSpace	12 hours	yes	yes

With the 15 minute interval, BrainCandy will sample the following statistics on one managed client 96 times per day:

- CPU processor utilization
- Physical memory utilization
- Virtual memory utilization
- Paging file utilization
- Disk I/O for each disk

Assuming a 3-month period, this would result in collecting roughly 8,640 rows of data (3 months * 30 days per month * 96 intervals per day) for each of these statistics.

With the 12 hour interval, BrainCandy estimates 180 intervals and 3 disks per system, or 540 intervals for one managed client.

Calculating space for the DATA table

With just these Knowledge Scripts and intervals, one managed client generates 61,060 rows of data (DATA table) over 3 months. BrainCandy manages 50 computers, and so estimates 3,053,000 rows in the DATA table.

On average, 14 rows in the DATA table use 1 KB, so it's roughly 14,150 rows per megabyte. BrainCandy then calculates the approximate size required for a three-month "active" data window as follows:

3,053,000 rows * 14,150 rows/MB = 215 MB

Indices for the DATA table use slightly less than half of the data space, so in this case the DATA table requires approximately 100 MB of index space.

Calculating space for the EVENT table

BrainCandy also uses the Knowledge Scripts to generate events (EVENT table). Events are typically deleted and purged from the database more frequently than data points, but as a guideline assume 1000 rows in the EVENT table uses 1 MB.

If all 50 managed clients each trigger 50 events a day for 3 months it would equal 225 MB of event information. Again the indices would be roughly half, or 112 MB.

Calculating the total for the repository

With these estimates, BrainCandy is almost ready to calculate the size for the AppManager repository. The other AppManager tables and indexes use relatively minimal data and index space compared to the DATA and EVENT tables. so for these other objects, assume a total of 35 MB. For log space, we recommend roughly 20% of the size of the data device space, so 90 MB should be sufficient.

BrainCandy can now estimate the disk space for the repository as follows:

315 MB	for DATA data space and index space
+ 337 MB	for EVENT data space and index space
+ 35 MB	for other AppManager objects
+ 90 MB	for log space
777 MB	Total with 687 MB of data device size and 90 MB log device for the AppManager repository database (for example, QDB)

Estimating without a management scenario

If it is not possible for you to estimate the scope of your deployment during the planning phase, it is still important that you have a starting point for sizing the repository and a strategy for how to handle expanding the database.

NetIQ recommends that you count the number of servers you plan to monitor and multiply that number by 1 MB to account for the events and data each will generate. You then multiply the result by the number of days you intend to keep data in the repository.

As an example, assume you have 180 servers that you intend to monitor in this deployment and that you want to retain data in the repository for 30 days to allow time for generating your monthly reports.

If you multiply the 180 servers by 1 MB, then multiply that 180 MB by 30 days, you can estimate that your database is likely to need 5.4 GB for full deployment and data collection.

Although this estimate serves as a good guideline for database growth, it is unlikely that you will run a full set of Knowledge Script jobs and collect all of the data in which you are interested during your initial deployment. Instead, it's likely you will increase the number of servers and the number of jobs you run over time. Therefore, the recommended practice is to set the initial database size during installation to be 2 GB, the initial log device to 600 MB, and to keep the data and the log on separate devices.

Note If you are using SQL Server 6.x, you should also plan to create some additional data devices for expanding the repository database when the database begins to approach this 2 GB initial setting. If you are using SQL Server 7.0, this isn't necessary because SQL Server 7.0 dynamically increases the size of databases, as needed.

Adjusting the size of other databases

As you increase the data you store, you may also need to increase the size of the `temp` database to handle queries that require

temporary space. By default, this database is usually 10 MB. Given the example scenario, a `temp` database of 50 MB would most likely be sufficient. The more data you store and access, the more space you should set aside for the `temp` database.

Note If you plan to make extensive use of AppManager's reporting capabilities, you may want to increase the size of `tempdb` to improve performance. In larger environments, it can be set to 15% - 33% half of the data space. If sufficient resources exist, you can have `tempdb` loaded into RAM. Check with your implementation service representative or NetIQ technical support if you are considering this option, however.

The size of the other databases, for example, the `master` database, can remain the same.

For more information about configuring and maintaining the AppManager repository, see Chapter 8, "Maintaining the repository."

Planning security

In AppManager, security is managed through two layers: the SQL Server security layer and the AppManager user layer.

Because AppManager users are tied to SQL Server login accounts, in planning security for your site you first need to consider the security for SQL Server and for the AppManager repository database (for example, the QDB database).

In planning, you need to decide on the security mode for the repository and how you want to set up SQL Server users, groups, and login accounts for the users who will access AppManager. Once you make these decisions, you use the SQL Enterprise Manager (in 7.0) or SQL Server Security Manager (in 6.x) to set the security mode, create SQL users and SQL login accounts, and grant permission to access the AppManager repository database.

After you have determined the SQL logins, you use the AppManager Security Manager to define which SQL logins are authorized AppManager users and the specific AppManager-related rights those users should have.

A simple method of security management might involve:

- Setting SQL Server to **Integrated** (SQL Server 6.x) or **Windows NT Authentication Mode** (SQL Server 7.0) to use NT user account information as the basis of security management.
- Creating a a SQL login account for each Windows NT user account authorized to access AppManager.
- Using the AppManager Security Manager to set up roles and define the AppManager-related rights for each role.
- Using the AppManager Security Manager to identify the SQL logins authorized to use AppManager as AppManager users and specify the role or specific rights for each of those AppManager users.

Note In addition to this basic relationship between SQL Server logins and AppManager users, there are other facets of security to consider after installation. For example, some Knowledge Scripts require special secure information to run. For a more complete discussion of security and the Security Manager, see Chapter 7, "Managing security."

Planning your management strategy

Planning your deployment has one additional key element: deciding on your core management tasks. Although it's likely you will refine your management policies throughout the deployment cycle, it's a good practice to define a base set of Knowledge Scripts and outline your event handling and data gathering strategy early on.

Identifying management goals

Before you can develop a meaningful management strategy, it is important to consider why you want to monitor the servers in your network and what you want to accomplish. Different organizations have different goals and those goals influence which Knowledge Scripts they run, their event notification policies, and their data collection requirements.

As a starting point, you should identify your primary and secondary objectives. For example, are you most concerned with:

- Immediate event notification when something goes wrong
- Proactive management that allows you to see and respond to potential problems before something goes wrong
- Automating event notification, acknowledgment, and response
- Ongoing analysis and capacity planning
- Service level reporting

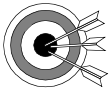
As you consider what you want to achieve through monitoring your environment, you should also prioritize your goals in realistic ways given the size and experience of your staff, the physical location of your servers and administrators, and the business needs of your organization. Having this high-level view of what you want to accomplish will help you decide what you need to implement when.

Selecting Knowledge Scripts and setting thresholds

One good way to begin developing your management strategy is to create a spreadsheet that describes the Knowledge Scripts, intervals, and thresholds you intend to start with. A sample spreadsheet with suggestions for a core set of commonly used Knowledge Scripts is provided in Chapter 9, "Samples and worksheets."

The Knowledge Scripts, intervals, and thresholds in the sample are intended as a baseline to get you started and illustrate a simple management set for basic Windows NT, Exchange, and SQL Server monitoring. Your team will need to review and revise the set of Knowledge Scripts to reflect your organization. For help in selecting the core Knowledge Scripts to run, see "Running the recommended core Knowledge Scripts" on page 52 and Appendix A, "Choosing Knowledge Scripts."

In addition to selecting some initial Knowledge Scripts to run, you may want to set the scripts to collect data for several periods during the planning and testing phases to identify "normal" operating values for your environment and any expected periodic spikes in usage. You can then use this information to set reasonable thresholds that reflect realistic usage patterns and don't create alerts for trivial spikes in performance.



Best Practices Although planning which Knowledge Scripts to run is important, you should also consider your selections to be "work-in-progress." Most likely you will find that you want to run additional Knowledge Scripts and expand your core set beyond your initial expectations. You should plan a review cycle 4 to 6 weeks after deployment to be sure you are covering all of the necessary areas.

Planning event handling and response

In addition to selecting Knowledge Scripts, you need to plan how you will manage any events generated. In most organizations, the event handling strategy evolves over time and is closely tied to which Knowledge Scripts you run. However, there are a few issues that you may want to at least consider during the planning phase.

The project team should articulate the role and expectations for the organization. For example, is the group geared primarily to receiving and responding to events? Is the manpower available to respond to a large number of events or should administrators only respond to events of a certain severity? What constitutes a severe

event and how visible should events of different severity levels be? How should console programs be distributed to let the right people see events that have been raised? AppManager gives you flexibility in controlling the severity levels and the visibility of events. You also have the capability to suppress events to reduce the number reported and filter out duplicate or insignificant alarms.

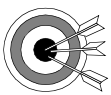
In addition to planning how you will configure events and severity level visibility, you should consider how your organization will respond to events. For example, you should consider whether you will send email, use a paging system, devise automated actions, or send events to a master event console, such as Tivoli Enterprise Console or Cabletron SPECTRUM.

Planning data handling and reporting

Because both event detection and data collection impact the sizing of the repository, it's important that you consider your goals for these operations from the beginning. This is especially true for data collection in that:

- Both real-time and historical graphs and reports rely on setting Knowledge Scripts to collect data and presenting data currently stored in the AppManager repository.
- You need to plan how much historical data you need access to and how and when to archive or delete data points. (Typically, events are closed and deleted frequently, but data is collected on an ongoing basis and is often most useful for trend analysis, requiring you to save the information for a longer period.)

In the planning phase, you should identify your reporting requirements and decide on the set of Knowledge Scripts you want to have collecting data for the reports you require. A sample of popular reports is provided in Chapter 9, "Samples and worksheets."



Best Practices In general, you should start with reports that have an immediate return, such as the Event Summary and System UpTime reports, and limit the number of performance-related

reports until you are sure you are collecting and querying the data you really want. This helps to prevent you from collecting data you don't need or at intervals that are too short. In planning the reports to run, be sure to contact managers who are going to receive the reports to determine the data needed. You may also want to discuss your reporting goals with a NetIQ representative and share ideas or ask for suggestions from other AppManager users on the AppManager mailing list. (To find out more about the mailing list, visit the NetIQ web site at *http://www.netiq.com/support/online/lists.asp*.)

You should consider the frequency and type of data collected to be "work-in-progress." Most likely producing a few successful reports initially will generate requests for additional reports. You should also plan a review cycle 4 to 6 weeks after deployment to evaluate your reporting needs.

Planning for customization

Although not required, many organizations find it useful to identify any custom Knowledge Script requirements during the planning phase. If you need to develop custom Knowledge Scripts, you may need to extend the testing phase to accommodate development.

In addition, you should consider whether you need to plan for customized reports that address your organization's specific needs.

Both Knowledge Script and report customization should be considered "work-in-progress." As new applications are installed and new situations arise, you may find the need to add customized scripts. New applications and requests from managers, applications groups, or end users may require you to develop customized reports. You should periodic reviews to evaluate your customization requirements.

Documenting your decisions and policies

An important part of a highly successful deployment is complete and accurate documentation. Throughout the deployment cycle it is critical that you document your plans, testing, results, and policy decisions. Ideally, each stage of the project should include a document that records the tasks completed and the strategies implemented. For example, you should articulate specific policies for who should respond to which events, how individuals should receive notification, who is responsible for closing or deleting events, and how or when events should be closed or deleted.

Documenting your decisions and communicating policies is an essential part of the implementation—whether it is a matter of recording the standard threshold values you've settled on or identifying who should be paged when problems of a certain severity are found.

Staging the deployment

Although it is useful to make thorough plans before you begin an implementation, the plan is really only an outline to what you need to do. Once you being installing components and using the product you will be constantly refining and clarifying your plans and expanding your management strategy. This is the natural evolution of an AppManager deployment.

Because of this natural progression, we recommend you approach the implementation in a series of stages. This chapter provides an overview of the typical stages and describes the recommended deployment process.

Note For step-by-step installation instructions, see the *NetIQ AppManager Installation Guide*.

Installing in a lab environment

Although it is not always necessary or practical to take the time to establish a test deployment, in many cases having the project team install AppManager in a lab environment can be instructive.

Note In many organizations, this stage is actually part of the AppManager evaluation prior to purchase.

Installing in a lab before any actual deployment can point out particularities of the organization that the project team may need to address to ensure a successful pilot. In addition, the test installation gives the project team experience in installing AppManager components and learning what to expect in the installation process.

Therefore, NetIQ recommends that you install in a lab initially if it is at all possible to do so, especially if the project team has no previous experience installing AppManager components.

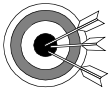
When you install AppManager in a test environment, you should focus on the following key goals:

- Uncovering potential conflicts between AppManager and other applications. (If you do uncover any problems, consult the Knowledge Base on the NetIQ web site for information about resolving the problem or hotfixes that correct it, or contact Technical Support at *support@netiq.com* for help.)
- Quantifying the resource usage requirements of AppManager components before distribution. (This allows you to test your assumptions in a safe but meaningful way, and verify the computers where you intend to install components during the actual deployment meet your requirements.)
- Documenting network utilization between components. (Although you are deploying in a test environment, if you set up a realistic sample of scripts and distribution of components, you'll be able to assess your bandwidth and latency assumptions.)
- Testing the distribution of AppManager agents to ensure you have reliable account information and permissions (for example, usable passwords and domain account names).

Preparing to install the pilot group

Depending on your organization's size, the criticality of your monitoring needs, the expertise of your deployment team, and the resources available to you, the pilot deployment may involve a small representative number of computers or all of the servers you intend to monitor. The recommended practice is install on enough computers to get a realistic view of the full scale deployment.

Identify and contact the owners of the computers that are going to be part of the initial deployment. In meeting with key individuals, provide a realistic estimate of the time it takes to install components. Although installation itself takes very little time, your estimate should provide enough buffer time to troubleshoot any failures.



Best Practices The AppManager setup program does not require you to reboot your servers. Therefore, it isn't necessary to schedule installation for non-production hours. You should, however, consider the importance of any applications running on the servers. For servers running business-critical applications, such as a messaging system, it is best to schedule the installation during a time that will cause the least disruption of service. This practice is recommended when installing any new software on a server that runs business-critical applications, not just AppManager.

The most common sources of failure stem from problems with account privileges and permission to access the servers. Even if you have researched and documented this information, allow enough time to resolve these kinds of issues.

- Identify dates and times the computers are available. In many organizations, access may be strictly controlled or need careful orchestration.
- Gather all the necessary information, such as passwords or user accounts required. Keep in mind you may need information for both the operating system and the application to be monitored.
- Run the AppManager Pre-install check.
- Run the AppManager setup program to install components (the AppManager repository, management server, console programs, web management server, and agent) locally on the selected computers. Although not required, it's recommended that you install the AppManager repository, management

server, and one Operator Console first to verify installation is succesful, then proceed to install additional components where required.

- Run the AMAdmin_AgentInstall Knowledge Script to install the AppManager agent and monitoring probes (managed objects) remotely on selected computers.

Running the recommended core Knowledge Scripts

Once you have installed AppManager on a number of computers in your environment, you need to make specific decisions about what to monitor. During the planning stage, you should have determined a list of the key Knowledge Scripts to run. That list may include many Knowledge Scripts and cover a large number of applications. At this stage, NetIQ recommends you start with a **core set** of Knowledge Scripts that monitor basic server health and availability.

This core set of **recommended** General and NT Knowledge Scripts includes:

Knowledge Script	Guidelines for running this Knowledge Script
General_EventLog	**Understanding what you are monitoring:** This script monitors the Event Log for events based on virtually any criteria. Initially, we recommend you monitor all logs for any Error (“Stop”) type of event. You can further filter the log entries to include or exclude specific IDs, descriptions, user names, computer names, or other criteria.
General_MachineDown	**Understanding what you are monitoring:** This script checks communication between the computer you drop the script on and a list of computers you specify. **Special requirements:** This Knowledge Script requires the AppManager agent services, `NetIQmc` and `NetIQccm`, to run as a service account with Admin privileges on the server where the script is running and on the servers the script is testing the connection to. **Where to run this script:** NetIQ recommends you run this Knowledge Script on a computer in the same subnet as the management server. When specifying the properties for that job, for the Machine List, you should specify a limited number of machines that represent different subnets in your network. You can then drop additional MachineDown jobs on each of the computers specified in the first job to monitor the machines in each of their own subnets. This gives you coverage without stressing network bandwidth. It also ensures that, if a router or subnet is down, you only receive one event for the server being monitored from the agent on the management server’s subnet. The other servers in that subnet will not post duplicate “Machine Down” events.

Knowledge Script	Guidelines for running this Knowledge Script
NT_CpuLoaded	**Understanding what you are monitoring:** This script monitors both the percentage of CPU used and processor queue length. By itself, high CPU usage may not indicate a problem. Instead, you need to consider several factors, including: ■ queue length ■ how you are using the computers monitored ■ your overall strategy for the environment For example, assume you have a **transactional** environment and a computer with CPU usage at 90% consistently. There's no room for growth on the computer, but if the queue length remains low and stable (for example, never more than 2 or 3 processes waiting), it may indicate the computer is sized perfectly for maximum efficiency. If the queue length increases and you have processes waiting, it's likely to be a problem you need to address. In a **batch** environment, you may want to set the thresholds differently; for example, during down times when batch jobs aren't running you may want an event if CPU usage is over 50% and any process is waiting (queue length at 0) to ensure the machine has enough CPU headroom for when the batch jobs are running. Other factors to consider are long range plans, such as the number of users you expect to support, for how long, and how much room for growth you need. For example, you may want to set the CPU usage lower to give you an early warning that you need to off-load some processing or order new systems. **Overall or individual CPU load:** We recommend you monitor load for each CPU individually. This provides more specific information about what is really happening on a system. For example, if you monitor overall load and see CPU usage is 100%, it doesn't doesn't tell you as much about the resource usage as seeing that CPU 0 is running at 90% and CPU 1 is running at 10%. **Handling spikes:** Because CPU and queue length are often subject to temporary spikes, we recommend you set a short interval (2 to 5 minutes), but raise an event only after thresholds are exceeded in 3 consecutive periods. **Collecting data:** This Knowledge Script is typically used for triggering events. If you use it to collect data, the information can be useful in identifying usage trends for your servers. For example, if you see the CPU usage pattern growing steadily, it can help you plan for growth. If you are interested in this type of analysis, you may want to run a second job at a less frequent interval.

Knowledge Script	Guidelines for running this Knowledge Script
NT_MemUtil	**Understanding what you are monitoring:** Physical memory is the most critical statistic you should monitor with this Knowledge Script. It is often used in conjunction with other Knowledge Scripts, such as NT_PagingHigh. **Handling spikes:** Because memory usage is often subject to temporary spikes, we recommend you set a short interval (2 to 5 minutes), but raise an event only after thresholds are exceeded in 3 to 5 consecutive periods.
NT_LogicalDiskSpace	**Understanding what you are monitoring:** This script is recommended for monitoring disk space used and free space available for all logical disks. Running this Knowledge Script is strongly recommended, especially on computers with Exchange Server or SQL Server 7.0. On computers with applications like Exchange Server or SQL Server 7.0 that take available disk space as needed, we recommend you set a lower than normal threshold (for example, 80%) to prevent unplanned changes in disk usage from overloading the system. If you see events, add disk or remove unnecessary files.
NT_LogicalDiskBusy or NT_PhysicalDiskBusy	**Understanding what you are monitoring:** These scripts are recommended for monitoring disk operation and queue length. ■ In a RAID array environment, use LogicalDiskBusy. ■ In a non-RAID array environment, use PhysicalDiskBusy. Events may indicate a slow disk controller, a bad physical disk, or an application that needs to be tuned (for example, a poorly tuned database application that is causing excessive swapping). These scripts can also be used in conjunction with NT_PagingHigh to help you determine the root of an event.

Knowledge Script	Guidelines for running this Knowledge Script
NT_ServiceDown	**Understanding what you are monitoring:** This script is recommended for checking that required services are running and automatically restarting down services. Although you can specify services individually, we recommend you enter an asterisk (*) to check all automatically started services unless you are collecting data. If you are collecting data, specify the services you want to collect data on (for example, you may want to exclude services such as Messenger and Spooler). If you are collecting data, you should not specify all (*) services.
NT_TrustRelationship	**Understanding what you are monitoring:** This script is recommended for checking that a computer in a domain (Brazil) trusts a specified domain (Peru). **Special requirements:** This Knowledge Script requires the AppManager agent services, `NetIQmc` and `NetIQccm`, to run as a service account with Domain Admin privileges. You should not run this Knowledge Script on the PDC.

Beyond this basic group of Knowledge Scripts that are universally applicable, each additional server type or application you are monitoring has its own set of recommended Knowledge Scripts. For a more complete summary of application-specific recommendations and suggestions for which Knowledge Scripts to run, see the *NetIQ AppManager Field Notes: Recommended Knowledge Scripts*, available on the World Wide Web. The Field Notes are updated periodically and provide the most comprehenive information about selecting and running Knowledge Scripts.

(For convenience, some of the most common Knowledge Scripts are listed in Appendix A, “Choosing Knowledge Scripts.”)

Setting thresholds for the recommended scripts

There are three approaches to setting Knowledge Script properties during the first stage of deployment:

- Use the default threshold values or your own experience and knowledge of your environment to begin monitoring right away.

- Set all Knowledge Scripts to only collect data (Event parameter set to n) for a week or more to identify normal baseline operating values before setting thresholds for events.
- Set only those Knowledge Scripts that address critical issues in your environment to check for events and set the remaining Knowledge Scripts to collect data. This approach can be employed enterprise-wide or only on the servers that you have identified as needing immediate attention.

If you haven't been doing any type of monitoring and are deploying a large number of Knowledge Scripts (particularly for applications such as Exchange or SQL Server), the second or third approach is recommended. The first approach, however, has the advantage of allowing you to begin actually monitoring the environment right away.

If you decide to start monitoring for events right away, keep in mind that all AppManager Knowledge Scripts provide default thresholds and intervals. These defaults are based on research, white papers, tuning guides, and field expertise. If you haven't already determined thresholds on your own, use the default values at this stage of your deployment.

In addition, at this stage it is a good practice to track the frequency at which you are seeing events if raising events, or the number of data points if you are collecting data.

What you should see

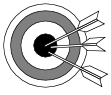
You should remain at this stage in your deployment for two to four weeks. The goal of this stage is to keep things simple, so it is recommended that you don't run additional Knowledge Scripts or strictly limit any additional scripts beyond the recommended core set. You should also strictly limit access to the Operator Console and restrict the number of users allowed to perform various activities, such as acknowledging and closing events or starting and stopping jobs.

The purpose of this initial stage is to have a "shake-out" period that reveals:

- Serious problems that need immediate attention (for example, computers that are dangerously low on disk space or that have pegged CPU).
- Any environmental issues you need to address (for example, problems with insufficient privileges, instability, SNMP or other services that need to be installed).

Keeping to a core set of Knowledge Scripts also helps you prevent your staff from being overwhelmed by a sudden barrage of events or resource problems. By focusing on a limited number of key Knowledge Scripts early in the deployment, you can develop an understanding of the events generated, begin to develop your methodology for responding to them, and effectively troubleshoot any issues that arise.

During this period you also need to begin evaluating your threshold settings and intervals. If you are seeing too many events, it may suggest that the thresholds are set too low for your environment or the interval is too short.



Best Practices Controlling the number of events in the early stages of deployment is an important ingredient in a successful implementation. If the operations or administrative staff becomes overwhelmed with events they aren't prepared to handle, they may become frustrated or, even worse, choose to ignore events as redundant or meaningless. For any monitoring system, the key to success is how the people who are supposed to respond react when an alarm is raised.

Using the data collected

NetIQ recommends that you set your Knowledge Scripts to collect data during the initial stage of your deployment, then run reports associated with the core Knowledge Scripts. From the reports, you can review the high, low, and average values for

core statistics in hourly, daily, weekly, monthly, and yearly format. Reports that provide this information include:

- NT CPU Usage of Processes Summary
- NT CPU Resource Summary
- NT Logical Disk Usage Summary
- NT Memory Utilization Summary
- Event Summary
- Exchange IS Private Resource Summary
- Exchange IS Public Resource Summary
- NT CPU Load Summary
- SQL Data Space Availability and Utilization Summary
- SQL DB Space Availability and Utilization Summary
- SQL Error Log Summary
- SQL User Connections Summary

Based on the information you find in these reports, you can begin to adjust threshold settings to more accurately reflect the specific characteristics of your environment. If the maximum value is still within an acceptable range, reset your threshold to that value or between the average and maximum values.

Adjusting thresholds and intervals

Throughout the first phase of your deployment you should review the data collected by the core set of Knowledge Scripts. Initially, you may see quite a few events as you stabilize and normalize your environment, but as you solve the serious problems that the core set of Knowledge Scripts are intended to point out, you should begin to develop a sense of your normal operating environment and begin adjusting thresholds and intervals to reflect that environment and the event load you can reasonably handle.

■ ■ ■ ■ ■ ■ ■ ■

Use the information in the reports and your own experience to modify the thresholds for the core scripts. Keep in mind that this is likely to be an ongoing process: as you gain experience or as the organization changes, you need to periodically review your settings.

Moving to the next stage of deployment

After running the core set of recommended Knowledge Scripts for two to four weeks, you should have a stable environment that is not generating a large number of events. At the end of this stage:

- You should have been able to identify and resolve any AppManager installation issues, such as problems with domains or account permissions.
- You should have been able to identify and resolve basic problems such as low disk space or available memory.
- You should have a clearer picture of your environment, in terms of the health and operation of the computers you are monitoring, the number of events to expect, and appropriateness of the default thresholds and intervals for your organization.

The time it will take to achieve this can be impacted by several factors, including the distribution of servers and staff, the condition of your environment before deploying AppManager, and the resources available to you. When you feel comfortable with the core set of Knowledge Scripts you are running and the stability of the operating environment, you are ready to move to the next stage of deployment.

Expanding the scope of your deployment

Depending on the scope of your initial implementation and the overall size and distribution of your environment, in the second

stage you may want to deploy AppManager on additional servers.

Typically, very large or widely distributed organizations phase in the deployment of AppManager over a period of time. For example, if your organization is going to monitor a group of servers in the United States, Germany, and Spain, you may decide to deploy first in Germany, stabilize the environment, then expand the deployment to include the servers in Spain and the United States. Alternately, you might decide to expand the deployment to include the servers in Spain, allow time to uncover problems and stabilize that environment, and deploy to the servers in the United States at a later time.

Deploying additional Knowledge Scripts

Regardless of the timing you use to widen the deployment to additional servers, in the second stage of deployment, you should also expand the scope of your management policy to include some or all of NetIQ's second tier of **suggested** Knowledge Scripts.

The set of suggested Knowledge Scripts varies according to the applications you are monitoring and your organization's goals, but as a baseline consider running all or some of the following:

Suggested Knowledge Script	Reason for running this Knowledge Script
General_AsciiLog	You can monitor any text file for a search string, for example, to find error or failure messages. This script is suggested because of the flexibility it provides and because in most environments there are log or text files that should be monitored.
General_Counter	You can monitor any Performance Monitor counter. This script is suggested because of the flexibility it provides and because most organizations have specific counters they are interested in that may not already be monitored in other AppManager scripts.
NT_NetworkBusy	You can monitor the percentage of bandwidth being used.

Suggested Knowledge Script	Reason for running this Knowledge Script
NT_PagingHigh	You can monitor paging activity per second, which is a good indicator of actual system performance.
NT_PrinterHealth	You can monitor the status of printers and print jobs. Recommended for file and print servers and may be useful for some application servers.
NT_PrinterQueue	You can monitor the printer queue length. Recommended for file and print servers and may be useful for some application servers.
NT_RunAwayProcesses	You can find and kill runaway processes based on sustained high CPU usage.
NT_SystemUpTime	You can monitor the number of hours that a computer has been operational since last rebooted.

See Appendix A, "Choosing Knowledge Scripts," for a summary of application-specific suggestions for additional Knowledge Scripts to run.

Identifying your reporting requirements

During this stage of the deployment, you should also focus on your reporting needs. You need to identify:

- The standard AppManager reports to generate and the Knowledge Scripts required to generate those reports
- Who should receive the reports and how frequently
- How reports will be generated (for example, automatically through a scheduled command line process or manually on demand)
- Who should be responsible for generating the reports (for example, you may want to restrict who can use the Report Manager or may want to assign Exchange reports to an Exchange administrator and SQL Server reports to your DBA group)
- The format you want for reports (for example, whether you want them printed, saved in HTML, as CSV, or as a text file)

- How reports should be delivered (for example, through email, posted on a web site, visible in the Web Operator Console, or printed).

There's no standard or recommended set of reports you should run. Some of the common reports to start with are the Discovery, Event Summary, System Up Time, CPU Load Summary, and Logical Disk Space reports.

Expanding the use of actions and notification policies

During the initial deployment, most organizations do not apply many actions or automated responses. As you expand the deployment and gain experience in using AppManager, you should begin to look for ways to expand the use of responsive and corrective actions.

At this stage, you are ready to begin automating some corrective actions, automating your event notification process, and looking for ways to automate the acknowledgment and closing of events. You may also be ready to begin scripting your own actions and planning for future development.

Reviewing and refining the deployment

As you deploy AppManager across your organization, it's likely you will continue to refine your management policies and uncover new requirements. Typically, once the basic monitoring program is in place, organizations develop a deeper sense of their environment and can begin to fine-tune thresholds and job intervals, articulate and automate resolution policies, and tailor event handling, notification, data collection, and the user interface to suit their needs.

At this stage, you are ready to look beyond job and Knowledge Script properties to how you can configure AppManager for your organization. The tasks to focus on include:

- Managing events and event notification.

- Handling data collection jobs and the archiving of data.
- Controlling communication between managed clients and the management server.
- Managing security within AppManager.

Moving beyond deployment to extend AppManager

Even after you have deployed AppManager to all or most of the servers in your environment, you will probably continue to improve and streamline your management process. This is an ongoing process that doesn't end when AppManager is fully deployed. Instead, over time, organizations tend to focus more on extending and customizing AppManager to suit specific needs and on developing more sophisticated monitoring and notification strategies.

In addition, it will become increasingly important for you to manage key aspects of the AppManager environment itself to ensure reliability and optimal performance. For example, you need to develop a consistent backup strategy and plan for when and how to perform routine database maintenance. (See Chapter 8, "Maintaining the repository," for information and recommendations concerning database maintenance.)

At this stage and beyond, you will typically concentrate on the following types of tasks:

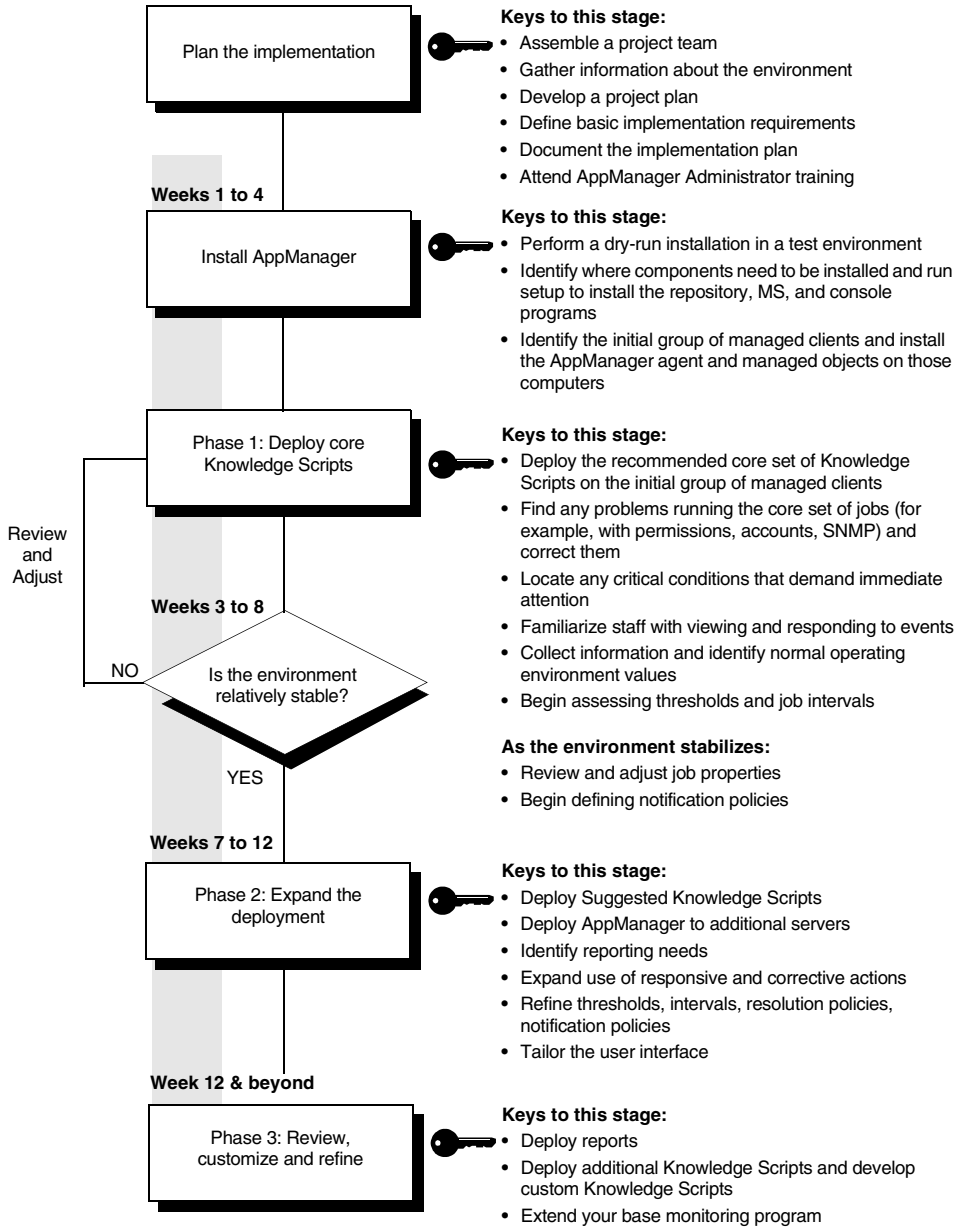
- Deploying reports automatically and designing requirements for any customized reporting (for example, to produce custom Service Level Agreement reports).
- Running additional Knowledge Scripts you have identified as uniquely useful for your environment or for troubleshooting specific problems.
- Identifying any extensions to your base monitoring program (for example, adding new applications or correlating additional monitoring tasks and responsive actions).

- Developing custom Knowledge Scripts.
- Designing more complex notification or resolution rules.
- Integrating AppManager with other products.
- Documenting your extensions to AppManager and your management and resolution policies.
- Maintaining the AppManager repository.
- Periodically reviewing and updating the AppManager environment, if necessary (for example, adding or modifying users and roles, or redistributing components).

Roadmap for a staged deployment

The roadmap on the next page summarizes a typical deployment scenario, including the main focus of each phase of the deployment. It's important to remember that phases tend to overlap and there's a continuous on-going process of refinement. In addition, a key to success for all phases is to document your policies and processes and to communicate those policies and process to others in the organization and make the documentation widely available.

Many of the tasks described generally in this chapter and roadmap are expanded or illustrated through examples in subsequent chapters. For example, to find out more about the kind of automated responses you might implement when expanding the deployment (phase 2), see Chapter 4, "Managing events." That chapter suggests several possibilities and walks through an example in some detail..


Plan the implementation
Keys to this stage:
• Assemble a project team
• Gather information about the environment
• Develop a project plan
• Define basic implementation requirements
• Document the implementation plan
• Attend AppManager Administrator training
Weeks 1 to 4
Install AppManager
Keys to this stage:
• Perform a dry-run installation in a test environment
• Identify where components need to be installed and run setup to install the repository, MS, and console programs
• Identify the initial group of managed clients and install the AppManager agent and managed objects on those computers
Phase 1: Deploy core Knowledge Scripts
Keys to this stage:
• Deploy the recommended core set of Knowledge Scripts on the initial group of managed clients
• Find any problems running the core set of jobs (for example, with permissions, accounts, SNMP) and correct them
• Locate any critical conditions that demand immediate attention
• Familiarize staff with viewing and responding to events
• Collect information and identify normal operating environment values
• Begin assessing thresholds and job intervals
Review and Adjust
Weeks 3 to 8
NO
Is the environment relatively stable?
YES
As the environment stabilizes:
• Review and adjust job properties
• Begin defining notification policies
Weeks 7 to 12
Phase 2: Expand the deployment
Keys to this stage:
• Deploy Suggested Knowledge Scripts
• Deploy AppManager to additional servers
• Identify reporting needs
• Expand use of responsive and corrective actions
• Refine thresholds, intervals, resolution policies, notification policies
• Tailor the user interface
Week 12 & beyond
Phase 3: Review, customize and refine
Keys to this stage:
• Deploy reports
• Deploy additional Knowledge Scripts and develop custom Knowledge Scripts
• Extend your base monitoring program
• Attend Developer training

Managing events

As you expand the deployment, you need to develop and refine your event handling policies, including:

- When events should be generated
- Who should see events
- How individuals should be notified when there's an event
- How events should be acknowledged and closed
- When and how events should trigger corrective actions

Most of these policies must be determined within your organization and involve your group's internal procedures, department structure, and management goals, independent of your AppManager implementation. This chapter addresses event handling and strategies for managing events and event notification from the perspective of how you can use AppManager to achieve your policy goals. It includes examples of some of the ways you can configure event handling within AppManager to suit your needs.

Setting consecutive event intervals

One of the first ways you will begin to control event notification is by setting the number of consecutive intervals a condition should be detected before raising an event. You can set this option as a global (site-level) default and on a job-by-job basis.

NetIQ recommends that you leave the global repository option for the number of consecutive intervals set to one to prevent delays in the reporting of key events and set this option to a higher value for the specific Knowledge Script jobs that:

- Run at frequent intervals because they monitor critical resources, and

- Are likely to experience dramatic spikes.

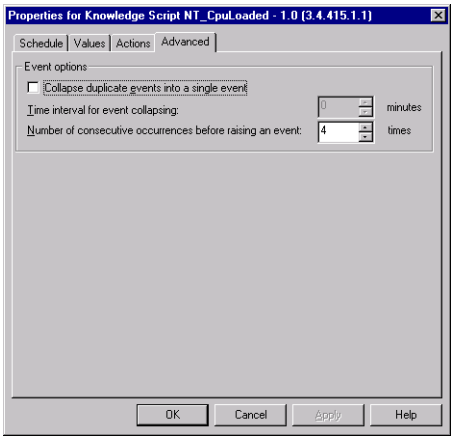
For example, some volatile or highly dynamic system resources such as CPU, memory, and queue length can experience temporary spikes that don't accurately reflect the actual ongoing performance. You can set the number of consecutive intervals option to filter out these temporary spikes for a better indication of a server's state.

To set the number of consecutive occurrences before raising an event on a job-by-job basis:

1. Drag the Knowledge Script to a target in the Operator Console.
2. In the Knowledge Script Properties dialog box, view the **Schedule** tab to see the current interval for the job. Typically, the more frequent the interval for the job, the higher you can set the number of consecutive intervals before raising an event. If you increase the interval between jobs, NetIQ recommends you decrease the number of consecutive occurrences before raising an event.
3. In the Knowledge Script Properties dialog box, click the **Values** tab to set appropriate thresholds.
4. In the Knowledge Script Properties dialog box, click the **Advanced** tab.
5. Uncheck the **Collapse duplicate events into a single event** option.

   Note Event collapsing and consecutive occurrences can work in conjunction with each other or independently of each other. Event collapsing is discussed in "Collapsing duplicate events" on page 71. For now, we are only considering raising an event when an event condition exists over several consecutive intervals.

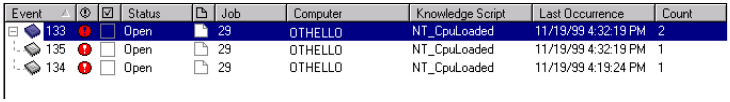
6 Set the **Number of consecutive occurrences before raising an event** to an appropriate value. Typically, this value is set in the range of 3 to 5 occurences.



For example, assume you are running the NT_CpuLoaded Knowledge Script every 3 minutes with the consecutive occurences set to 4. If the percentage of CPU used exceeds the threshold four consecutive times, the fourth occurrence raises an event in the Operator Console and initiates an action, if any action is defined. With this scenario, the first three events occur (the event condition is detected), but are essentially ignored. The interim events are not recorded in the repository, and opening the parent event for the job displays **only** the parent and one child event.

If the problem that caused the event is not resolved during the next three consecutive intervals (that is, the event condition is still present when the job runs for the fifth, sixth, and seventh

time), at the eighth interval a new child event is recorded and another action is initiated.



Event			Status		Job	Computer	Knowledge Script	Last Occurrence	Count
133			Open		29	OTHELLO	NT_CpuLoaded	11/19/99 4:32:19 PM	2
135			Open		29	OTHELLO	NT_CpuLoaded	11/19/99 4:32:19 PM	1
134			Open		29	OTHELLO	NT_CpuLoaded	11/19/99 4:19:24 PM	1

The job was started at 4:10 p.m. and the event condition was found (first interval). The event persists at the second (4:13) and third (4:16) interval. The first parent and child event are recorded at 4:19 (the fourth interval). The condition persists and at 4:32 a second child event is recorded.

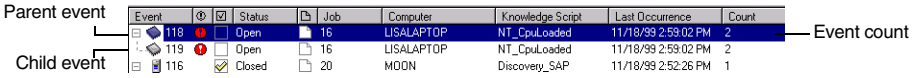
This approach essentially hides the trivial spike events that may occur when a Knowledge Script runs at frequent intervals. Events triggered by temporary spikes are not particularly useful from a practical point of view. They may cause your operations or administrative staff to spend vital time responding to events for conditions that will not cause any disruption to your environment. In addition, these spikes may clutter your database unnecessarily with events you actually do want to ignore.

You should consider this option to help prevent the operations or administrative staff from being diverted from important tasks by responding to non-disruptive conditions. You should also consider this option if you are concerned about accumulating unnecessary clutter in the database.

However, once the job properties are set this way, you do not have visibility of the child events that occur in the intervening intervals. Users unfamiliar with the job properties may assume that the event notification is the first occurrence of a problem, when, in fact, the problem has been detected for several consecutive periods and may need immediate attention. In addition, since you don't see an event until the number of consecutive intervals has passed, you should use caution in applying this option to Knowledge Scripts that run at longer intervals.

Collapsing duplicate events

The first event generated by a particular job creates both a parent and a child event entry in the AppManager repository and Operator Console. Subsequent events from the same job are listed as additional child event entries under the parent event, and an event count keeps track of how many child events are generated.



A Knowledge Script job might detect unique events (for example, events with a different severity or message), or multiple identical events. An event is considered identical, or a **duplicate** event, when a new child event with the same object name, event message, severity, and Job ID as a previous event occurs. Just like unique events, duplicate events are logged in the repository and, if event collapsing is not enabled, are listed as additional child event entries under the parent event. Actions, if specified, are initiated for each duplicate event.

Although duplicate events are typically valid, it isn't useful to receive multiple alerts caused by the same problem or condition. In addition, you probably do not want these identical events to initiate duplicate actions, such as repeated pager notification or messenger dialogs.

Duplicate events can be **collapsed** into a single event. Instead of creating new child event entries, duplicate events are collapsed into the original child event and the child event count increased. Actions associated with collapsed duplicate events, whether the actions are initiated on the managed client or management server computer, are suppressed.

Setting the event collapsing interval

Duplicate events are collapsed within a specified time interval (the default is 20 minutes). The **event collapsing time interval**

begins again 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 five minutes, then the 20 minute interval is restarted every five minutes, which means it never effectively expires – unless you set an option to have events ignored.

Setting consecutive occurences to ignore duplicate events

In addition to the time interval option, you can also set how many consecutive duplicate event occurrences must take place before an actual event is raised, allowing AppManager to ignore duplicate events (not just collapse them). This option is useful in reducing the number of trivial events reported to the management server. Duplicate occurrences that do not generate an event are not logged in the repository and do not increment the event count of the original event. Actions associated with ignored event occurrences are also suppressed.

If the original child event is closed within the event collapsing time interval and a new child event is created, the new child event will initiate a new management server action, if one is specified. The change in the original child event's status does not affect actions that take place on the managed client. After the event collapsing time interval expires, if a new child event is created, a new action is initiated regardless of whether the action is a management server action or a managed client action.

All event collapsing options can be configured globally, as site-level defaults for all Knowledge script jobs, or for individual jobs.

Event collapsing scenarios

As an example to see how event collapsing works, assume you have a job that detects a problem on a computer and continues to detect the same condition every 5 minutes for the next hour, generating a total of 12 identical events.

Without event collapsing enabled

If you uncheck the **Collapse duplicate events into a single event** option, in the first interval the event condition is detected, a parent and child event gets added to the Event pane in the Operator Console. Every 5 minutes a new child event is added under the parent entry for the next hour.

At the end of the hour, there are 12 child entries. The count for the parent entry is 12; the count for each child entry is 1. All 12 events are logged in the repository and all 12 instances of the event have initiated an action, if an action has been specified.

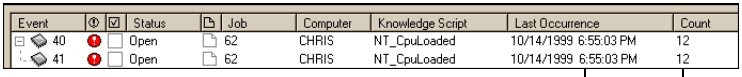
Event	Status	Job	Computer	Knowledge Script	Last Occurrence	Count
65	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:55:03 PM	12
66	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:00:08 PM	1
67	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:05:03 PM	1
68	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:10:03 PM	1
69	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:15:03 PM	1
70	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:20:03 PM	1
71	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:25:03 PM	1
72	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:30:03 PM	1
73	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:35:03 PM	1
74	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:40:03 PM	1
75	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:45:03 PM	1
76	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:50:03 PM	1
77	Open	83	CHRIS	NT_CpuLoaded	10/14/1999 6:55:03 PM	1

With event collapsing enabled

If you check the **Collapse duplicate events into a single event** and use the default values of 20 minutes for **Time interval for event collapsing** and 1 for **Number of consecutive occurrences before raising event**, the first event occurrence creates a parent and a child entry, initiates an action, if one is specified, and begins the 20 minute collapsing interval.

When the Knowledge Script jobs runs five minutes later, it detects the event again and, recognizing it is a duplicate event, collapses the event into the previous child entry. The count is incremented, but a new child is not added. Because this script is running every **five minutes** and the collapsing interval is set to **20** minutes, each new event is identified as a duplicate (only five minutes have passed since the last identical Open event was logged).

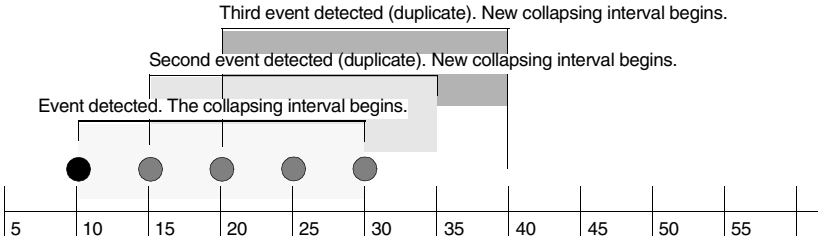
At the end of the hour, the Operator Console displays a parent entry and one child entry; the count for each is 12. All 12 events are logged in the repository but only one instance of the specified action is initiated.



Event			Status		Job	Computer	Knowledge Script	Last Occurrence	Count
40			Open		62	CHRIS	NT_CpuLoaded	10/14/1999 6:55:03 PM	12
41			Open		62	CHRIS	NT_CpuLoaded	10/14/1999 6:55:03 PM	12

Both the parent and single child event display **Last Occurrence** and **Count** associated with the most recent job interval and child event

With the five minute interval, the event collapsing interval becomes a sliding window as long as the event continues to be raised each time the Knowledge Script runs.



Job interval is every five minutes. Event collapsing is set to 20 minutes. The event is detected each time the job runs. A **sliding window** for event for collapsing results. The window will be closed when the event is closed or when no event is detected for 20 minutes.

When no event has occurred for more than the collapsing interval (for example, after 20 minutes have passed), the next event is not suppressed and a new child event is created, a new action is initiated and a new collapsing interval begins.

If the script ran every **forty minutes**, you would see the parent and child from the first time the event is detected and a second child event (and second action) generated when the same event is detected after the collapsing interval expires.

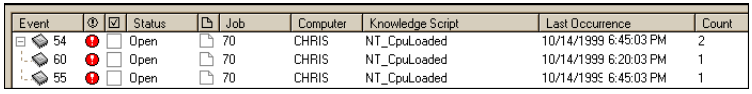
If an event is closed, the next event is not suppressed even if the collapsing interval hasn't passed. A second child event is added to the parent event. If a management server-based action has been specified, it is initiated and a new collapsing interval

begins. (If a managed client action is specified, closing the event does not initiate a new action.)

With event collapsing enabled and some events ignored

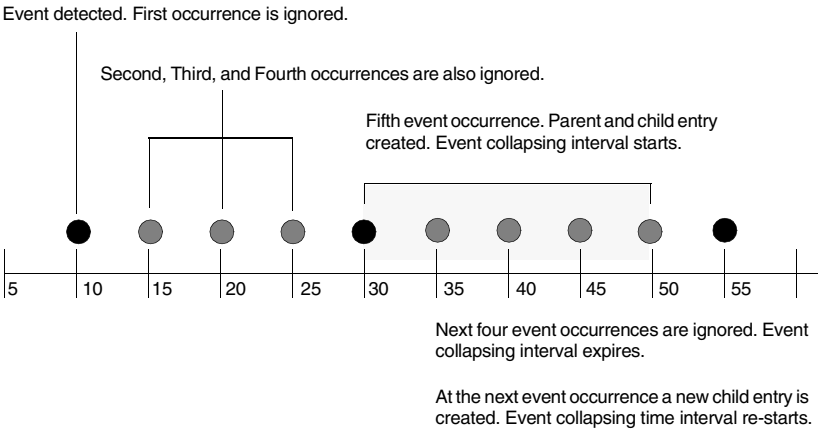
As a final example, assume you have enabled event collapsing, have set the collapsing interval to 20 minutes, and set the **Number of consecutive occurrences before raising event** to 5. With this configuration, the first four event occurrences are ignored. The fifth occurrence generates a parent and a child entry, and begins the 20 minute time interval. The next four occurrences (six through nine) take place during the 20 minute interval and are ignored. The 20 minute interval then expires, which means that the next occurrence (ten) generates another child event and restarts the time interval. The eleventh and twelfth occurrences are ignored, and so on.

A parent entry and two child entries are listed. The count for the parent entry is 2 and the count for each child entry is 1. Only the two listed events are logged in the repository, and their actions initiated. The other events and actions are ignored.



Event	Status	Job	Computer	Knowledge Script	Last Occurrence	Count
54	Open	70	CHRIS	NT_CpuLoaded	10/14/1999 6:45:03 PM	2
60	Open	70	CHRIS	NT_CpuLoaded	10/14/1999 6:20:03 PM	1
55	Open	70	CHRIS	NT_CpuLoaded	10/14/1999 6:45:03 PM	1

To help you visualize this scenario, consider the following illustration:



Determining the event collapsing rules to use

It is worth noting that there is always a one-to-one relationship between child events and management server based actions. Any time a new child event is generated based on the collapsing rules you set, the action you've specified is taken. In addition, as indicated in the examples above, the schedule you've set for the job can impact how event collapsing works.

In general, NetIQ recommends that you enable event collapsing (it is enabled by default) and selectively set the number of consecutive occurrences before raising an event. In setting an event collapsing interval, keep in mind:

- The interval must be set to at least 10 minutes.
- If the job interval is shorter than the collapsing interval and the job detects an event every interval, new identical events will always be collapsed into the same child event (unless the job is interrupted in some way, the collapsing interval is never reached so each new identical event is considered a duplicate event).

- If the job interval is longer than the collapsing interval, you will see multiple child events and initiate multiple actions even though the events are identical.
- If setting the number of consecutive occurrences to be more than one, be sure to consider the job's schedule. For example, if a job runs once every twelve hours and you set the Number of consecutive occurrences before raising an event to 3, you won't be notified of the event until the third occurrence—36 hours later.
- Child events accumulate in the database. You should try to strike a reasonable balance between timely notification and eliminating trivial or redundant events.

Selectively disabling events for a managed client

Occasionally, you may want to selectively disable events. For example, you may want to ignore events of a certain severity temporarily, or you may want to trigger an action without raising an event that needs to be acknowledged and closed. You can do this for any managed client by setting the registry key `HKEY_LOCAL_MACHINE\Software\NetIQ\NetIQmc\Config\NoEventSev`.

By default, the `NoEventSev` registry key is set to null. If you have administrative privileges, you can change the registry setting on any managed client to specify a severity level that won't raise an event. If any Knowledge Script on the managed client detects a condition that normally raises an event, but the event's severity level is equal to or greater than the value you set in the registry, no event is raised in the Operator Console or recorded in the repository. However, any managed client actions you've specified are still carried out.

To see how this works, assume you want to discontinue all informational events on the managed client AJAX. To do this you might set the `HKEY_LOCAL_MACHINE\Software\NetIQ\NetIQmc\Config\NoEventSev`

registry key on AJAX to a value of 25, then restart the `NetIQmc` service on AJAX.

Disabling events raised by specific Knowledge Scripts

You can also use this registry key to control how events raised by specific Knowledge Scripts are handled. For example, assume you want to send a page when the number of sessions connected to the managed client AJAX exceeds 50 without raising an event in the Operator Console. To do this, you might:

1. Set the `NoEventSev` registry key on AJAX to a special value you don't typically use, such as 40.
2. Set the properties on the Values tab for the NT_NetSessions job that runs on AJAX.

For this:	Set to:	For this reason:
Event	y	Because you still want the Knowledge Script job to recognize the condition and trigger an action.
Collect data	y or n	Depending whether you want to collect data.
Connected session threshold	50	Because this is the threshold you want to have trigger the desired action
Severity	40	Because this is the registry key setting that disables the event.

3. Set the action for the NT_NetSessions job that runs on AJAX to Action_Page and set the paging properties. For this particular action, you also need to check the Managed Client Action checkbox (The Action_Page Knowledge Script must run as a managed client action).

Note that any event with a severity level of 40 on AJAX will be ignored. If you have other Knowledge Scripts that do raise diagnostic events you don't want ignored, you can set the severity level in those Scripts to some other value, such as 39.

Using a custom severity value

You can also set the NoEventSev registry key to a custom value to control how events raised by specific Knowledge Scripts are handled. However, to do this you need to either:

- Modify the severity levels allowed in the Knowledge Script.
- Create a custom Knowledge Script that accepts the custom severity level.

For example, assume you want to send a page when the number of sessions connected to the managed client AJAX exceeds 50 without raising an event in the Operator Console, but you want to use a custom severity level to ensure no other Knowledge Script jobs or events are affected.

To do this, you might:

1. Set the NoEventSev registry key on AJAX to a custom value, such as 99.
2. Copy the NT_NetSessions Knowledge Script to create a custom NT_NetSessionNoEvents.
3. Modify the following line in the KPP section to accept a severity level of 99:

   ```
   Const Severity = 8 '[M<Event Severity>M][T<Integer,,,1, 99, SevLevel>T]
   ```

4. Drag the NT_NetSessionsNoEvents Knowledge Script to the managed client AJAX.
5. Set the properties on the Values tab for the NT_NetSessionsNoEvents job.

For this:	Set to:	For this reason:
Event	y	Because you still want the Knowledge Script job to recognize the condition and trigger an action.
Collect data	y or n	Depending whether you want to collect data.

For this:	Set to:	For this reason:
Connected session threshold 50		Because this is the threshold you want to have trigger the desired action
Severity	99	Because this is the registry key setting that disables the event.

6 Set the action for this job to Action_Page, set the paging properties, and check the Managed Client Action checkbox. (The Action_Page Knowledge Script must run as a managed client action.)

How to set the registry key value

If you want to use this feature, you can use `regedit32` or the NTAdmin_RegistrySet Knowledge Script to modify the registry key setting.

If you use the NTAdmin_RegistrySet Knowledge Script, set the following parameters:

Description	How to set
Event upon successful setting	Set to y to trigger events when the registry key is set successfully. The default is n.
Root key	Enter the registry root HKEY_LOCAL_MACHINE (default).
Path name	Enter Software\NetIQ\NetIQmc\Config.
Key name	Enter NoEventSev.
Key type	Enter REG_DWORD (default).
Value of the key	Enter the value you want the registry key set to in decimal format. For example, to set the severity level to 40.
Machine list	Specify the computer(s) for which you want to set this value. If you leave this parameter blank, the registry key located on the local machine is set. To set the value for multiple computers, enter the host name or IP address for each computer in a comma separated list. For example: AJAX,TRISTAN,10.15.221.5.

Description	How to set
File name for machine list	Enter the full path to the file containing a list of the computers for which you want to set this registry key value. The file should contain the host name or IP address for each computer in one or more lines. Each line can have multiple computer names, separated by commas. For example: NYC01,NYC02 SALES01,10.15.221.5,SF01 LABMACH,QATEST
Event severity levels	Set the event notification level. By default, the severity levels are: ■ Set operation successful and the event parameter is set to y: severity 16 (yellow event indicator) ■ Set operation failed: severity 5 (red event indicator)

Automating event notification

Event notification and closure policies vary from one organization to another. In some cases, it is a restricted process, whereas in others it is highly automated. In general, it is a good idea to work closely with your implementation consultant to define a notification and acknowledgment strategy that is tailored to your unique requirements. This section simply gives you examples of some strategies you may want to consider.

Sending an email message

AppManager provides several options for sending an email message in response to an event:

- MAPI mail messages
- SMTP mail messages
- Lotus Notes mail messages

To decide which type of email message to send, you need to consider the email product you use, where and to whom you

are sending the email message, and where the email message originates (from a central location or at each monitored server). For example, to send MAPI mail, you must have Microsoft Exchange or Outlook installed. To send Notes mail, you must have Lotus Notes installed. For SMTP mail, you must have an SMTP mail server.

In addition to the email product you have, consider how you have set up mailing lists, aliases, and network connectivity. For example, if you have defined email aliases for different groups and the addresses are all internal, you may want to use MAPI or Notes mail so that you aren't sending messages through your Internet gateway to deliver internal mail.

On the other hand, you should consider the requirements for each type of mail and how those requirements suit your environment. For example, Notes mail must be sent from a Domino server so the mail must be sent from each server you are monitoring. MAPI mail requires you to install a MAPI client (such as Outlook) on the server that will send mail. In most cases, unless you have already installed the managed object for Exchange, this means to send MAPI mail, you need to:

- Install a MAPI client on either the management server (to send mail from a central location) or each server you are monitoring.
- Create an account for the AppManager agent service, `NetIQmc`, to use on either the management server (to send mail from a central location) or each server you are monitoring. (The service account is required to link to an Exchange profile and mailbox.)
- Set the AppManager agent services to run under the user account linked to the Exchange profile.

In general, SMTP mail provides the best method for email 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 servers you are monitoring. The major drawback to using

SMTP mail is that your SMTP gateway may have security rules that limit the users that 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 have permission to send mail on the servers from which the mail will originate.

Depending on your geographic or departmental make up you may want to use more than one mail method for event notification. As an example, assume you have several Exchange Servers you want to monitor on one continent and your SMTP gateway on another continent. The IT staff that supports the Exchange Servers is located in the same location as the servers, but the main administrative staff is located elsewhere. With this scenario, 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 routed through the SMTP gateway for distribution.

Forwarding an event to a paging system

There are several ways to forward an event to a paging system. You can:

- Use the Action_Page Knowledge Script
- Use SMTP mail messages to send pages
- Use MAPI or Notes mail to send email for paging aliases
- Use an alternative method for sending event messages to pagers

Using Action_Page to send pages

The Action_Page Knowledge Script relies on a file called `NetIQpage.ini` located in your `%WINNT%` directory on each

monitored server. The `NetIQpage.ini` file is pre-configured with much of the information necessary to send paging messages to several paging systems, including Attention!, HipLink, TelAlert, AlarmPoint, and CuraSoft.

To use the Action_Page script, you need to install the client software for the paging application on the server from which you will run the Action_Page. Because of this, the client software is typically installed on the AppManager management server and the paging action is initiated by the management server.

For more information about how to edit the `NetIQpage.ini` file and use this Knowledge Script, see the *NetIQ AppManager Knowledge Script Reference Guide* or select Action_Page in the Operator Console and press F1 for help.

Using SMTP mail messages to send pages

Some paging applications accept SMTP mail messages to send pages. If your paging application accepts SMTP mail, you can configure the action to be initiated on the management server or on the individual servers being monitored. As discussed under "Sending an email message" on page 81, SMTP mail does not require any client software to be installed, but there may be security issues that may prevent you from using SMTP mail messages to send pages to your paging application.

To use SMTP, you need to 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 may 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, you 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 email message to the pager.

Use MAPI or Notes mail to send email to paging aliases

Some paging applications accept MAPI or Notes mail messages to send pages. As discussed under "Sending an email message" on page 81, both Notes and MAPI mail have special requirements for sending mail.

- MAPI mail requires client software to be installed and configured. It is usually best to set up MAPI mail as an action that executes on the management server.
- Notes mail method can only be run from the Domino server(s) you are monitoring.

Using another method of posting event messages to pagers

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

In addition, some unsupported Knowledge Scripts that interface with paging applications without using the `NetIQpage.ini` are available from the KS Depot on the NetIQ web site, *http://www.netiq.com/support/online/ksdepot.asp*.

You can also make your own modifications to scripts like Action_DosCommand or Action_RunSQL to interface with paging applications. If you decide to create your own interface, see to the *NetIQ AppManager Developer's Guide* for information on writing your own scripts.

Forwarding an event to another console

In larger organizations, administrators often implement AppManager as the solution for monitoring Windows NT, Windows 2000, and the applications that run on those platforms and then integrate AppManager events into another console or redirect events to a help desk application.

Using AppManager Connectors

To make the forwarding of events as transparent as possible, AppManager supports a variety of **Connectors** to the most common event consoles and help desk applications. For example, AppManager has Connectors for:

- Tivoli TEC
- CA Unicenter TNG
- Cabletron Spectrum
- HP Network Node Manager
- HP VantagePoint Operations
- Micromuse NetCool
- Remedy Action Request System

Connectors provide a centralized method for interfacing with the applications to which you want to forward events. The Connector is installed on the AppManager management server to forward events as they are sent from the managed client agents. For more information on supported Connectors, visit the NetIQ web site at *http://www.netiq.com/support/online/worksmarter.asp*, or contact your NetIQ technical representative.

Using AppManager Action Knowledge Scripts

If you prefer to interface with an application for which AppManager does not currently have a Connector, then you may want to use actions such as sending an SNMP trap, dumping information to the NT Event Log, or writing information to an ASCII log file to transfer AppManager event information to

another application. You can also create your own custom action script that calls a COM object or command line interface to the application of your choice.

Using SNMP traps Sending an SNMP trap is a relatively simple and cost effective solution for transferring event information. It does not require any additional client software to configure or maintain or any additional client licenses for your event console and/or help desk application. If you use SNMP traps, it is useful to run them as managed client actions because it allows you to configure two network paths for events to travel to separate applications.

Using log file or command line interfaces Actions that write to the Event Log, ASCII log files, or send command line arguments can be configured to run on the managed client or the management server. However, these methods usually require that you install, configure, and maintain a client executable for the product to which you want to connect on each managed client from which you want to be able to forward events. This may also incur additional licensing costs. For these reasons, if you decide to forward events through the Windows NT Event Log, ASCII log files, COM, or a command line interface, you will probably want to configure the event forwarding action to take place at the management server to reduce monetary and maintenance time costs.

Automating corrective action

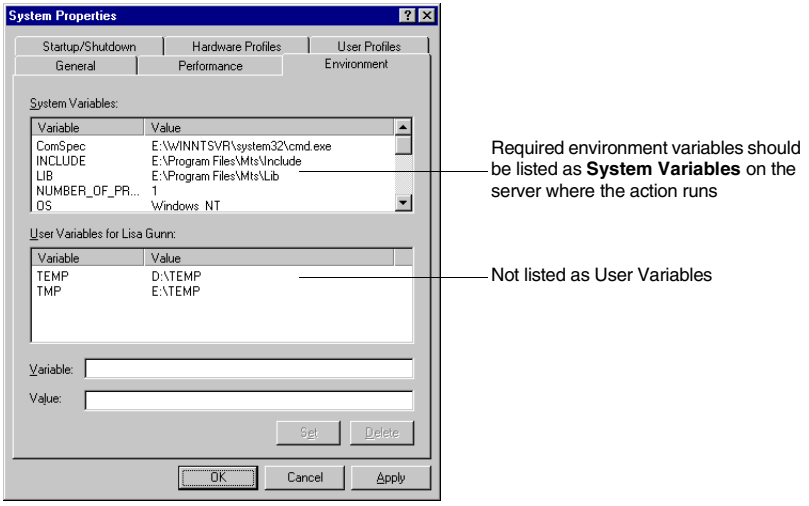
Most actions revolve around event notification and visibility. However there are cases when an action Knowledge Script can actually be used to automatically correct a problem that has been detected. Typically, corrective actions run a command or execute SQL statements on the managed client computer in response to an event. Therefore, the most important thing to remember is that you want any corrective action to take place on the managed client not the management server.

For an Action Knowledge Script you are running as a correction action (for example, Action_DosCommand, Action_DumpTran, or Action_RunSql), check the **Managed Client Action** check box on the Action tab in the Knowledge Script Properties dialog box.

In addition to ensuring that the action runs on the managed client, you should be sure that 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 will run.

Another consideration in automating corrective actions is that some actions require environment variables to be set to run properly. For example, using the Action_DosCommand to call the Attention! executable, `attn.exe`, requires the environment variable `attnsrv=<servername>` to be set.

If your corrective action requires environment variables, be sure the environment variables are either instantiated by the action script or that they are preset as **System Variables** in the System Control Panel for the managed client computer. For example:



Finally, be sure that any command you execute with an action script is a **non-interactive** command. The command or script you execute should not require an input or user response. For example, if you decide to run a command line command to delete files, you should use the `del /Q` option rather than just `del` to ensure that any wildcard deletions take place without prompting for confirmation.

Managing data collection

This chapter discusses the impact of data collection and the options you should consider when planning to set Knowledge Scripts to collect data.

Deciding when to collect data

Typically, there are four reasons to collect data:

- To identify normal operating environment performance values or baseline performance values.
- To periodically check values for troubleshooting (for example, to view the top CPU consumers after uncovering problems with processes failing).
- To create data streams for real time graphs that you view in the Operator Console or the Web Operator Console to help you identify short-term trends or perform ad-hoc comparisons of performance data.
- To store historical information for trend analysis, capacity planning, or management and 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.

As an example, consider a script that checks server connectivity. It may be that you want to run this script every 5 minutes to ensure prompt event notification if the connection goes down. For the purposes of data collection, however, you might want to run the script less frequently or only during business hours because you only need to report connectivity percentages for those hours. By adjusting the schedule for the job that collects

data, you prevent the script from cluttering the database with information you aren't interested in.

In some cases, getting a data point very 5 minutes is useful but there are also cases where it may not really provide you with much more information than collecting the point every hour. For example, CPU and memory usage may change significantly in a 5 minute period, but logical disk space is not likely to change as frequently and may be more effectively monitored at 12 or 24 hour intervals.

The best practice is to have a clearly defined purpose for collecting the data, for example, to produce a weekly report of the top ten email users. In addition, it is 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 will help you make the most intelligent decisions about when and how to collect data.

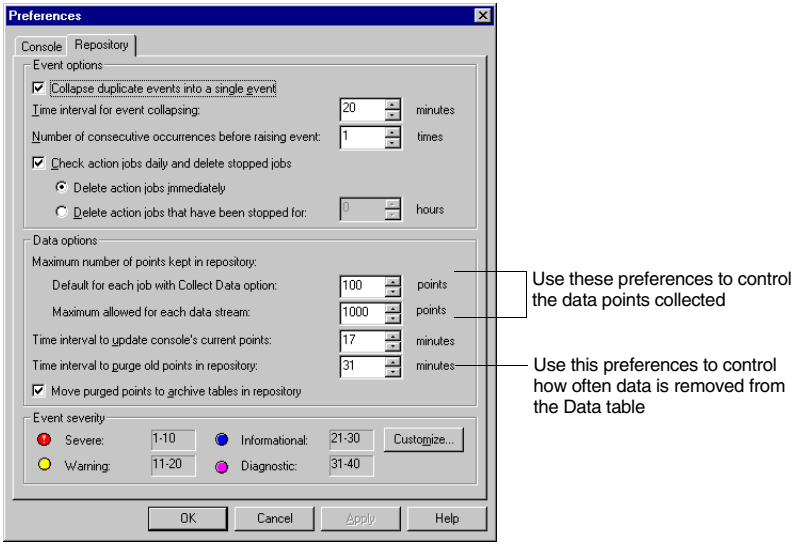
To illustrate the importance of this point, consider a Knowledge Script such as NT_TopCpuProcs. Occasionally, you may want to run this script to report the CPU usage for all processes running on a managed client, for example, because you are investigating a problem on the computer and want a complete profile of the processes running and their CPU usage. It's unlikely (and not recommended) that you would run this job to collect data for all processes on multiple computers on an ongoing basis because of the potential overhead (in database space and performance) involved in returning back that much data in each interval.

Setting the maximum number of data points

In the AppManager repository, current data streams and data points are stored in the database tables **DataHeader** and **Data**. Because data is collected at each interval the size of these tables can quickly grow and affect your database maintenance requirements.

To prevent this, AppManager allows you to define the maximum number of data points stored in the Data table per data stream. You set the repository-wide defaults for a maximum number of data points on the Repository tab in the AppManager Preferences dialog box (for example, in the Operator Console click **File > Preferences**). There are two options:

- **Default for each job with Collect Data option** sets the default number of data points collected for each data stream. You can change this value for individual data streams but the value can't be greater than the global maximum defined in Maximum allowed for each data stream.
- **Maximum allowed for each data stream** sets the maximum number of data points that can be collected for each data stream.



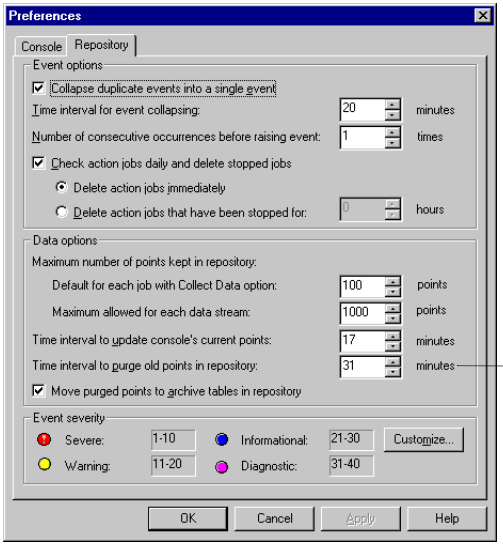
When the number of data points in a data stream exceeds the maximum you set, the oldest data points in the data stream are removed from the Data table. A SQL Server scheduled task runs at regular intervals to remove the old data points. For example, if you set the maximum number of data points to 500, when a

Knowledge Script runs and collects its 501st data point, the newly-collected data point is recorded as the most recent in the stream and the oldest data point in the data stream is removed from the Data table at the next run of the scheduled task (see “Purging older data points” on page 94 for more information about this task).

Setting the maximum number of data points ensures the most recent information is available for graphs and reports while minimizing the impact on database performance and maintenance.

Purging older data points

In addition to setting limits on the **number** of data points in a data stream, AppManager allows you to set an **interval** for removing older data points to keep information you are no longer interested in from impacting your database performance. You set the **Time interval to purge old points in repository** on the Repository tab in the AppManager Preferences dialog box:



Use this preference to control the frequency for purging old data

When a Knowledge Script runs and collects a data point, the data point is recorded in the **Data** database table. To keep this table from growing continuously, a scheduled task runs at the interval you set (by default every 31 minutes) and removes the oldest data from the table.

Note The maximum number of rows the NetIQ PurgeData QDB scheduled task will remove in any single execution is 100,000 rows (100,000 data points). If you are collecting large amounts of data and change the interval for this task, you may need to change the maximum number of rows it will remove. For example, if you are collecting a million rows of data per day and change the task to run every 12 hours but don't change the maximum rows removed, data will continue to accumulate in the Data table to the point where the database becomes practically unusable. Changing the maximum number of data points removed requires you to tune the stored procedure invoked by the NetIQ PurgeData QDB scheduled task.

Editing the stored procedure parameters for PurgeData

The PurgeData task invokes the PurgeAllData stored procedure. This stored procedure purges and moves data in a progressive fashion to minimize the SQL Server load and reduce the possibility of deadlock. It has three parameters:

This parameter:	Does this:
@batchsize	Defines the batch size for any purge for each DataID. A value of 0 means no limitation. The default value is 1000.
@maxdataid	Defines the maximum number of DataID's (Data Headers) processed in each invocation. A value of 0 means no limitation. The default value is 600.
@maxpoint	Defines the maximum number of data points purged per execution. A value of 0 means no limitation. The default value is 100000.

Normally, the NetIQ PurgeData QDB task invokes the PurgeAllData stored procedure without explicitly specifying any

parameter values. Therefore, each parameter takes its default value.

Tuning these parameters and adjusting the schedule for the NetIQ PurgeData QDB task can help you optimize SQL Server performance and reduce the possibility of deadlock.

To edit the parameter values when you are using SQL Server 6.5:

1. Start SQL Enterprise Manager.
2. Select the AppManager repository server.
3. Click **Server > Scheduled Task**.
4. Highlight the **NetIQ PurgeData QDB** task.
5. Right-click and select **Edit** from the right-click menu.
6. Edit the Command text to specify the parameters you want to change. If you are only changing one parameter to use a value different from the default, you only need to add that parameter and value to the Command.

   For example, if you only need to change the maximum number of data points removed to 500,000:

   ```
   EXEC dbo.PurgeAllData @maxpoint=500000
   ```

   Tip The `@maxpoint` parameter is the parameter you are most likely to change. Be sure to set it to a value equal to or greater than the number of data points you expect to collect in the execution interval for the scheduled task.

To edit the parameter values when you are using SQL Server 7.0:

1. Start SQL Enterprise Manager.
2. Select the AppManager repository server and expand to display the scheduled tasks (under Management > SQL Server Agent > Jobs):

3 Highlight NetIQ PurgeData QDB and right-click or double-click to display the Properties dialog box.

4 Click the **Steps** tab.

5 Edit Step 1.

6 In the Edit Job Step dialog box, modify the Command text to specify the parameters you want to change. If you are only changing one parameter to use a value different from the default, you only need to add that parameter and value to the Command.

   For example, if you only need to change the maximum number of data points removed to 500,000:

   ```
   EXEC dbo.PurgeAllData @maxpoint=500000
   ```

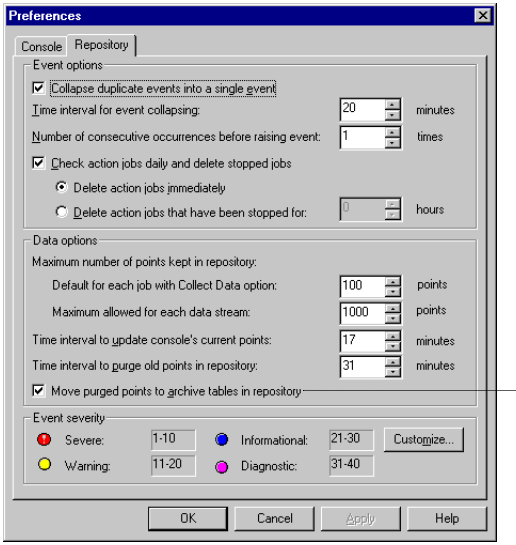
   Tip The `@maxpoint` parameter is the parameter you are most likely to change. Be sure to set it to a value equal to or greater than the number of data points you expect to collect in the execution interval for the SQL Server Agent Job.

Archiving data for historical reporting

In the AppManager database, the most current data point information is stored in the **Data** database table and this table is periodically purged for database efficiency. The purged information may still be included in previously-created graphs that were active at the time the information is purged, but you won't be able to drill down to the data point because the data point has been removed.

To save the purged data points, AppManager allows you to archive the older data by moving the data points to a new table where they are still available for reports. You do this by selecting the **Move purged points to archive tables in repository**

option in the AppManager Preferences dialog box on the Repository tab (by default this option is selected).



Use this preference to save older data points for reports

When you set this option, the data points that are purged from the Data table, for example, every 31 minutes, are moved to the **ArchiveData** table. The ArchiveData table allows you to save the older data points for historical reporting. This option also enables you to save data points if a data stream exceeds the maximum number allowed. For example, if the maximum number of data points in a data stream is 500, when the 501st data point is collected, the oldest data point in the data stream is still removed from the Data table, but it is automatically recorded in the ArhiveData table.

If you deselect the **Move purged points to archive tables in repository** option, when a data stream exceeds the maximum number of data points allowed, the oldest data points get deleted from the database as new data points are collected.

Should you archive data?

AppManager does not require you to save data points in the ArchiveData table. If you don't need access to the historical data at a later time, you may choose not to archive data points at all. Choosing not to archive data reduces the sizing requirements for your database. In addition, archiving data requires periodic maintenance to prevent the archive tables from growing continuously in much the same way as the Data tables would if you set no limits on the data points collected (see "Removing archived data and old events" on page 141).

If your primary interest is in current data (what's happening right now or in a small window 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. However, setting the maximum number of data points too high is likely to drag down database performance.

If you choose to move data points to the archive tables, you will have access to historical information that's been moved out of the current stream. If you save archived data you should be sure you allocate enough space for your AppManager repository. In addition, you should make periodic archive backups that include snapshots of old data so you can purge the data from the AppManager repository. Limiting the data you need to store in AppManager database improves performance and reduces size and memory constraints.

Note Unlike the automated removal of old data handled by the NetIQ PurgeData QDB task, data is **not automatically removed** from the **ArchiveData** table. To remove data from the ArchiveData table, you must manually run the DeleteOldData stored procedure, create a scheduled task to run DeleteOldData, or manually run a "`DELETE From`" SQL command. For more information on database maintenance and configuration, see Chapter 8, "Maintaining the repository."

Configuring site communication

This chapter describes ways you can configure and customize the communication between your managed client computers and the AppManager management server. Configuring site communication is optional, but it allows you to tailor when and how managed clients communicate with the management server to suit your network requirements, bandwidth, latency, and operational policies.

Authorizing a management server to communicate with a managed client

Whenever you install the AppManager agent on a computer, you have the option of specifying the list of management servers that are allowed to communicate with the local AppManager agent (NetIQmc) on that computer.

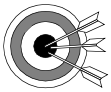
By default, this option, whether you use the setup program or the AgentInstall Knowledge Script, is set to * to authorize all management servers to communicate with the local computer. To restrict which management servers are authorized to communicate with the computer, you enter a comma-separated list of computers names. This information gets stored on the managed client in the registry under:

HKEY_LOCAL_MACHINE\SOFTWARE\NetIQ\NetIQmc\Security\AllowMS

For example, if you have specified that `shasta` and `dynamo` are the only management servers authorized to communicate with the AppManager agent on the local computer, the `HKEY_LOCAL_MACHINE\SOFTWARE\NetIQ\NetIQmc\Security` registry key will contain the following:

```
AllowMS:REG_SZ:shasta,dynamo
```

Note If a managed client can't resolve the name of the management server you specify, the AppManager agent services may fail to start and you won't be able to monitor the computer.



Best Practices Placing restrictions on the servers that can communicate with the AppManager agent on a managed client computer also restricts the use of AppManager diagnostic tools (for example, `NetIQctrl` or AppManager Troubleshooter). If you decide to restrict access, NetIQ recommends that you always include the name of the workstation or server the administrator of your AppManager environment uses in the list.

Changing the authorization list

There are two ways to change the list of managements servers authorized to communicate with a managed client:

- By manually editing the AllowMS registry key value using the Registry Editor or the NTAdmin_RegistrySet Knowledge Script.
- By re-installing the AppManager agent and specifying the new list of authorized management servers.

The method you choose may depend on when you decide to make the change and whether you have applied any hotfixes or customization since first installing the AppManager agent on the managed client. In general, modifying the AllowMS registry key value is a safe practice.

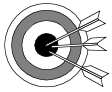
You can modify the AllowMS registry key value if you have administrative privileges on the managed computer where you want to make the change. For more information about AppManager registry keys, see Appendix B, "Registry keys."

Note If you change the list of management servers for a managed client, be sure to stop and restart the AppManager agent services on that computer to make the change effective.

Changing the name of a management server

If you specifically authorize the management servers that are allowed to communicate with a managed client, changing the name of the management server will require you to manually set the AllowMS key value on **each managed client** that references the old management server name.

You should be especially careful about changing the name of a management server if each managed client only has one authorized management server as this would completely sever communication with the managed client and prevent you from seeing any events raised or getting collected data into the repository.



Best Practices If you must change the name of the management server, run the NTAdmin_RegistrySet Knowledge Script on all of your managed client computers to change the value of the `NetIQms` registry key to the new name before changing the management server name. After you change the registry key value, restart the `NetIQmc` service on each managed client.

Configuring network communication

There are several AppManager Administration (AMAdmin) Knowledge Scripts that are used to configure different aspects of site communication.

Using an IP address or hostname

By default, the `NetIQccm` service uses an IP address to locate and communicate with the management server. However, in some environments this may be problematic. For example, if your management server and managed clients are connected through a remote dial-up and use DHCP, IP addresses may be assigned dynamically and change from one connection time to the next, or your management server may be installed on a cluster

requiring you to use a cluster name rather than a specific IP address.

For each management site (managed client, primary management server, and repository), decide whether you want the NetIQccm service to use an IP address or hostname to locate the management server. Once you decide the best approach, you can use the AMAdmin_ConfigSiteCommType Knowledge Script to set or change the communication type.

Controlling the flow of network traffic

The ConfigSiteNetFlowCtrl Knowledge Script helps you manage network bandwidth and control and tune the transfer of data from managed clients to the management server to suit your network capacity. With this Knowledge Script, you can restrict the amount of data the NetIQccm service 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 100K, a low watermark of 2K, and a communication interval of one hour (3600 seconds). With this configuration, NetIQccm sends up to 100K of data per hour to the management server until the data waiting to be transferred falls below 2K. NetIQccm then stores the data in the local repository. At the next interval, if the data to be transferred is greater than 2K, NetIQccm resumes sending the data to the management server. If the data package is still below 2K, NetIQccm continues to store the data in the local repository until the next interval..

The ConfigSiteNetFlowCtrl Knowledge Script also provides dynamic tuning to allow the NetIQccm service to respond to load changes on the management server. With dynamic tuning, each time the NetIQccm service connects to transfer data, it checks the management server's current load. If the management server is

busy and load has increased, NetIQccm reduces the data sent and increases the communication interval.

Each time the NetIQccm service connects to transfer data, it checks the management server's current load. If load has increased, NetIQccm further reduces the data sent. NetIQccm continues to reduce the data sent until the amount of data to be sent falls below the low watermark, or load on the management server decreases, freeing up bandwidth.

Configuring a local repository

You can use the SetLocalRPSize Knowledge Script to control the maximum number of events or data points that can be stored in a managed client's local repository. If the managed client is not able to communicate with the management server for any reason, the local repository for the managed client 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, figure that 1000 rows is roughly equivalent to 1 MB.

For more information, see the *NetIQ AppManager Knowledge Script Reference Guide* or AMAdmin_SetLocalRPSize online help.

Configuring and scheduling data transfers

The SiteSchedUpload Knowledge Script is used to specify a schedule for uploading data and/or events from a managed client's local repository to the current management server.

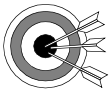
This Knowledge Script allows you to store performance and event data in the local repository until you're ready to upload it to the management server. By giving you the flexibility to transfer events and data during off-peak hours or when network traffic is light, the AppManager management server and 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 managed clients, you may want to store the data locally on those managed clients 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 managed clients to be uploaded at staggered times, further reducing the load on the management server and repository.

You can set up a specific schedules for data, events, or both, as needed. The `NetIQccm` service stores the events or data points in the local repository until the scheduled upload time. At upload time, `NetIQccm` reads the events and/or data points from the local repository and sends them to the management server.

The SiteSchedUpload Knowledge Script is often used in conjunction with two other Knowledge Scripts to provide maximum control over data transfers:

- You can configure the size of message batches delivered in the upload with the ConfigSiteNetFlowCtrl Knowledge Script (see "Controlling the flow of network traffic" on page 104).
- You can configure the maximum number of data points or events to store in the local repository with the SetLocalRPSize Knowledge Script (see "Configuring a local repository" on page 105).



Best Practices Configuring your managed clients to immediately forward events while storing performance data locally provides you with maximum flexibility in determining your transfer strategy without impacting event notification.

Scheduling maintenance periods

In most organizations, application servers are periodically shut down to allow administrators to perform housekeeping tasks. These regularly scheduled maintenance periods, or downtime, can cause your monitoring jobs to issue unnecessary alerts or to fail when they attempt to run.

For example, an organization may shut down half of their Exchange Servers every Friday at 9 p.m. This interruption causes several Connectivity events as test messages fail to reach their destinations. It also causes all of the AppManager Exchange jobs on the down servers that aren't explicitly stopped to error out, forcing the administrator to restart the jobs manually when the servers are brought back online.

It is also possible that AppManager may attempt to automatically restart services that have been intentionally stopped, and thereby interfering with the maintenance planned.

Temporarily blocking jobs

To avoid this problem, you can use the AMAdmin_SchedMaint Knowledge Script to specify the maintenance period for target computers. You have the option to suspend all jobs during the maintenance period or to suspend specific types of jobs (for example, all Exchange jobs on the target computer).

During the maintenance period, the AppManager jobs you specify are prevented from running. Other AppManager jobs, on other computers or in other categories, are not affected. This enables administrators to selectively disable jobs during a planned maintenance period.

It is worth noting that you can only specify one maintenance period per server for a specified group of jobs. For example, if you create a job to suppress Exchange Knowledge Scripts on the server FALCON from Saturday at 10:00 p.m. until Saturday at 11:00 p.m., you can't create a second job on FALCON that suppresses

Exchange Knowledge Scripts on Sunday from 3:00 a.m. to Sunday at 4:00 a.m. You can, however, create one job to suppress Exchange Knowledge Scripts using one schedule, then create a second job that suppresses all Knowledge Scripts for a second maintenance period.

For more information, see the *NetIQ AppManager Knowledge Script Reference Guide* or AMAdmin_SchedMaint online help.

Note The AMAdmin_SchedMaint script disables Knowledge Script jobs that **run on** the target server. If you are running Knowledge Scripts such as PingMachine or MachineDown Knowledge Scripts that attempt to **access** the target server you have scheduled maintenance for, these scripts will still raise an event if the target server cannot be reached.

Setting a resource dependency

You can further control when Knowledge Scripts run by setting a resource dependency. For example, if regularly scheduled maintenance periods aren't reliable or hard to anticipate, you may want specify that jobs only run when required file-system related resources or specific services are available.

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 monitor Exchange Server, you may want to check that the MSExchangeDS, MSExchangeIS, and MSExchangeSA services are running before running Exchange Knowledge Script jobs. Even if you have established a maintenance period and the maintenance period has expired, if these services are not running, the Exchange Knowledge Script jobs are prevented from restarting. For more information, see the *NetIQ AppManager Knowledge Script Reference Guide* or AMAdmin_SchedMaint online help.

Chapter 7

Managing security

This chapter provides an overview of AppManager's security features and describes the relationship between AppManager security and the security and administrative privileges you manage outside of AppManager.

It also includes general information about using the AppManager Security Manager to perform security-related tasks, such as setting up AppManager users, defining roles for AppManager users, and storing sensitive information needed by Knowledge Scripts. For more complete information about using the AppManager Security Manager, see the *NetIQ AppManager User's Guide* or the online help for the AppManager Security Manager.

The following topics are covered:

- Understanding the layers of security
- Understanding user access to AppManager
- Setting up AppManager users
- Understanding AppManager roles
- Special administrative privileges
- Storing security information for Knowledge Scripts

Understanding the layers of security

Before considering the specifics of AppManager security, it's useful to understand some of the larger security issues that you manage outside of AppManager and how AppManager can use the security layers you already have in place.

Because the AppManager repository is a SQL Server database, it's helpful to understand something about SQL Server's security.

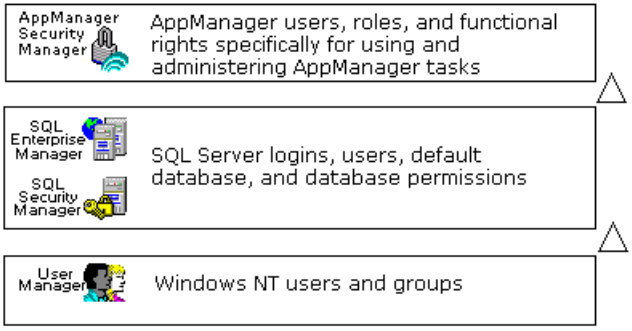
Therefore, as a starting point, you need to be familiar with the security mode the SQL Server you are using for AppManager is configured to use. The security modes in SQL Server are:

- **Integrated** (SQL Server 6.x) or **Windows NT Authentication Mode** (SQL Server 7.0) security uses Windows NT authentication to validate SQL Server logins for all connections.
- **Mixed Mode** security allows SQL Server login requests to be validated using either Windows NT authentication or other authentication methods (such as Kerberos authentication).
- **Standard** security, available in SQL Server 6.x but discontinued in SQL Server 7.0, uses SQL Server's own login validation process for all connections.

Each security mode has its own rules for validation, and the configuration you use may impact how you define your AppManager users and roles. As an example, if you are using Integrated security then the basis for all user rights is tied to the user's network name and account permissions. You use the Windows NT/2000 User Manager to create groups and individual user accounts. You then can map those groups and users to SQL Server logins (using the SQL Server Security Manager in SQL Server 6.x or the SQL Enterprise Manager in SQL Server 7.0). You then use the SQL Enterprise Manager to set the specific database permissions for the SQL users and logins you've created.

You then can use the AppManager Security Manager to authorize the SQL logins that should have access to AppManager, and to set the specific rights and roles for each login. When users log on to the AppManager repository with their Windows NT user account name and password, they bring with them the privileges assigned to that account, mapped to SQL Server permissions, and finally mapped to the specific

AppManager rights you've assigned. Thus, security is actually achieved through layers that are transparent to the ordinary user.



On the other hand, if your SQL Server uses the **Standard** security mode in SQL Server 6.5, there's no dependence on Windows NT users and groups and you can add new SQL logins directly from the AppManager Security Manager.

In addition to the security mode, you should be familiar with:

- SQL Server database groups (for example, the default `admin` and `public` groups) and assigning SQL logins to groups
- Special accounts (for example, the system administrator or `sa` account, and the database owner or `dbo` account)

As an example, users who are part of the SQL Server `admin` group are allowed to run certain Knowledge Scripts that users in other SQL Server groups, such as `public`, are not allowed to view or run. Other Knowledge Scripts require the `sa` account and password to run.

Because of the potential for these interrelationships, it's important that you review your SQL Server security mode, your SQL Server logins and groups, and in some cases your Windows NT user accounts and groups.

Once you understand the underlying security, managing access to AppManager programs is a relatively simple process.

■ ■ ■ ■ ■ ■ ■ ■

Note AppManager treats Mixed Mode security as Standard security. If you use Mixed Mode, you can't base logins and security on the Windows NT/2000 user name.

Understanding user access to AppManager

To work with AppManager, every user must have a valid login ID and password for the SQL Server where the AppManager repository database is running. This SQL Server login name must be mapped to a user name that is a member of a group with permission to access the repository database.

By convention, the SQL Server login ID and user name for a database are the same. However, it is also possible to create a user name for the AppManager repository database that is not the same as the SQL login name or to create aliases for a SQL login name. This situation can create some confusion for users and administrators. One way to minimize the confusion and distinguish between these names is to remember that:

- The SQL login ID is essentially permission to access the SQL Server on a particular computer and it uniquely identifies a SQL Server user.
- The SQL Server user name is a name by which a user to known to a specific database and the name only needs to be unique within a database. The user name, however, must always be assigned to a login ID to allow the user to access a database. The abilities a user has within a database depend on the permissions granted to the user name, and the permissions granted to any groups the user name is a member of.
- AppManager always uses the SQL login ID as the **AppManager user name**. Therefore, the SQL login name is what AppManager users type in the **Name** field of the AppManager Logon dialog box.

Setting up AppManager users

When you first install AppManager, only the `sa` login account is defined as a valid **AppManager user** with permission to log in to the AppManager repository database. This restriction is intentional to allow you to selectively control which SQL Server logins should have access to AppManager and to establish security policies before making AppManager widely available. This means that, after installation, you need to start the AppManager Security Manager and log in to the repository using the `sa` account to set up other AppManager users before any other users can begin to use AppManager programs.

The process of setting up **AppManager users** consists of using the AppManager Security Manager to select the SQL Server users that you want to authorize as AppManager users.

Adding existing SQL Servers users to AppManager

1. Click **Start > Programs > NetIQ AppManager > Tools & Utilities > Security Manager** and log in to the AppManager repository using the `sa` account and password (Standard or Mixed Mode) or an account that is mapped to `sa`-level privileges (Integrated).

   You are prompted to create the default AppManager roles (see "Understanding AppManager roles" on page 115). Although not required, NetIQ recommends you create the default roles as a starting point for managing security.

2. Click **Security > User Setup**. In the AppManager User Setup dialog box, the **SQL users** list shows only the existing SQL Server user names that have been granted access to the AppManager repository database. The **AppManager users** list shows the SQL Server users that you have authorized to log on to AppManager. Users are automatically assigned a default Security Role, which you can modify (see "Understanding AppManager roles" on page 115.)

3 Select user names from the **SQL users** list and click **Add**. Each selected SQL user name moves into the **AppManager users** list, which authorizes that user to log on to AppManager.

4 When you finish adding users, click **Close**.

Adding new SQL Server users from the Security Manager

If SQL Server is configured for Standard or Mixed security mode, you can use the AppManager Security Manager to add new SQL Server user names. AppManager automatically gives any new SQL user you create access to the AppManager repository, and creates a corresponding SQL login ID of the same name.

If the SQL Server is running in Standard or Mixed security mode:

1 Click **Start > Programs > NetIQ AppManager > Tools & Utilities > Security Manager** and log in to the AppManager repository using the `sa` account and password.

2 Click **Security > User Setup**.

3 In the AppManager User Setup dialog box, click **New SQL User**.

4 In the New SQL User dialog box, type the **SQL user name** you want to create.

5 Select a valid **SQL group** for the user to belong to. The built-in SQL Server groups are `admin` and `public`. Your organization may have additional groups. The default is `public`.

   If you want the new user to be an AppManager administrator, choose **admin**. Users with admin privileges can perform all Security Manager tasks except for creating new SQL Server users (only the `sa` login account can be used to create new users from AppManager). Users in the admin group can also run administrative Knowledge Scripts.

6 Type the **SQL login password** for the new SQL Server user.

7 Click **OK**.

Adding SQL Server users when using Integrated security

When your AppManager repository is on a SQL Server that is running in Integrated or Windows NT Authentication mode, the mapping of Windows NT group and user accounts to SQL Server logins requires additional attention because you are creating a chain of security relationships (Windows NT domain security to SQL Server and database security to AppManager security).

In the Integrated/Windows NT Authentication mode, you create users and groups in the Windows NT User Directory/User Manager. Each user who will be accessing SQL Server should have an account and be placed into appropriate domain or local groups on the repository server. You then grant those users permission to log in to SQL Server.

The basic requirement is to ensure all the users with the same security requirements are grouped into one unit, which can then be granted access to SQL Server.

The procedures for granting access to SQL Server and the AppManager repository vary depending on the version of SQL Server you are using. Refer to your Microsoft SQL Server documentation for specific information on how to give groups and users access. You may also want to visit the Microsoft web site for technical papers that discuss implementing security or work with your NetIQ technical representative to identify your security requirements.

Understanding AppManager roles

The AppManager user name and password is sufficient to determine who can log on to AppManager. However, in addition you frequently need to limit what individual users can do within AppManager. For example, depending on your organization's

needs, you may not want some users to be able to start and stop jobs. Instead, you may want to limit those users to viewing events or running reports.

To control what users can see and do with AppManager, you define **AppManager roles** and then assign each AppManager user a role. An AppManager role is a collection of rights and privileges called a **security profile**. For example, roles can:

- Control access to basic AppManager functions, such as whether users can run Knowledge Scripts, acknowledge and close events, or modify the TreeView.
- Determine whether users can work with AppManager programs such as the Operator Console, Distributed Event Console, and the Report Manager.
- Control which views users can see in the Operator Console and Operator Web Console.

AppManager provides three pre-defined roles. When you start the Security Manager for the first time, AppManager asks if you want to generate these predefined roles. The three predefined roles are Administrator, Standard, and Read-Only User. These predefined roles have the following default security profiles.

Security role:	Allows:	Does not allow:
Administrator	■ Access to all AppManager views ■ Functional rights to perform all AppManager tasks ■ Access to all AppManager components	■ Creation of SQL users (only `sa` can create new users)

Security role:	Allows:	Does not allow:
Standard User	■ Access to all AppManager views ■ Functional rights to perform all AppManager tasks ■ Permission to start the Report Manager and generate reports	■ Access to the Repository Browser ■ Access to the Distributed Event Console ■ Permission to change report properties ■ Access to the Icon Manager
Read-Only User	■ Access to all AppManager views ■ Permission to start the Operator Console ■ Permission to start the Report Manager and generate reports	■ Functional rights to perform AppManager tasks (including starting jobs, stopping jobs, and responding to events) ■ Access to the Repository Browser ■ Access to the Distributed Event Console ■ Permission to change report properties ■ Access to the Icon Manager

Although the predefined AppManager roles are optional, you must create at least one AppManager role before you can set up AppManager users (every AppManager user is assigned to a role). Once created, you can modify the rights for the predefined roles to suit your needs.

You can also create custom roles to use along with the predefined roles or as an alternative to the predefined roles. In most cases, the custom roles reflect the types of activities that a group of AppManager users perform. For example, you may want to set up a custom role for your Exchange administrators, or create a custom role for managers who only want to access graphs and reports.

Special administrative privileges

In addition to the Administrator role you can create and manipulate through the Security Manager, AppManager recognizes a special class of users that have broader administrative privileges. This special class of administrative privileges is controlled through the SQL Server privileges that are granted to the `sa` login and the AppManager repository's `admin` group.

The special privileges that are granted to users who log in to AppManager using the `sa` login or logins that are part of the `admin` group are:

- The ability to use the Security Manager to set up AppManager users, create and assign AppManager roles, and store sensitive information needed by Knowledge Scripts.
- The ability to view and run administrative Knowledge Scripts. These Knowledge Scripts typically perform tasks that you want to carefully control and restrict access to. For example, the Knowledge Script Action_RebootSystem allows AppManager to shut down and restart a computer in response to an event. It is only visible to users in the `admin` group. Most of the Knowledge Scripts that require special administrative privileges appear under the AMAdmin and NTAdmin tab of the Knowledge Script pane. Knowledge Script developers identify which Knowledge Scripts require administrative privileges using the `AdminOnly` parameter. (For more information about the `AdminOnly` parameter and custom Knowledge Script development, see the *NetIQ AppManager Developer's Guide*.)

If you log in to AppManager with the System Administrator (`sa`) login ID, you can perform all Security Manager tasks, which includes being able to create new SQL Server users from within the Security Manager. You can also run all administrative Knowledge Scripts.

If you log in with a login ID that has been set up in SQL Server as a member of the `admin` group for the AppManager repository database, you cannot create new SQL Server users, but you can perform all other Security Manager tasks and can run all administrative Knowledge Scripts.

Storing security information for Knowledge Scripts

Some Knowledge Scripts require sensitive information that you may not want users to specify explicitly as a job parameter or you may not want users to be aware of. For example, some SQL Knowledge Scripts need a password to log in to the SQL Server on the computer where the job runs. With the Security Manager, you can store a SQL login ID and its corresponding password for the SQL Server on that computer. The password is stored in the repository in an encrypted format. When running a SQL Knowledge Script on that SQL Server, the AppManager user specifies the login ID that was previously stored in the repository. The user needs to know only the login ID; AppManager retrieves the encrypted password for that login ID from the repository and then runs the Knowledge Script.

The Security Manager also provides a way to store sensitive information needed by custom Knowledge Scripts developed by your organization.

You store information of this type for specific computers. To store information needed by a Knowledge Script:

1 Expand the list of **Computers**.

2 Select the computer for which you want to enter information.

■ ■ ■ ■ ■ ■ ■ ■

3 Depending on the information you want to enter, click an appropriate tab:

Click this tab:	To do this:
SNMP	Set the community string that Knowledge Scripts should use when communicating with managed clients through Simple Network Management Protocol (SNMP).
Exchange	Set the Exchange profile name and mailbox alias that AppManager for Exchange Knowledge Scripts should use on the selected computer.
Oracle	Set and edit user names and passwords that some Knowledge Scripts need to log in to a database on an Oracle database server.
SAP	Set up SAP R/3 login information and an AppManager for SAP R/3 configuration name to run SAP Knowledge Scripts on the selected computer.
SMS	Set the `sa` password for an SMS SQL Server that AppManager for SMS Knowledge Scripts should use on the selected computer.
SQL	Select the SQL Server logins that are allowed to run SQL Knowledge Scripts on a remote computer.
Custom	Set custom security information for AppManager to check before running a Knowledge Script. **Note:** Custom information is not stored encrypted in the AppManager repository.

4 Enter the appropriate information.

5 Click **Apply**.

In most cases, the Security Manager can't verify the information you enter. For example, if you type an SNMP community string incorrectly, the Security Manager cannot detect this, and your SNMP Knowledge Scripts will fail. If you change passwords or other information being stored, remember to update the Security Manager with the new information as well.

Note Normally, the initial values for sensitive information, such as passwords, are set when you install the AppManager agent on a

computer (either automatically by the setup program or based on your input). These initial values are automatically stored in the AppManager repository. You can modify or reset the values at any time using the Security Manager.

Setting the SNMP community string

Some Knowledge Scripts communicate with managed clients through Simple Network Management Protocol (SNMP). As an example, most Knowledge Scripts that monitor hardware resources use SNMP to get information about a computer's current status.

SNMP requires a community name to communicate with a specific managed client. You use the Security Manager to store the community name in the AppManager repository for each computer in your network that requires this information for monitoring purposes. If you don't specifically enter a community name for a computer, the default community name of public (essentially "no security" is used).

If AppManager attempts to run a Knowledge Script on a computer without the correct community name, the Knowledge Script fails and an event is sent to the Operator Console.

In practice, you may have environment with twenty computers you are monitoring for hardware problems. Assume that ten of these computers are Compaq servers with Compaq Insight Management Agents installed and ten are Dell servers with the OpenManage agent (or HIP) installed but only three—MONSTER, GREEN, and FLIP—are critical enough that you protect them with a community name (the others still use the default community name of public). In the Security Manager, you only need to enter the proper community name for those three servers.

Setting Exchange profile and mailbox names

Some Knowledge Scripts for Exchange require a profile and mailbox alias for sending and receiving test mail messages. You use the Exchange tab in the Security Manager to store or change this information in the AppManager repository.

Once stored, AppManager users don't need to enter profile or mailbox information when they run Exchange Knowledge Scripts. AppManager gets the information directly from the repository.

Setting Oracle user names and passwords

Some Knowledge Scripts that run on an Oracle database require a user name and password to log in. You use the Oracle tab in the Security Manager to store or change this information in the AppManager repository. You can add user names and passwords to the AppManager repository, change passwords for existing user names, or remove user names.

Setting a configuration name for SAP Knowledge Scripts

Some Knowledge Scripts for SAP R/3 require an AppManager for SAP (AMSAP) configuration name to run. The configuration name maps to an SAP R/3 login on an SAP R/3 system and is used to simplify what a user needs to enter when running SAP Knowledge Scripts. You use the SAP tab in the Security Manager to store or change the SAP R/3 login information in the AppManager repository.

Setting the SMS SQL Server login

Some SMS Knowledge Scripts require an SMS SQL Server `sa` login to run. You use the SMS tab in the Security Manager to store or change the password for the SQL Server `sa` login for the SMS SQL Server. In this case, the Security Manager does validate the information you enter.

Setting SQL Server logins and passwords

Some Knowledge Scripts for SQL Server require a SQL Server login and password to run. Through the Security Manager, you can select which SQL Server login accounts are allowed to run SQL Knowledge Scripts on a computer and store the passwords for those logins. You can also update the SQL Server password of accounts that are already authorized to run SQL Knowledge Scripts. In this case, the Security Manager does validate the information you enter.

Maintaining the repository

The AppManager repository is a Microsoft SQL Server database. As with any database, there are a number of routine procedures you need to perform regularly to maintain the database and ensure database integrity.

This chapter provides an overview of how AppManager uses SQL Server, and describes some of the key activities for maintaining the database. For more detailed information on configuring and maintaining SQL Server databases, see your Microsoft SQL Server documentation.

Note Procedures and examples in this chapter illustrate how to perform the tasks described using Microsoft SQL Server 6.5. If you are using another version of SQL Server, such as SQL Server 7.0, there are variations in how tasks are handled and how terminology is used.

How AppManager uses SQL Server

The AppManager repository is a Microsoft SQL Server database that includes important tables for storing your data, stored procedures to manipulate or capture data, and custom scheduled tasks to perform routine operations.

What are the key tables

AppManager stores a great deal of information about your environment in the repository, including details about the computers you are monitoring, the jobs you are running on each computer, the job properties you've set, the events that have

■ ■ ■ ■ ■ ■ ■ ■

been raised, and the data that has been collected. The most important tables you may need to be familiar with are:

Table name	Type of information
Event	Basic event identification, including the Knowledge Script and job that raised that raised the event, the computer where the event was detected, the event severity and the short event message. Linked to the EventDetail table by the EventID key field.
EventDetail	Event properties such as the detail message, comment, and any action that was initiated. Linked to the Event table by the EventID key field.
DataHeader	Data stream identification and graphing properties, job ID and interval, computer where the data is being collected, and the minimum and maximum values returned. Linked by the DataID key field.
Data	Data point values, timestamp, and the data short message and detail message. Linked by the DataID key field.
ArchiveDataHeader	Data stream identification and graphing properties, job ID and interval, computer where the data is being collected, and the minimum and maximum values returned for archived data. Linked by the DataID key field.
ArchiveData	Data point values, timestamp, and the data short message and detail message for archived data. Linked by the DataID key field.
Job	Job identification and properties.

What stored procedures do

Stored procedures are used to handle many different operations within AppManager. For the most part, you do not need to use or view any of the stored procedures directly. They serve as the backbone for creating jobs, handling events, managing data, and producing reports. In fact, most stored procedures are encrypted and cannot be viewed even with `sa` or database owner privileges.

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 (`rp*`) and can be useful if you are creating custom reports. If you are creating custom reports, you can use existing report stored procedures as templates that you copy and modify to create new reports.

Because they perform so many key operations it's important that you do not delete stored procedures and that you periodically back them up as part of your regular database maintenance.

What scheduled tasks do

Scheduled tasks are SQL stored procedures that run at regular intervals to perform routine maintenance. Depending on the specific task being performed, the frequency at which scheduled tasks run can impact your database performance or efficiency. The key AppManager scheduled tasks include:

Scheduled task	What it does
NetIQ daily QDB	Performs internal maintenance operations once a day at 0300.
NetIQ minutely QDB	Calculates the number of current data points collected. This value is displayed in the Operator Console, and by default, the count is updated every 17 minutes. You may want to adjust this interval to more accurately reflect the current points count if you are monitoring when the latest data is collected. However, be aware that this task can be very expensive.
NetIQ hourly QDB	Performs internal maintenance operations once every two hours.
NetIQ PurgeData QDB	Purges data points from active data streams. By default, this task runs every 31 minutes to keep data streams efficient and manageable. You have the option of moving purged data to archive tables by setting a repository preference.

In most environments, there's no reason to change the default schedule for tasks. Changing the schedule can impact database performance or cause other problems. It is recommended you

contact Technical Support or your Implementation consultant before changing scheduled tasks.

Checking your SQL Server configuration

Depending on the size of your planned deployment, how your AppManager components are distributed, and the type of SQL Server security you are using, you may need to optimize SQL Server to handle the data storage and retrieval that AppManager performs.

Tuning configuration options

SQL Server includes many database configuration options for tuning your environment. Although the specific settings you may need vary according to the size and complexity of your deployment, for AppManager, the configuration options you are most likely to need to review are:

Configuration for	Setting and suggestions for changing
Hash buckets	Hash buckets are used to speed access to a given 2 KB page in the SQL Server data cache. Each hash bucket uses 8 bytes of RAM. Because hash buckets use little memory for what you gain in performance, it is better to over-allocate hash buckets rather than under-allocate. In general, you should set the number of hash buckets using a ratio of one hash bucket for every two to three SQL Server 2 KB pages you have allocated for SQL Server memory. For servers that are not running SQL Server 6.5, Enterprise Edition, with over 1 GB of RAM, the maximum value should be set to 265003 hash buckets.

Configuration for	Setting and suggestions for changing
Locks	SQL Server lock structures are 28 bytes and lock owner structures are 32 bytes, sp_configure increases both of these structure allocations at the same time under the generic label of 'locks'. If several users have the same page locked, one lock structure is needed plus a lock owner structure for each user. Since the internal memory cost of locks is relatively low it does not hurt to overestimate. When SQL Server runs out of locks, it doesn't show up as a performance hit. SQL Server will provide an error message ("SQL Server has run out of LOCKS") indicating that the number of locks needs to be increased. It is best to avoid the error message by setting locks higher on busy systems. In general, you can calculate the value for locks by estimating (user connections) x (max number of tables simultaneously accessed by any user at the same time) x (max rows per table accessed by any user at the same time). This estimation assumes no lock escalation and much more data page activity than is likely, but is a good starting point for configuration.
Memory	Sets the minimum and maximum RAM allocated to SQL Server. 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 NT to page frequently. Keep in mind that Windows NT may need substantially more memory to support a large-scale SQL Server environment that has 2 GB of memory and hundreds of SQL threads. When you configure the memory for SQL Server, be sure to leave enough RAM for Windows NT. **Note:** Performance improvement is typically small for incremental changes in data cache. For example, if a 500 megabyte data cache gives a 90% cache hit ratio, going to a 550 megabytes may only give 91%. Thus, SQL Server performance may not visibly improve. In configuring memory, you should also consider 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 (so that you leave enough RAM for Windows NT and NetIQms) than if the server were a dedicated SQL Server.

Configuration for	Setting and suggestions for changing
Open objects	Sets the maximum number of database objects that can be open at one time on SQL Server. Open objects are used for all SQL Server temporary objects in tempdb as well as all persistent objects in the user databases. It is better to overestimate than underestimate this value. A typical configuration in an active environment is 5000 open objects. If you are collecting and purging a large amount of data, you should increase the number of open objects, for example, to 20,000, or even 50,000. **Note:** Because open objects consume memory, increasing this value can make it necessary to increase the amount of memory dedicated to the server. 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.
Procedure cache	Sets the maximum portion of RAM allocated to SQL Server's procedure cache. By default, 30% of SQL Server memory is set aside as SQL Server procedure cache. On servers with a large amount of RAM, this value may be significantly larger than necessary. Reducing the procedure cache percentage increases the SQL Server data cache, thereby improving performance. In most environments, the procedure cache uses approximately 30 MB of memory. You can use this value as an estimate and compare it to the RAM on your server to calculate the percentage of RAM accommodates 30 MB. For example, if the server has 300 MB of RAM, you could set this value to 10 (10%) to reserve the 30 MB of memory.
User connections	Sets the maximum number of simultaneous connections to SQL Server allowed. Typically, 80 to 100 user connections are required. The management server service, NetIQms, uses between 40 and 60 connections. Each console simultaneously connecting to the repository uses from one to five connections. If your organization has several console program connecting to the repository at the same time, you may need to increase this value. For example, if, on average, you have 12-15 console program simultaneously accessing the same repository, you would increase the user connections to a value within the 150 to 200 range (depending on the amount of buffer you want in the case that any connections get blocked and have to time out).

Configuration for	Setting and suggestions for changing
Tempdb	Sets the size of the tempdb database in MB. The recommended minimum size is 10 MB. You may need to increase this value if you need to adjust the overall size of the AppManager repository to accommodate a large number of events and collected data points. See “Sizing the AppManager repository” on page 38 for more information. **Note:** If you have a server with a significant amount of memory available, you may want to increase the size of tempdb and put the tempdb in RAM for faster access of sorted data. This configuration can significantly improve the time it takes to run reports.
Truncate log on checkpoint	Specifies whether the transaction log is truncated each time the CHECKPOINT process occurs. You should turn this option off if you intend to do incremental backups.

Note If you are using SQL Server 7.0, this may not be necessary because SQL Server 7.0 can dynamically tune these values, as needed. Check whether you have the dynamic tuning option set.

Configuring network communication for SQL Server

In addition to configuration options for SQL Server and the AppManager repository database, use the SQL Client Configuration Utility to set the default network configuration method. At a minimum, you need to install the Named Pipes protocol as an accepted network communication library. Installing other network protocols, such as TCP/IP Sockets, is optional.

Note SQL Server must already be configured to support the communication method you select with the SQL Client Configuration Utility.

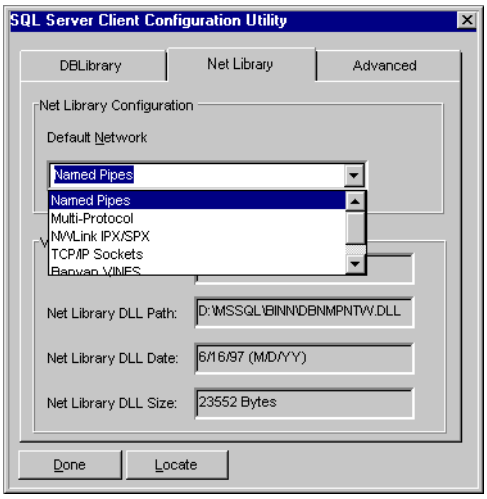
The network setup for the SQL Server enables you to define multiple protocols for communication between your SQL clients and the SQL Server.

- If you are using **Standard** security, set your SQL Server network protocol to use Named Pipes and TCP/IP Sockets. Named Pipes should be the default protocol. (Do not remove Named Pipes from the supported protocol configuration. If desired, you can add Multi-Protocol to the protocol configuration.)
- If you are using **Integrated** or **Mixed** security, set your SQL Server network protocol to use Named Pipes as the communication protocol.
- If you are using Mixed security and you place your AppManager web management server on an IIS server in a different, untrusted domain than the repository server, then you also need to set your SQL Client configuration to use TCP/IP sockets.

To check your current configuration:

1. Start the **SQL Server Client Configuration Utility**.
2. Click the **Net Library** tab.

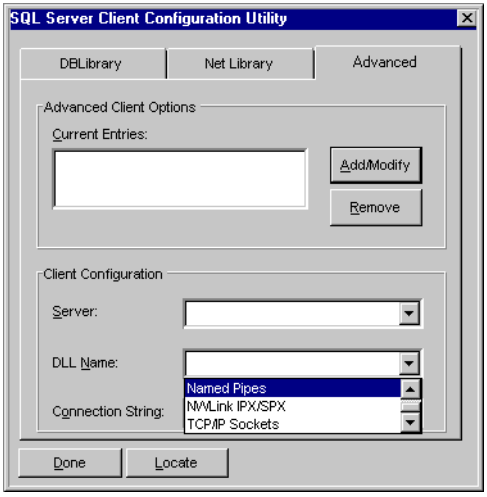
3 Check the **Default Network** list for the network protocols enabled. For example:



To modify the configuration:

1 Click the **Advanced** tab.

2 Select **Named Pipes** from the DLL Name list.



Creating data devices for expanding the database

When you install the AppManager repository you set the initial size for the database and its associated log. If you are using SQL Server 6.x, NetIQ recommends you create additional data devices into which you can grow your database, as needed. (If you are using SQL Server 7.0, this isn't necessary because SQL Server 7.0 dynamically increases the size of databases, as needed.)

Creating additional data devices before a significant amount of information is written to the data drive helps to ensure that the devices are comprised of contiguous blocks and maximizes the I/O performance of your AppManager database from a disk access perspective.

To create additional data devices for AppManager:

1. Start the SQL Server Enterprise Manager
2. Select the where the AppManager repository is installed.
3. Click **Manage > Database Devices**.
4. Click the **New Device** command button.
5. Type a name for the device. For example, `QDBData2`.
6. Change the location for the data device, if necessary.
7. Set the **Size** for the data device in MB. In setting the size, keep in mind that making the data device too large will require added memory and take longer to backup, check for consistency, and restore. In general, data devices under 2 GB are easier to manage.
8. Click **Create Now**.
9. Repeat Step 3 through Step 8 for any additional devices.

Backing up the repository

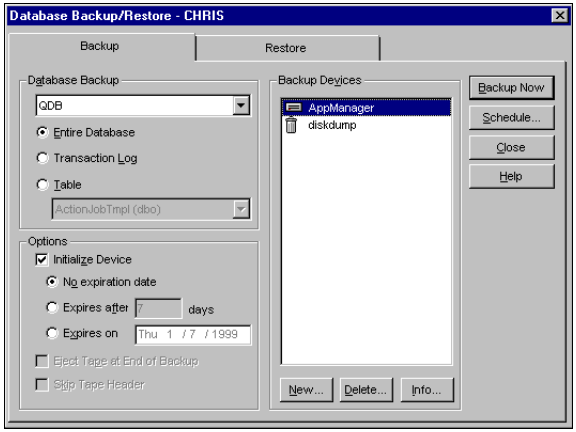
To ensure the integrity of the data stored in the AppManager repository, the repository database, SQL tasks, and extended SQL stored procedures should be periodically backed up.

There are two ways to back up the repository database:

- Using SQL Enterprise Manager
- Running the SQL_RunSql Knowledge Script

Backing up the repository using SQL Enterprise Manager

1. In SQL Enterprise Manager, select the server on which the AppManager repository resides.
2. Click **Tools > Database Backup/Restore**.
3. In the Database Backup/Restore dialog box, select the name of the AppManager repository (the default is **QDB**) in the **Database Backup** list.



4. Select **Entire Database**.
5. Select an item in the **Backup Devices** list.

6 To create a backup device, click **New**. Fill in the dialog box options and click **Create**.

7 (Optional) If you want to discard existing backups on the selected backup device, select **Initialize Device**. If you leave **Initialize Device** deselected, the backup is appended to the backup device.

8 Click **Backup Now**.

9 In the Backup Volume Labels dialog box, click **OK**.

10 When the backup procedure is completed, click **OK**.

Backing up the repository with SQL_RunSql

To automate and simplify the backing up the AppManager repository, you may want to use SQL_RunSQL Knowledge Script. This Knowledge Script enables you to create a job that backs up the repository at scheduled intervals.

1 In the AppManager Operator Console or Operator Web Console, start an SQL_RunSql Knowledge Script job on the SQL Server where the repository you want to back up is located.

   For information on starting jobs, see the *NetIQ AppManager User's Guide*.

2 In the Values tab in the Knowledge Script Properties dialog box, either:

   - Specify a SQL script that contains all the necessary commands for backing up the AppManager repository, msdb, and master databases (set **Load SQL Script** to **y** and, for **SQL Script File**, enter the full path to the script file)
   - Specify a SQL statement (for **SQL Statement**, enter the statement).

Here is an example of a SQL statement for backing up the data in one database:

```
dump database <QDB> to <dump_device>
```

where `<QDB>` is the name of the AppManager repository and `<dump_device>` is the name of the backup device.

For information about SQL scripts and SQL statements, see your Microsoft SQL Server documentation.

For more information about using the SQL_RunSql Knowledge Script, see the *NetIQ AppManager Knowledge Script Guide* or select the script in the Knowledge Script pane in the Operator Console and press `F1`.

Backing up tasks and stored procedures

In addition to the database, the AppManager repository includes SQL Server **tasks** and **stored procedures**. SQL Server tasks are used to perform routine maintenance and updates at scheduled intervals. Stored procedures are used in generating AppManager reports.

In addition to the AppManager database (for example, QDB), you should periodically back up your SQL tasks and stored procedures.

- To back up SQL tasks, follow the steps described in Backing up the repository using SQL Enterprise Manager, except select **msdb** in Step 3, then continue with Step 4 through Step 10.
- To back up the extended SQL stored procedures, follow the steps described in Backing up the repository using SQL Enterprise Manager, except select **master** in Step 3, then continue with Step 4 through Step 10.

Backing up the master database

In SQL Server, the **master** database holds all of the device configuration information and SQL logins. Therefore, you should

also periodically back up the master database, especially if you add data devices, databases, or SQL logins or change any of your SQL server configuration options.

Scheduling backups

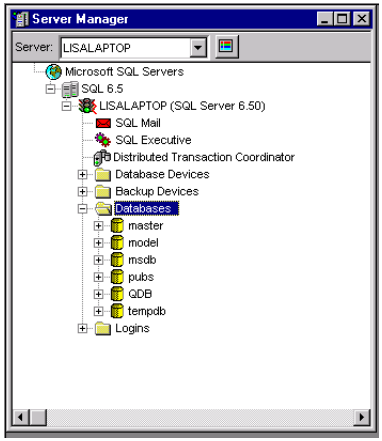
You can use SQL Enterprise Manager to create a scheduled task for backing up the repository or use the Schedule tab in the Knowledge Script Properties dialog box for the SQL_RunSql Knowledge Script to set a regular interval for backups. The frequency at which you perform full backups depends on the nature of your environment, but it general it is recommend that you perform a complete backup every week at a minimum.

Restoring a repository from a backup

If you need to restore your AppManager repository from the backup, follow these steps:

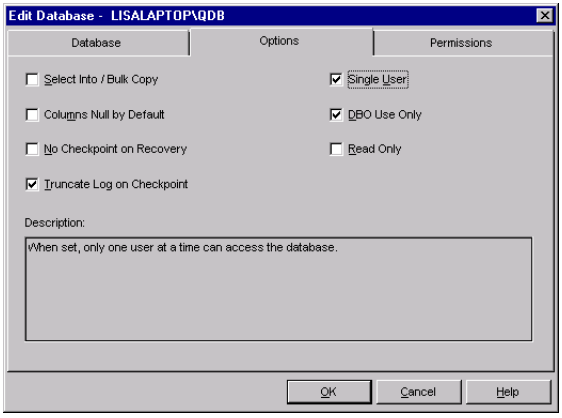
1. Close any open console programs connecting to the repository you want to restore.
2. Open the Services Control Panel and stop the `NetIQms` service that communicates with the repository server. You may need to wait a few moments to allow the service to stop and for connections to the repository to be terminated.
3. If you have any other programs or users connecting to the repository, close those connections. The restore operation requires exclusive use of the system.
4. Open SQL Enterprise Manager.

5 In the Server Manager, expand the repository server and the Databases folder to view the AppManager repository database (for example, QDB).



6 Right-click on AppManager repository database and select **Edit** from the right-click menu.

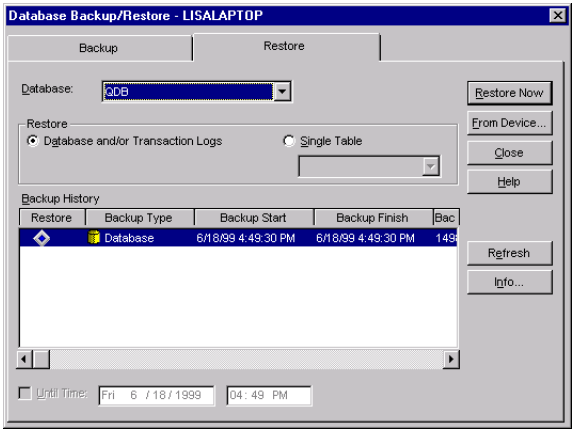
7 In the Edit Database dialog box, click the **Options** tab and select the **Single User** option. This option ensures the System Administrator (sa) user has exclusive use of the database. Click **OK**.



8 Right-click on AppManager repository database and select **Backup\Restore** from the right-click menu.

9 In the Database Backup/Restore dialog box, click the **Restore** tab.

10 Check that the database listed is the AppManager repository and the backup is the backup you created most recently, then click on **Restore Now**. For example:



After the database is restored, follow these steps to return AppManager to a working state:

1 Right-click on AppManager repository database and select **Edit** from the right-click menu.

2 In the Edit Database dialog box, click the **Options** tab and deselect the **Single User** option, then click **OK**.

3 Open the Services Control Panel and start the `NetIQms` service that communicates with the repository server.

4 Log on to the AppManager repository with any console program or utility.

Note If you have multiple devices, you must restore all of the devices to completely restore the database. Partial restoration is not permitted.

Removing archived data and old events

In the AppManager repository, the most current data points and data stream information is stored the **Data** and **DataHeader** table. If you elect to archive data, periodically older information is moved to the **ArchiveData** database table. Event information is saved in the **Event** table.

Over time, these tables will increase in size and may threaten database performance and consistency. As with other database applications, you should periodically remove archived data and event information that you no longer need.

To simplify this periodic maintenance, there are two specialized stored procedures available for you to use:

- DeleteOldData
- DeleteOldEvent

These stored procedures enable you to specify the number of days of days of data to keep in the repository. For example, to delete all of the data from the `ArchiveData` table that's more than 100 days old, you would run the following SQL command:

```
DeleteOldData 100
```

Note This stored procedure imposes heavy SQL load. Run it only during off-peak hours and when the NetIQ PurgeData task is not running. If you have accumulated a large amount of data (for example, 2 million rows or more) before running this stored procedure for the first time, gradually decrease the number of days back to delete. For example, if you have 120 days worth of data, run `DeleteOldData 115`, then `DeleteOldData 110`, then `DeleteOldData 105`, until the desired amount of data has been deleted.

Because of the way archive data tends to grow, it is more critical to perform periodic maintenance for data than events. However, you should also periodically remove obsolete event information. To perform a similar clean-up operation for the Event table, you would type:

```
DeleteOldEvent 100
```

This stored procedure deletes **Closed** child events where the LastOccurTime is older than the number of days you specify. It also removes the parent event if the last children event is removed and modifies the occurrence information for the parent event.

Note This stored procedure imposes heavy SQL load. Run it only during off-peak hours.

Although it is recommended that you use these stored procedures to remove data and events you no longer need, you can also use standard database utilities — such as `isql` and SQL Enterprise Manager — to view, export, or remove archived data (for example to remove data based on some other criteria such as the `DataID`). For example, to remove all records for the data stream with `DataID` 2, you can use a SQL statement like this:

```
DELETE dbo.ArchiveData WHERE DataID = 2
```

You should, however, use caution in performing database operations with SQL statements such as `DELETE` or `TRUNCATE TABLE`.

Expanding the size of an existing repository

To expand the size of an existing AppManager repository, you can first increase the size of the current database device or expand the repository into additional database devices.

Note If you are using SQL Server 7.0, expanding the existing repository isn't necessary because SQL Server 7.0 dynamically increases the size of databases, as needed.

Expanding the current database device

1. In SQL Enterprise Manager, expand the **Database Device** folder on the server on which the AppManager repository resides.

   To expand the list, click the plus sign (+) next to an item.

2. Double-click the name of the AppManager repository database data device.

   The data device name is `<DBNAME>Data`, where `<DBNAME>` is the name of the AppManager repository. For example, if the name of the repository is QDB (the default), then the data device name is `QDBData`.

3. In the Edit Database Device dialog box, select a size that is larger than the current data device size. (The default the size of the AppManager repository data device is 100 MB.)

4. Click **Change Now**.

5. Expand the **Database** folder on the server on which the AppManager repository resides.

6. Double-click the name of the AppManager repository database (the default is **QDB**).

7. In the Edit Database dialog box, click **Expand**.

8. In the **Data Device** list, select the name of the AppManager repository data device. (See Step 2 on page 143 for information on the data device name.)

9. For **Size**, the number corresponds to the amount you increased the data device in Step 3 on page 143. You can use this number or specify a smaller number.

10. Click **Expand Now**.

Note If you need to expand the size of the log device, then repeat Step 8 (selecting the log device) through Step 10. The log device name is <DBNAME>Log, where <DBNAME> is the name of the AppManager repository. For example, if the name of the repository is QDB (the default), then the data device name is QDBLog.

Expanding into a new database device

1. In SQL Enterprise Manager, expand the server on which the AppManager repository resides.
2. Expand the **Databases** folder.
3. Double-click the name of the AppManager repository database, for example QDB, to display the Edit Database dialog box.
4. In the **Edit Database** dialog box, click **Expand**.
5. Select a device name from the **Data Device** drop-down list. For example, if you have created a data device called QDBData2, select this item from the list.
6. Click **Expand Now**.

Verifying the size of the AppManager repository

The query results are produced by using Microsoft SQL Server's TRANSACT-SQL statements. The following SQL statements may be issued using the Microsoft SQL Server Enterprise Manager, ISQL.EXE, or ISQL/W utilities.

If the database was expanded from it's original size, you will need to expand the new repository in the same segment order and size for both any data or log segments created. To determine the way the segments were created use the following command:

```
sp_helpdb 'qdb'
```

Put into the single quotes the name of your repository.

Something similar to the following should be returned.

```
Name db_size     Owner   dbid     created     status
QDB  450.00 MB   netiq   6      Aug 25 1998  trunc. log on chkpt.

device_fragments size         usage
QDBData        100.00 MB     data only
QDBLog         50.00 MB     log only
QDBData         50.00 MB    data only
QDBData         50.00 MB    data only
QDBData        100.00 MB    data only
QDBData         50.00 MB    data only
QDBData         50.00 MB    data only

devicesegment
QDBDatadefault
QDBDatasystem
QDBLoglogsegment
```

If you have more than the original database and log size, you will need to expand the new database in the same order by segment size. So in this case you would need to expand this new QDB database five times (once for each segment the QDBdata segment was expanded) in the order listed (50,50,100,50,50). In this example the original size of the QDB data segment was 100 MB and the log was 50 MB. The data segments added an additional 300 MB the data size.

Migrating the repository to another SQL Server

To migrate an existing AppManager repository from one SQL Server (the source) to another (the destination), you must first install the AppManager repository component on the destination SQL Server.

1. On the destination SQL Server, run the AppManager setup program.
2. Install the AppManager repository and any other AppManager components you want to reside on this SQL Server. For

example, you might want to install the AppManager management server.

Tip NetIQ strongly recommends installing the AppManager agent on the AppManager repository server. This facilitates database management and closed-loop monitoring of your AppManager environment.

3 When installing the AppManager repository, be sure that the size of the database and the data device are equal to or greater than (=>) those of the existing repository on the source SQL Server.

For more information about installing AppManager, see the *NetIQ AppManager Installation Guide*.

4 Stop the management server service, `NetIQms`, that communicates with the repository you are about to move.

5 On the server where the AppManager repository you want to move resides, start SQL Enterprise Manager and expand the **Databases** folder

To expand the list, click the plus sign (+) next to an item.

6 Select the name of the AppManager repository database; right-click and then click **Transfer**.

7 In the Database/Object Transfer dialog box, select the **Destination Server** and **Destination Database**.

The destination server is the one on which you installed the AppManager repository in Step 1 through Step 3.

8 Select all options in the Transfer Options group. In the Advanced Options group, deselect **Transfer All Objects** and then click **Choose Objects**.

9 In the Choose Objects to be Transferred dialog box, select **All Tables**, **All Views**, **All Defaults**, **All Rules**, and **All User-Defined Datatypes**. Then click **OK**.

10 Click **Start Transfer**.

Restarting the management server

When the transfer is completed restart the management server service, `NetIQms`, and if necessary, point it to the new repository. If you moved the AppManager management server along with the AppManager repository to the new server, you only need to restart the service.

If you did not move the AppManager management server to the new destination server, then you must execute the `netiqms -r` statement from a Command Prompt using the following syntax:
`netiqms -r QDBMS:<QDBname>:netiq:<netiqpw>:<ServerName>`

where:

- `<QDBname>` is the name of the AppManager repository database on the destination server
- `netiq` is a valid SQL login on the repository server
- `<netiqpw>` is the password for the `netiq` SQL Server login account
- `<ServerName>` is the name of the destination SQL Server.

After running this command, restart the `NetIQms` service.

Restoring database triggers after migration

After the AppManager repository has been transferred to the destination server, the database triggers must be restored using a SQL script that can be obtained from NetIQ Technical Support. For information on how to contact Technical Support, see "Questions or suggestions? Contact us...." on page 12.

Restoring SQL users after migration

The SQL Server `syslogins` table is unique on each SQL Server. Therefore, when the AppManager repository is moved to a different server, the SQL logins IDs from the source server

needed to be mapped to the new destination server after the move.

After you have migrated the repository to the new server, run the following script to update the SQL users:

```
UPDATE dbo.KPC SET CreateUserID=1 WHERE
SUSER_NAME(CreateUserID) IS NULL
UPDATE dbo.KPP SET CreateUserID=1 WHERE
SUSER_NAME(CreateUserID) IS NULL
UPDATE dbo.KPS SET CreateUserID=1 WHERE
SUSER_NAME(CreateUserID) IS NULL
UPDATE dbo.KP SET CreateUserID=1 WHERE
SUSER_NAME(CreateUserID) IS NULL
UPDATE dbo.KPW SET CreateUserID=1 WHERE
SUSER_NAME(CreateUserID) IS NULL
UPDATE dbo.Views SET CreateUserID=1 WHERE
SUSER_NAME(CreateUserID) IS NULL
UPDATE dbo.Job SET SubmitUserID=1 WHERE SubmitUserID IS NOT
NULL AND SUSER_NAME(SubmitUserID) IS NULL
UPDATE dbo.Job SET StopUserID=1 WHERE StopUserID IS NOT NULL
AND SUSER_NAME(StopUserID) IS NULL
UPDATE dbo.Job SET PauseUserID=1 WHERE PauseUserID IS NOT NULL
AND SUSER_NAME(PauseUserID) IS NULL
```

Checking the integrity of the repository

Although AppManager checks data dependencies in the tables in the repository database, it's a good idea to regularly check the integrity of the database for early detection of possible problems, such as corruption of the database or its tables and database structure inconsistencies.

How regularly you should check database consistency depends on several factors, such as the number of jobs you run, events you generate, and data points you collect. In general, the larger your database, the more frequently you should check for problems. For most organizations, once a month is appropriate.

Note Database consistency processes can take several hours to run. The larger the database, the more time the processes will

take. Therefore, you should plan to run these processes during off-peak hours.

Here are a list of DBCC commands you can use. (Consult the *Microsoft Transact-SQL Reference Guide* for a detailed explanation of the syntax.)

Command:	Performance/ Effectiveness:	Description:
DBCC CHECKCATALOG	Fast/High	Checks the system tables for consistency and displays segment information.
DBCC NEWALLOC	Slow/High	Ensures that page allocation is correct, and that the page structure of the data and index pages are 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	Slow/High	Ensures that all pointers and data and index pages in a specified table are consistent and properly linked.
DBCC CHECKDB	Slow/High	Ensures that all pointers and data and index pages for all tables in the database are consistent and properly linked.

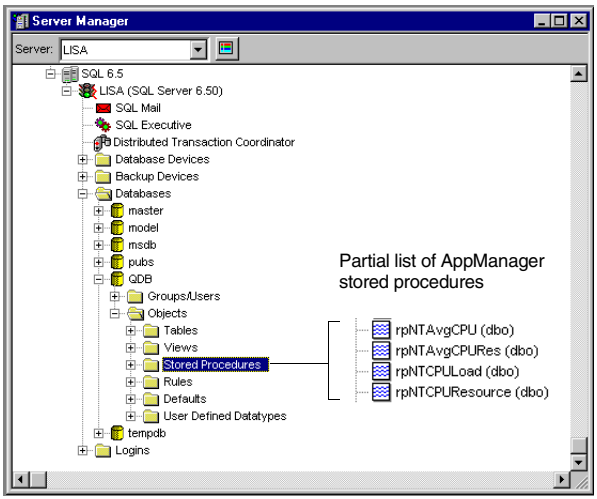
DBCC commands can be executed in any SQL query tool, such as SQL Enterprise Manager or `isql`.

Setting permissions for stored procedures

AppManager reports use SQL stored procedures. You can specify which users and groups can use individual stored procedures (and thus specific reports).

Note Changing permissions on some stored procedures may prevent AppManager from functioning properly.

1 In SQL Enterprise Manager, expand the **Stored Procedures** folder for the AppManager repository database (the default is **QDB**) on the server on which the AppManager repository resides.



To expand the list, click the plus sign (+) next to an item.

2 Select a stored procedure in the list.

3 Click **Object > Permissions**.

4 In the Object Permissions dialog box, you can set permissions in either the By User or By Object tabs.

Use this tab:	To:
By User	Select a user or group, and then set permissions for the stored procedures.
By Object	Select a stored procedure, and then set permissions for a user or group.

5 Select the settings you want. Then click **Set**.

6 When you're finished making changes, click **Close**.

Samples and worksheets

This chapter provides samples and worksheets for you to use and modify as you plan the AppManager deployment. These materials are intended as guidelines and aides to help you determine the information you need for a successful installation.

Note You should check with your NetIQ implementation consultant for online templates and tools you can use in planning your deployment.

Simplified sample project plan

Your project plan should identify project phases and the specific tasks for each phase. The plan should indicate when each task or activity occurs and whether they occur in sequence or in parallel. Activities should be linked to deliverables to define project milestones. In addition, you may want to spell out the responsibilities of project team members, and the time it will take to accomplish each task or each phase.

In the sample, specific time durations are blank. The program manager and project team determine the actual time to be allotted to each activity. The time needed for each activity can be influenced by a variety of factors, including company size, geographic location, or internal policies as well as project scope.

The following is a simple example of a project plan. In this example, milestones that close each phase are noted in bold text:

Phase	Tasks	Time	Team
Design			
	Assemble project team	1 day	JT: Project Mgr DD: ExchAdmin KM: EngSupport
	Collect and collate data on ■ physical and logical network ■ Windows NT/2000 domains/trusts ■ computer configuration Analyze the collected data		All
	Analyze monitoring objectives Identify repository and management server requirements		
	Document monitoring objectives and deployment plan		
	Review the deployment plan		
	Finalize and approve the deployment plan		
	Activities complete. Deliverables are: Completed worksheets Detailed environment document Project plan approval **Design phase complete**		
Testing	Install AppManager in test lab Run core set of Knowledge Scripts in the lab Generate, acknowledge, close events Uninstall AppManager	4 days	Install team: JT KM
Pilot	Stage the initial deployment to selected servers Resolve installation issues Resolve critical problems	3 weeks	
Expand	Roll out AppManager to additional servers		

Sample inventory worksheet

Many organizations find it helpful to use a worksheet to review the computer resources for the computers they plan to monitor or the computers where they plan to install AppManager components. These worksheets are relatively similar. The following is a simplified example of an inventory worksheet for AppManager components.

Computer	CPU	RAM	Disk	OS	ODBC	SQL DB	AppManager component
AQUA	Pent II	64MB	1.2 GB	4.0	ODBC32	Y	Repository
ZODIAC	Pent II	64 MB	110MB	4.0	ODBC32	N	Management server
SHASTA	Pent II	64 MB	80MB	4.0	ODBC32	N	Operator Console
MONTAGE	Pent II	32 MB	68MB	4.0	ODBC32	N	Web management server

The following is a simplified example of an inventory worksheet for managed clients.

Computer	CPU	RAM	Disk	OS	ODBC	What's monitored
TONGA	133	32 MB	30 MB	4.0	ODBC32	NT, Domino
HOPPER	168	64 MB	50MB	4.0	ODBC32	NT, Exchange
THATCH	230	64 MB	80MB	4.0	ODBC32	NT, CIM, Exchange
DELACROIX	300	32 MB	1.18GB	4.0	ODBC32	NT

Required permissions worksheet

The AppManager setup program requires access to various administrator and user accounts in order to install each of the AppManager components. In addition, some Knowledge Scripts require special user accounts or permissions. The following sample worksheet summarizes the accounts required for installation and to run the most common Knowledge Scripts.

Note that the list is not exhaustive, but is intended to illustrate the most common requirements and how a worksheet can help you plan out the accounts and permissions you need..

Computer	Component installed	What's required	Account to use
All	Any	Windows NT user account with local or domain Administrator privileges to use when running setup.	njadmin (aloha)
ROME	Repository	■ Password for the SQL Server `sa` login account. ■ The netiq SQL Server login account (created automatically) and password.	sa (apache) netiq (netiq)
ROME	Management server	Account for `NetIQms` to use (LocalSystem or a Windows NT user account) **Note:** A Windows NT user account is required for SQL Server 6.5 in Integrated security mode or SQL Server 7.0 in Windows NT Authentication mode.	LocalSystem
METZ	Console programs (all)	■ Password for the SQL Server `sa` login account to set up AppManager users in the Security Manager. ■ Account with permission to access the repository.	sa (apache) all IT accounts
BALI	Web management server	After installation, no special requirements.	
SANTERIA	Agent, NT managed object	After installation, no special requirements.	

Computer	Component installed	What's required	Account to use
MILANO	Agent, Microsoft Exchange Server managed object	■ Domain, user name, and password for a Windows NT account for `NetIQmc` to run as. ■ Account needs **Log on as Service** privileges for each domain that contains an Exchange Server to be monitored ■ Account must be an **Exchange Administrator** with the **Permissions Admin** role for the **Recipients** configuration level. ■ Account needs to be linked to the **Exchange profile** and **mailbox alias** name that AppManager will use (default naming convention is netiq_servername, netiq_Milano)	Domain: EURO User: gilbert Password: 8days
RIO	Agent, MSMQ managed object	Account access and password for the local MSMQ SQL Server `sa` login account.	sa (paolo)
ORLEANS	Agent, SQL Server managed object	■ Password for the local SQL Server `sa` login account (for installation). ■ SQL Server login account with permission to access the local SQL Server system tables (for monitoring).	sa (cestbon) lyon01 (leroi)
ZURICH	Agent, SMS managed object	■ SMS Administrator. ■ Account access and password for the SMS SQL Server `sa` login account.	sa (lausanne)

Sample management worksheet

Many organizations find it helpful to use a worksheet to plan out the basic management strategy and the baseline Knowledge Scripts they intend to use as part of the scoping process. The following is a simple example of this type of worksheet.

Knowledge Script	Description	Interval	Properties	Severity	Notify
General category					
ServiceDown	Determines if service is running.	5 min	y: restart	5 (restart fails) 25 (restart OK)	Chris
ServiceHung	Determines if a service is hung.	5 min	y: stop and restart	5 (restart fails) 25 (restart OK)	
URLContent	Content changes to specified URLs.	12 hours	URL list	8 (any change)	
NetServices category					
RASConnections	Average number of RAS connections.	30 min	Average > 50 connections per min	8 (threshold)	Wayne
Windows NT category					
CpuLoaded	Total CPU % and queue length.	5 min	y: collect data CPU: 90% CPU queue: 2	5	Admins4NT
FailedLogon	Failed logon attempts in interval.				
LogicalDiskIO	Logical disk IO activity.				
LogicalDiskSpace	Logical disk space usage.				
MemUtil	Physical and virtual memory, and paging files.				

Knowledge Script	Description	Interval	Properties	Severity	Notify
TopCpuProcs	Processes consuming the most CPU resources.				
TopMemoryProcs	Processes consuming the most memory.				

Choosing Knowledge Scripts

In general, Knowledge Scripts fall into one of three management tiers:

- The first tier of **Recommended** Knowledge Scripts that monitor the most important or most broadly applicable characteristics of the environment.
- The second tier of **Suggested** Knowledge Scripts that expand the monitoring environment in key ways by may not be applicable in all situations.
- The third tier of **Useful** Knowledge Scripts that monitor for very specific conditions, perform unique or infrequent tasks, or have a narrow focus for troubleshooting. Generally, these Knowledge Scripts are not run on a regular, ongoing basis.

This appendix summarizes some of the recommended and suggested Knowledge Scripts for different types of servers and applications. A small number of other Knowledge Scripts are highlighted because they are of particular interest, for example, because they are used in the generation of reports.

Note This appendix does not provide recommendations for all Knowledge Script categories. For more recommendations on selecting the Knowledge Scripts to run, you should consult NetIQ AppManager *Work Smarter* guides, *NetIQ AppManager Field Notes: Recommended Knowledge Scripts* available on the World Wide Web, the NetIQ Online Support web site, and your NetIQ implementation consultant. NetIQ also highly recommends you attend the NetIQ AppManager Administrator training class.

For all Knowledge Scripts, we recommend you review the online reference information or consult the *NetIQ AppManager Knowledge Script Reference Guide* for more information.

Windows NT operating system

The core set of Recommended Knowledge Scripts for monitoring Windows NT and Windows 2000 operation are described in "Running the recommended core Knowledge Scripts" on page 52.

The following table summarizes the recommended and suggested Knowledge Scripts for monitoring the basic components of the Windows NT or Windows 2000 operating system.

Recommended General and NT Knowledge Scripts	
General_EventLog	
General_MachineDown	
NT_CpuLoaded	
NT_MemUtil	
NT_LogicalDiskSpace	
NT_ServiceDown	
NT_TrustRelationship	
Suggested Knowledge Scripts	
General_AsciiLog	
General_Counter	
NT_NetworkBusy	
NT_PagingHigh	
NT_PrinterHealth	Recommended for file and print servers and may be useful for some application servers.
NT_PrinterQueue	Recommended for file and print servers and may be useful for some application servers.
NT_RunawayProcesses	
NT_SystemUpTime	

Other Knowledge Scripts you may find useful	
NT_TopCpuProcs	Useful for troubleshooting when you find CPU utilization is outside of normal operating values or when looking for Dr. Watson processes.
NT_TopMemProcs	Useful for collecting data on the memory usage of processes.

NetServices

The Knowledge Scripts you should run depend on the network services you have installed. The following table summarizes the recommended and suggested Knowledge Scripts for monitoring the different network services.

Windows Internet Name Service (WINS)

Recommended Knowledge Scripts for WINS
NetServices_WINSReplication
NetServices_WINSConfilct
NetServices_WINSFailure
Suggested Knowledge Scripts
NetServices_WINSHealthCheck

Dynamic Host Configuration Protocol (DHCP)

Recommended Knowledge Scripts for WINS
NetServices_DHCPLeases
Suggested Knowledge Scripts
NetServices_DHCPHealthCheck

Remote Access Service (RAS)

Recommended Knowledge Scripts for WINS
NetServices_RASConnections
Suggested Knowledge Scripts
NetServices_RASErrors

Domain Name Service (DNS)

Recommended Knowledge Scripts for WINS
NetServices_DNSSync
Suggested Knowledge Scripts
NetServices_DNSHealthCheck

Microsoft Exchange Server

The following table summarizes the recommended and suggested Knowledge Scripts for monitoring the Exchange Server. For more information about monitoring Exchange Server and suggestions about what Knowledge Scripts to run, see the *Monitoring Exchange with AppManager* Work Smarter Guide, available on the NetIQ web site.

Recommended Knowledge Scripts	
Exchange_Connectivity or Exchange_ResponseTime	Depending on your primary interests, it is recommended that you run one of these Knowledge Scripts. For most organizations, it isn't necessary to run both.
Exchange_ISDbSize	
Exchange_LogFileSize	
Exchange_ServicesDown	Monitors the up/down status of Exchange services, using the known order of dependency
Suggested Knowledge Scripts	
Exchange_TopNISPublicRes	

Exchange_TopNISPrivateRes	
Exchange_TopNReceivers	
Exchange_TopNSenders	
Exchange_MsgsThroughIMC	There are several Knowledge Scripts for monitoring Exchange message traffic.

Microsoft SQL Server

The Knowledge Scripts you should run for Microsoft SQL Server vary depending on how you are using SQL Server and your operational policies. The following table summarizes the minimum recommended and suggested Knowledge Scripts for monitoring SQL Server.

Recommended SQL Knowledge Scripts	
SQL_Accessibility	
SQL_CacheHitRatio	
SQL_DataSpace	
SQL_Errorlog	
SQL_LogSpace	
SQL_NearMaxLocks	
SQL_ServerDown	
Suggested Knowledge Scripts	
SQL_RunSql	Allows you to run virtually any SQL command or script automatically at regular interval. For example, you can use this Knowledge Script to handle daily database backups or to run the database consistency checker (dbcc) on a weekly schedule.
SQL_ServerThroughput	
SQL_UserConnections	
Other Knowledge Scripts you may find useful	
SQL_Bcp	
SQL_TopCpuUsers	

Recommended SQL Knowledge Scripts
SQL_TopIOUsers
SQL_TopLockUsers
SQL_TopMemoryUsers

Microsoft Systems Management Server (SMS)

The Knowledge Scripts you should run for Microsoft Systems Management Server vary depending on the version of SMS you are using and your interests. In addition, because SMS relies on Microsoft SQL Server, some of the most critical management tasks you will want to perform are actually achieved through monitoring SQL Server.

The following table summarizes the recommended and suggested Knowledge Scripts intended specifically for monitoring SMS:

Recommended SMS Knowledge Scripts	
SMS_NewHealthCheck	Useful for checking share to share connectivity. Only applies to SMS 1.x.
Suggested Knowledge Scripts	
SMS_AdNotReceived	Only applies to SMS 2.0.
SMS_AdProgramFailed	Only applies to SMS 2.0.

AppManager Admin (AMAdmin)

Most of the Knowledge Scripts in the AMAdmin category are used only infrequently to perform very specific tasks. However there are a couple that you want to use more frequently. The following table summarizes the recommended and suggested Knowledge Scripts for monitoring and managing AppManager itself.

Recommended AMAdmin Knowledge Scripts	
AMAdmin_DBHealth	Run regularly to help monitor the health of the AppManager repository.
Suggested Knowledge Scripts	
AMAdmin_SchedMaint	Useful for temporarily stopping jobs during regularly scheduled server maintenance.

Registry keys

This appendix describes many of AppManager's registry keys. The registry keys are listed in bold with key values listed beneath each key. For each key value, the following information is provided:

- Value name
- Value type
- Description of what the value represents
- The default value for the key or sample values to illustrate the entry format

In most cases, you set or modify registry key values from within AppManager components, for example, 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. To run the NTAdmin_RegistrySet Knowledge Script on a target computer, the AppManager agent service, `NetIQmc`, on the target computer must have Administrator privileges (the service must be running as `LocalSystem` or an account that has local Administrator privileges on the computer being modified).

All of the keys described in this appendix are under the **HKEY_LOCAL_MACHINE on Local Machine** registry.

Note Not all AppManager registry keys are documented in this appendix. Instead, the intent is to highlight some of the keys of particular interest or that you may be called upon to check or change.

Basic NetIQ keys

All AppManager registry keys are under **HKEY_LOCAL_MACHINE\SOFTWARE\NetIQ**. Depending on the components installed on the local computer, you will see some combination of the following **NetIQ** subkeys:

Subkey	Where you'll find it	What you find in this subkey
3.0	Computer where the Repository Browser or the Security Manager is installed.	Repository Browser configuration values and/or Security Manager configuration values.
AgtLocalRP	Any managed client	Key values for the configuration of the local repository.
AppManager	Any computer where you've installed an AppManager component	Key values that describe which AppManager components are installed and version information for each component.
Generic	Any computer where you've installed an AppManager component	Key values for generic application tracing.
NetIQccm	Any managed client	Key values that configure communication with the management server.
NetIQmc	Any managed client	Key values that configure the AppManager agent's operation.
NetIQms	Management server	Key values that configure communication with the repository.
NetIQrp	Repository server	Key values that describe the configuration of the AppManager repository database.
Web	Web management server	Key values that describe the path to the web management server.

In addition to these subkeys, **SOFTWARE\NetIQ** holds these values.

Key value	Type	Description and example
RegisteredOrganization	String	Name of organization that is registered to use NetIQ AppManager. For example, “NetIQ Corporation” or “Zero, Inc.” This value is typically recorded when you install the AppManager repository. Therefore, the default value appears exactly as it was entered during installation. For example: RegisteredOrganization:REG_SZ:Netiq Corp
RegisteredOwner	String	Name of the User that is registered to use NetIQ AppManager. For example, John Smith. This value is typically recorded when you install the AppManager repository. Therefore, the default value appears exactly as it was entered during installation. For example: RegisteredOwner:REG_SZ:John Smith
UIInstallPath	String	Path to the AppManager installation (program) directory. For example: UIInstallPath:REG_SZ:C:\NetIQ

AppManager subkey

The **SOFTWARE\NetIQ\AppManager** subkey provides version and build information for the AppManager components installed on a computer. The key values are useful for informational purposes, for example to check for compatibility issues. You should not, however, manually change any of these values.

Under **SOFTWARE\NetIQ\AppManager**, the entries you see depend on what is installed on the computer you are viewing

the registry for. For example, you may see some or all of the following:

Key value	Type	Description and example
dbbrowse.exe	String	Version and build number for the Repository Browser in the form of #.#.###.#. This value is recorded during installation. For example: dbbrowse.exe:REG_SZ: 3.4.414.2 **Note:** The format for the version and build number is the same for all components.
iconedit.exe	String	Version and build number for the Icon Manager. This value is recorded during installation.
InstallPath	String	Absolute path of NetIQ AppManager. This value is recorded during installation. For example: InstallPath:REG_SZ:D:\netiq\
netiq.dll	String	Version and build number for the AppManager dynamic link library. This value is recorded during installation.
netiq.exe	String	Version and build number for the AppManager Operator Console. This value is recorded during installation.
NetIQccm.exe	String	Version and build number for the autonomy component. This value is recorded during installation.
netiqdec.exe	String	Version and build number for the Distributed Event Console. This value is recorded during installation.
NetIQmc.exe	String	Version and build number for the managed client's AppManager agent service. This value is recorded during installation.
NetIQms.exe	String	Version and build number for the NetIQ management server component. This value is recorded during installation.
netiqole.dll	String	Version and build number for the NetIQ OLE interface that provides access to AppManager data and functionality. This value is recorded during installation.

Key value	Type	Description and example
NIQNEDIT.exe	String	Version and build number for the NetIQ Knowledge Script Editor component. This value is recorded during installation.
Q<component>a[3].dll	String	Version and build number for NetIQ managed objects. These values are recorded during installation. Depending on the managed objects you have installed and registered, you will see a different list of key values. The following example illustrate the registry key values when NT, SQL, and IIS managed objects are installed. ■ Qiisa3.dll:REQ_SZ:3.4.414.2 ■ Qnta3.dll:REQ_SZ:3.4.414.2 ■ Qsqla3.dll:REQ_SZ:3.4.414.2 The optional version number (for example, 3) indicates the major release version of the DLL. This version identifier is used to ensure both new and existing Knowledge Script jobs always run as expected, even if managed objects change.
Repository	String	Version and build number for the AppManager repository component. This value is recorded during installation.
RptMgr.exe	String	Version and build number for the AppManager Report Manager component. This value is recorded during installation.
SecMgr.exe	String	Version and build number for the AppManager Security Manager component. This value is recorded during installation.

■ ■ ■ ■ ■ ■ ■ ■

Generic\Tracing subkey

The **SOFTWARE\NetIQ\Generic\Tracing** subkey provides information about the general tracing configuration. Typically, there's no reason to manually change any of these values.

Key value	Type	Description and example
TraceLogMaxSize	Number	Maximum size for the AppManager trace log files. The default value is 2 MB. When the log file size reaches this value, the log file is backed up to a log backup file (`*.log.bkup`), The current log file then begins recording new log data.
TraceLogPath	String	Path to where the trace log files are located. The default path is: <NetIQ Install path>\Temp\NetIQ_debug For example: C:\NetIQ\Temp\NetIQ_debug

Managed client registry keys

There are several subkeys associated with managed client computers. They include:

- **SOFTWARE\NetIQ\AgtLocalRP**
- **SOFTWARE\NetIQ\NetIQccm\Config**
- **SOFTWARE\NetIQ\NetIQccm\Tracing**
- **SOFTWARE\NetIQ\NetIQmc**
- **SOFTWARE\NetIQ\NetIQmc\Config**
- **SOFTWARE\NetIQ\NetIQmc\Security**
- **SOFTWARE\NetIQ\NetIQmc\Tracing**

AgtLocalRP subkey

Under **AgtLocalRP**, the key values define the characteristics of the managed client's local repository.

Key value	Type	Description and example
RpMaxCacheSize	Number	Defines the maximum cache size for the local repository.
RpPath	String	Identifies the path to the local repository. For example: C:\NetIQ\db
RpUseODBC	Number	Flag to indicate whether communication is handled through ODBC or DAO. The default of one (0x1) is yes, indicating the computer uses ODBC. A value of zero (0) indicates the computers uses DAO.

NetIQccm subkeys

The key values under **NetIQccm\Config** are especially significant in recording the characteristics of a managed client's autonomy and communication with the management server. The **NetIQccm\Config** includes:

Key value	Type	Description and example
CacheRpcConn	Number	Flag to control whether RPC connections are cached. Zero disables caching. Any non-zero value enables RPC connection caching. The default is zero.
DataTableSize	Number	Defines the maximum size (number of records) for the local repository's Data table, if one has been defined. If you haven't configured a maximum size with AMAdmin_SetLocalRPSize, the key value is zero (0) which indicated no limit.

Key value	Type	Description and example
EventLogLvl	Number	Level of AppManager-generated event log messages to log into the NT Event Log. The valid values for this key include: **0x0** - Don't log anything into the NT Event Log **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 events and Error events. The default: 0xF (Log everything).
EventTableSize	Number	Defines the maximum size (number of records) for the local repository's Event table, if one has been defined. If you haven't configured a maximum size with AMAdmin_SetLocalRPSize, the key value is zero (0) which indicated no limit.
MonitorInterval	Number	Interval, in seconds, to check whether the NetIQmc service is running. If an interval is set and the service is down, it is restarted automatically. To turn off self-monitoring, set this value to 0. The default: 1800 seconds (30 min).
PingMSInterval	Number	Interval, in seconds, to check the status of the management server service. The default is 30 seconds, for example, 0x1e. **Note:** The NetIQccm service needs to check the management server status periodically to determine whether communication is up or down. The communication status determines whether the NetIQccm service should send data and events to the management server or log them in the local repository. This key is used when managed clients run in **Autonomous** mode.
PollMCInterval	Number	Interval, in seconds, for polling the managed client's NetIQmc service to get any new data that needs to be sent to the management server. The default is 5 seconds. This key is used when managed clients run in **Autonomous** mode.

The **NetIQccm\Tracing** describes the tracing level for the NetIQccm agent service:

Key value	Type	Description and example
TraceCCM	Number	Flag indicating NetIQccm trace level. The higher the value, the more verbose the tracing output. A value of zero (0) turns tracing off. The default value is 17 (0x11).

NetIQmc subkeys

The key values under **NetIQmc** and its subkeys are the registry keys you are most likely to want to review and in some cases modify. They control many important characteristics of a managed client's behavior.

Some of the key values directly under the **NetIQmc** subkey include:

Key value	Type	Description and example
Exchange Mailbox	String	If you install MAPI mail to allow the managed client to send mail as an action or install the Exchange managed object, installation will ask for and record this value.
Exchange Profile	String	If you install MAPI mail to allow the managed client to send mail as an action or install the Exchange managed object, installation will record this value.
Exchange Server	String	If you install MAPI mail to allow the managed client to send mail as an action or install the Exchange managed object, installation will ask for and record this value.
NetIQms	String	Name of the computer where the management server (NetIQms) is installed. **Note:** This value is only used for pre-discovery. A managed client can talk to multiple management servers.

Key value	Type	Description and example
NetIQms port	Number	The RPC port number that the management server uses to communicate with the agent service, NetIQmc. The default is 9999 (0x270f).
Port	Number	Current RPC port that the agent service, NetIQmc, uses to communicate with other AppManager components. The default is 9998 (0x270e).
ServiceDependency	String	A list of services, separated by commas, that the managed client is dependent upon. The NetIQmc service checks for the dependent services before starting up. The value is dependent on the managed objects installed. For example, if SQL and IIS objects are installed: MSSQLSever,SQLExecutive,w3svc,msftpsvc, nntpsvc,iisadmin
User Domain	String	If you install MAPI mail to allow the managed client to send mail as an action or install the Exchange managed object, the domain of the user that owns the Exchange MailBox.
User Name	String	If you install MAPI mail to allow the managed client to send mail as an action or install the Exchange managed object, the name of the user that owns the Exchange MailBox.
User Password	String	If you install MAPI mail to allow the managed client to send mail as an action or install the Exchange managed object, the password for the user that owns the Exchange MailBox.

The key values under **NetIQmc\Config** are especially significant in recording the characteristics of a managed client's autonomy and event handling. The **NetIQmc\Config** subkey includes:

Key value	Type	Description and example
Autonomy	Number	Flag to switch the Autonomy mode. Zero turns Autonomy off. One enables autonomy. The default is 1 (Autonomy on).
AutoUpdateMS	Number	Flag to control whether the managed client can automatically update its management server information. Zero prevents an automatic update. One enables the automatic update. If this flag is set to 1 and the management server has been moved to another computer or the name of the management server computer has changed, the managed client will update its internal information about the management server it communicates with to reflect the new information. The default is 1 (Auto Update MS information)
EventLogLvl	Number	Level of AppManager-generated event log messages to log into the NT Event Log. The valid values for this key are: **0x0** - Don't log anything into the NT Event Log **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 events and Error events. The default: 0xF (Log everything). The default: 0xF (Log everything).
MonitorInterval	Number	Interval, in seconds, to check the whether the NetIQccm service is running. If an interval is set and the service is down, it is restarted automatically. To turn off self-monitoring, set this value to 0. The default: 1800 seconds (30 min).

Key value	Type	Description and example
NoEventSev	Number	Defines a severity level that does not raise an event. Events with this severity will not be raised. For example, you can set a value for this key if you want to trigger an action without raising an event. See “Selectively disabling events for a managed client” on page 77 for more information about using this registry key. The default value of 0 disables this key.
NoMSEvent	Number	List of management server that the managed client will not send events to. To restrict which management servers the managed client sends events to, set this key to a comma-separated list of computers names. For example, if MARS and AJAX are the management servers that are not authorized to receive events from the AppManager agent on the local computer: NoMSEvent:REG_SZ:MARS,AJAX The default is no restriction (null).
Persistent	Number	Flag to switch Persistent mode. Zero turns Persistent mode off. One (default) turns it on
vbStringSpaceSize	Number	String size, in bytes, allocated for the Summit engine’s use. Default value is 1048576 characters (0x100000).

The key values under the **NetIQmc\Security** subkey control access to the managed client and the operations a management server is allowed to perform on the managed client. The key values include:

Key value	Type	Description and example
AllowDosCmd	String	Specifies the management servers that are allowed to run DOS commands. By default, the AllowDosCmd key is set to * to allow all management servers to initiate DOS commands. To restrict which management servers are allowed to run DOS commands, set this key to a comma-separated list of computers names. For example, if shasta and dynamo are the only management server computers 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 on this managed client. **Note:** Checking is based on the management server that initiated the job, not the user account that started the job. If the management server `Tango` started a RunDOS job on `SHASTA`, but was not included in the list of servers allowed to run DOS commands, the job on `SHASTA` would abort with an error.

Key value	Type	Description and example
AllowMS	String	Specifies the list of management servers that are allowed to communicate with the local AppManager agent (NetIQmc). By default, the AllowMS key is set to * to authorize all management servers to communicate with the local computer. To restrict which management servers are authorized to communicate with the computer, set this key to a comma-separated list of computers names. For example, if shasta and dynamo are the only management servers authorized to communicate with the AppManager agent on the local computer: AllowMS:REG_SZ:shasta,dynamo **Note:** If you specifically authorize the management servers that are allowed to communicate with a managed client, changing the name of the management server will require you to manually set this key value on each managed client that references the old management server name.
AllowReboot	String	Specifies the management servers that are allowed to request the local managed computer to reboot. By default, the AllowReboot registry key is set to null to prevent **any** management servers from rebooting managed clients. To restrict reboot operations to specific computers, enter a comma-separated list of computer names. For example: AllowReboot:REG_SZ:mktg,sales,190.12.1.28 You can use the wild card * to permit all management servers to request the local managed client to reboot.

The **NetIQmc\Tracing** describes the tracing level for the NetIQmc agent service:

Key value	Type	Description and example
TraceKS	Number	Tracing level of Knowledge Scripts. The higher the value, the more verbose the tracing output. A value of zero (0) turns tracing off. The default is 0x1. **Note:** This registry key is not currently implemented. It is intended for use in a future release.
TraceMC	Number	Tracing level for the NetIQmc component. The higher the value, the more verbose the tracing output. A value of zero (0) turns tracing off. The default is 0x1. **Note:** This registry key is not currently implemented. It is intended for use in a future release.
TraceMO<component>	Number	Tracing level for specific managed objects (for example, TraceMOcim). The higher the value, the more verbose the tracing output. A value of zero (0) turns tracing off. The default is 0x10. **Note:** This registry key is not currently implemented. It is intended for use in a future release.

Management server registry keys

There are several subkeys associated with management server computers. They include:

- **SOFTWARE\NetIQ\NetIQms**
- **SOFTWARE\NetIQ\NetIQms\Integration\SNMPTRAP**
- **SOFTWARE\NetIQ\NetIQms\Tracing**

The key values under **NetIQms** are the registry keys that you are most likely to be interested in, and in some rare cases, may need to modify. They control many important characteristics of a management server's behavior.

Some of the key values directly under the **NetIQms** subkey include:

Key value	Type	Description and example
Action Log	Number	Flag to turn on action debugging. Setting this key to a value of one logs information about management server actions for debugging purposes. Zero turns logging off. The default is 0 (off).
Comm Timeout	Number	Time, in milliseconds, the management server will wait to send a message to a managed client if there is an outstanding call. The default: 30 milliseconds.
Data Thread	Number	Number of data worker threads. Increasing this value increases the number of connections to the database, thereby increasing overhead. The default is 2.
Enable Flow Control	Number	Flag that indicates whether the management server allows the flow of network traffic to be controlled by the NetIQccm service through high and low watermarks and a transfer interval. Zero disables flow control. One enables flow control. The default is 1 (on).
Event Thread	String	Number of event worker threads. Increasing this value increases the number of connections to the database, thereby increasing overhead. The default is 1.
Job Poll Interval	Number	Sleep interval, in seconds, for the job polling thread that checks for outstanding (pending) job requests. The default is 5.
Job Status Change Thread	Number	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	Number	Sleep interval, in seconds, for the job status check thread. The job status check thread performs a “job handshake” with each managed client. The default is 500 seconds.

Key value	Type	Description and example
Job Status IP Refresh Interval	Number	The minimum interval, in seconds, before an IP name resolution refresh is performed by the job status check thread. The default 1000.
Machine Poll Interval	Number	Sleep interval, in seconds, for the machine polling thread. The default is 500 seconds.
NetIQmc Port	Number	The RPC port number on the managed client that the NetIQms service communicates with. The default is 9998 (0x9998).
Ping All Machines Interval	Number	Sleep interval for the ping all machines thread. The default is 250 seconds.
Ping Machine Poll Interval	Number	Sleep interval for the ping machine thread. The default is 5 seconds.
Port	Number	The RPC port number that the NetIQms uses to communicate with other remote components. The default is 9999 (0x9999).
QdbLog	Number	Flag to turn on repository debugging. Setting this key to one logs information about repository communication errors for debugging purposes. Zero turns logging off. The default is 0 (off).
RecordEvents	Number	Flag to control how the management server handles events. The valid values for this key are: **0** Don't record events in the AppManager repository **1** Record events in the AppManager repository (normal operation) **2** Record and automatically acknowledge events in the AppManager repository The default is 1.
RP Connection Refresh Wait	Number	Sleep interval, in seconds, between the management server's attempts to reconnect to the repository when SQL server goes down. The default is 300 seconds.
RP database	String	Name of the repository database. The default is QDB.

Key value	Type	Description and example
RP DSN	String	Repository Data Source Name for the ODBC connection. The default is QDBms.
RP Logon Timeout	Number	Maximum amount of time, in seconds, the management server waits for SQL Server if SQL Server is down when the management server starts up. The default is 600 seconds.
RP password	String	Encrypted password for logging on to the repository.
RP RPC Shutdown Threshold	Number	Number of times management server tries to connect to SQL Server before shutting down the RPC Server thread when SQL Server goes down. The default is 0.
RP server	String	Name of the machine where the AppManager repository is located.
RP username	String	User name the management server uses to connect to the AppManager repository. The default is netiq.
Rpc Log	Number	Flag to turn on RPC debugging. Setting this key to one logs information about RPC failures for debugging purposes. Zero turns logging off. The default is 0 (off).
Stop Rpc On SQL Failure	Number	Flag to stop the RPC service when SQL Server is detected down. Setting this key to one stops the service. Zero turns this feature off. The default is 1.
UUID	Binary	Unique identifier for the management server.

AppManager repository registry keys

The key values under **SOFTWARE\NetIQ\NetIQrp** reflect the configuration of the AppManager repository database that you specified during installation.

Key value	Type	Description and example
Data Device Name	String	Name for the repository data device specified during installation. The default is QDBData
Data Device Path	String	Path for the repository data device specified during installation. The default is <MSSQL DIR>\DATA. For example: C:\MSSQL\Data
Data Device Size	String	Size of the repository data device specified during installation. The default is 100 (MB).
Database Name	String	Name for the repository database specified during installation. The default is QDB.
Log Device Name	String	Name for the repository log device specified during installation. The default is QDBLog.
LogDevice Path	String	Path for the repository log device specified during installation. The default is <MSSQL DIR>\DATA. For example: C:\MSSQL\Data
LogDevice Size	String	Size of the repository log device specified during installation. The default is 50 (MB).

Web management server registry keys

The key values under **SOFTWARE\NetIQ\Web** reflect the placement of the AppManager web management server that you specified during installation.

Key value	Type	Description and example
InstallPath	String	Path for the web management server files The default is <Drive>:\Inetpub\wwroot\NetIQ. For example: C:\Inetpub\wwroot\NetIQ
vroot	String	Path for the virtual root where the AppManager Active Server Pages are located. For example: /netiq

Index

A

Action_Page 83
actions
  corrective 87
  email notification 81, 82
  expanding the use of 63
  initiated by management server 76
  initiated on management server 72
  sending a page 83, 85
  SNMP traps 86
  suppressing duplicate 71
Administrator role, default security profile 116
applications
  maintenance periods 107 to ??, 107
AppManager
  account requirements 153
  architecture 14
  communication 17
  connectors available 86
  customization 64
  granting user access 112, 113, 114
  management site 32
  required components 14
  requirements
    accounts 153
    authorized SQL logins 110
    system resources for components 35
  security layers 109
  web management server 14
AppManager Admin (AMAdmin)
  network flow control 104
  site communication type 103
AppManager agent
  introduction 14
  purpose 17
AppManager console
  introduction 14
  network connection 31
  web interface 14
AppManager management server
  actions 72
  actions initiated 76
  associated data repository 32
  data transfer 18
  determining location 34
  introduction 14
  network connection 31
  resource requirements 37
  using connectors 86
AppManager repository
  archive data
    periodic maintenance 99
    truncating 98
  backing up
    using a Knowledge Script 136
    using SQL Enterprise Manager 135
  checking integrity 148
  expanding size 143, 144

how data is stored 98
how used 19
introduction 14
moving to another SQL server 145
network connection 31
removing archived data 141
setting permissions for stored procedures 149
setting preferences 33
sizing
factors to consider 38
guidelines for estimating <$endpage 41
guidelines for estimating <$startpage 41
master database 42
temp database 41
with management scenario <$endpage 40
AppManager roles
predefined 116
AppManager users
granting access to AppManager 113, 114
understanding access rights 112
architecture, AppManager 14

B

backing up the AppManager repository 135

C

COM/OLE objects 18
community strings 121
components
checking system requirements 35
communication between 17
illustration 16
selecting during setup 15
web 14
ConfigSiteCommType 103
ConfigSiteNetFlowCtrl 104
Connectors 86
conventions 11

D

data collection
removing old 141
data streams
truncating 98
data transfers
batch size 104
DeleteOldData stored procedure 99
deployment
core Knowledge Scripts 43
developing a project plan 20
documenting decisions 48
lab environment 49
obstacles 24
on pilot group 50
project team 20
refining 63
refining the schedule 25
roadmap 66
stablizing 60
suggestions for expanding 61
time line 23
typical stages 22
widening 61
Developer's Console Utilities 15
Distributed Event Console 15
documentation
additional 11

conventions 11
importance to deployment 48

E

email notification
advantages of SMTP 82
options available 81
requirements 82
using multiple 83
events
automating notification 63
collapsing 68, 71
consecutive intervals 67, 68
custom severity value 79
disabling 77
duplicate 71
email notification 81
filtering spikes 70
forwarding to other consoles 86
general policies 67
ignoring 72
using SNMP traps 86
Exchange
monitoring
private mailbox space 163
public folder space 162
recipients 163
top senders 163
sending MAPI mail 82

G

General_EventLog 53
General_MachineDown 53

I

implementation, see deployment
installation
reasons for failure 51
scheduling 51
test environment 49
Integrated security mode 110

J

jobs
consecutive event intervals 67
information returned 19
recorded in repository 19
starting 17

K

Knowledge Scripts
disabling events 78
Exchange
MsgsThroughIMC 163
TopNISPrivateRes 163
TopNISPublicRes 162
TopNReceivers 163
TopNSenders 163
general operation 17
General_EventLog 53
General_MachineDown 53
generating reports 58
identifying core set 52
interaction with management server 19
NT_CpuLoaded 54
NT_LogicalDiskSpace 55
NT_MemUtil 55
planning customization 47
recmmendations 53
second management tier 61
setting thresholds 56
SQL_RunSql 136
storing security information

AMSAP configuration name 122
Exchange mailbox aliases 122
Exchange profiles 122
Oracle passwords 122
Oracle user names 122
SMS SQL Server logins 122
SNMP community strings 121
SQL Server logins 123
SQL Server passwords 123
use of managed objects 18

L

local repository
sizing 105

M

maintenance periods 107 to ??, 107
managed clients
defined 17
local repository
when data is written 18
network connection 31
physical location 29
primary management server 33
registry keys 77
resource requirements 36
managed computers
configuration
bandwidth limits 104
hostname or IP address 103
managed objects
DLL packaging 18
introduction 17
management server
how used 19
Windows NT service 19
management sites
deciding the number of 33
defined 32
planning network requirements 29
recommendations 34
MAPI mail 82, 85
Microsoft Exchange Server
setting up Knowledge Scripts 122
Microsoft SQL Server
AppManager repository 14, 19
setting up for Knowledge Scripts 123
setting up for SMS Knowledge Scripts 122
understanding security 109
updating passwords for Knowledge Scripts 123
Mixed security mode 110
MsgsThroughIMC 163

N

NetIQccm
communication between services 18
introduction 14
NetIQccm service
bandwidth limits 104
hostname or IP address 103
NetIQmc
communication with NetIQccm 18
introduction 14
starting jobs 17
system resources 36
NetIQms

communcationbetweenrepository and clients 19
system resources 37
network
bandwidth and latency 29
domains and trusts 29
load and demand 29
physical location of servers 29
NT_CpuLoaded 54
NT_LogicalDiskBusy 55
NT_LogicalDiskSpace 55
NT_MemUtil 55
NT_PhysicalDiskBusy 55
NT_ServiceDown 56
NT_TrustRelationship 56

O

Operator Console
system resources 38
updates 19
Oracle RDBMS Server
setting up Knowledge Scripts 122
storingusernamesandpasswords 122

P

pager notification
Action_Page 83
custom scripts 85
MAPI or Notes mail 85
options available 83
SMTP mail messages 84
using log files 85
planning
AppManager usage 28
customization 47
data collection 46
evaluating current environment 19
event handling 45
identifying computers to be monitored 27
impact of architecture 19
management tasks 43
network configuration 29
project phases 22
project team
skills required 20
repository sizing 38
security layers 42
time line 23
where to place the management server 34
worksheet sample 30
predefined AppManager roles
default security profile 116
preferences
consecutive intervals 67
event collapsing interval 71
event suppression 71
pre-installation check 35
project plan
assembling the team 20
introduction 20
time line 23
when required 19
project team
installing AppManager in the lab 50
skills required 21

R

Read-Only User, default security profile 117

Report Manager 15
reporting 62
repository
    preferences 67

S
SAP R/3
    storing login information 122
scheduling
    data transfers (uploads) 105
    installation of components 51
Security Manager
    introduction 15
security profiles
    defaults for predefined roles 116
    introduction 116
    see also AppManager roles
severity 77
SMTP mail 82, 84
SNMP community strings 121
SNMP traps 86, 87
SQL Enterprise Manager 135
Standard security mode 110
Standard User role, default security profile 117
stored procedures
    DeleteOldData 99
support 12
system requirements
    accounts 153

T
technical support 12
TopNISPrivateRes 163
TopNISPublicRes 162
TopNReceivers 163
TopNSenders 163

U
users
    See AppManager users
    see AppManager users
    top mail senders 163
    top receivers 163

V
versioning
    used in Knowledge Scripts 171

W
web management server
    introduction 14
Windows NT Authentication
    security mode 43, 110
Windows NT services
    interaction 18
    NetIQccm 14
    NetIQmc 14, 17
    NetIQms 14, 19