Management Guide

NetIQ® AppManager® for Microsoft Cluster Server

April 2019

Legal Notice

Contents

About this Book and the Library 5

About NetIQ Corporation 7

1 Introducing AppManager for Microsoft Cluster Server 9

1.1 Understanding MSCS 9

1.2 Why Monitor MSCS? 10

1.3 How AppManager Can Help. 11

2 Installing AppManager for Microsoft Cluster Server 13

2.1 System Requirements 13

2.2 Installing the Module 14

2.3 Deploying the Module with Control Center. 15

2.4 Silently Installing the Module 16

2.5 Permissions for Running MSCS Knowledge Scripts 17

2.6 Post-installation Considerations. 17

2.7 Discovering Microsoft Cluster Server Resources. 17

2.8 Upgrading Knowledge Script Jobs 18

3 MSCS Knowledge Scripts 21

3.1 EventLog 22

3.2 GroupDown 24

3.3 GroupOwnerChange 25

3.4 HealthCheck. 26

3.5 NetInterfaceDown 28

3.6 NetworkDown. 29

3.7 NodeDown 30

3.8 ResourceDown. 30

3.9 ResourceOwnerChange. 31

A Monitoring Clustered Resources and Applications 33

A.1 Monitoring Clusters with AppManager 33

A.2 Planning to Monitor Cluster Resources 33

A.3 Installing on an Active Cluster Node 34

A.4 Monitoring a Log File on a Shared Cluster Disk. 34

A.5 Monitoring Applications in an Active/Passive Cluster. 38

A.6 Monitoring Applications in an Active/Active Cluster 39

A.7 Using Troubleshooter to Check Cluster Configuration 42

A.8 Resetting Resource Dependencies 42

About this Book and the Library

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

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

Intended Audience

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

Other Information in the Library

The library provides the following information resources:

Installation Guide for AppManager

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

User Guide for AppManager Control Center

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

Administrator Guide for AppManager

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

Upgrade and Migration Guide for AppManager

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

Management guides

Provide information about installing and monitoring specific applications with AppManager.

Help

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

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

About NetIQ Corporation

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

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

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

Enabling critical business services, better and faster

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

Our Philosophy

Selling intelligent solutions, not just software

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

Driving your success is our passion

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

Our Solutions

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

Contacting Sales Support

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

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

Contacting Technical Support

For specific product issues, contact our Technical Support team.

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

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. The documentation for this product is available on the NetIQ Web site in HTML and PDF formats on a page that does not require you to log in. If you have suggestions for documentation improvements, click **Add Comment** at the bottom of any page in the HTML version of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

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

1 Introducing AppManager for Microsoft Cluster Server

The Microsoft Cluster Server (MSCS) clustering software allows multiple Microsoft Windows servers to be linked together and perform as one virtual server. The Cluster Administrator defines the servers as nodes in a cluster. If a server on a cluster node fails or is taken offline, the other server in the cluster takes over the failed server's tasks. Resource allocation moves seamlessly from the failed server to the available server, with users experiencing little or no interruption.

This chapter provides a brief introduction to MSCS and an overview of key concepts and terminology. It also summarizes the ways AppManager can help you monitor MSCS.

1.1 Understanding MSCS

A Windows cluster is a collection of one or more Windows servers that work together. MSCS provides uninterrupted service by detecting the failure of applications and servers and automatically migrating resources and workload to other servers in MSCS. Each Windows server that is part of the cluster is called a cluster node. Nodes share one or more SCSI disks, which are called shared cluster disks.

A server cluster is a group of independent servers running the cluster service. MSCS supports a minimum of two nodes and a maximum of eight nodes in a cluster setup. If a node in the cluster is unavailable because of failure or a planned outage, resources and applications move from the failed node to another node in the cluster. The cluster service ensures uninterrupted resource availability to users.

A cluster resource is a basic system entity, such as a physical disk, process, service, network address, or network name. A cluster group is a collection of cluster resources related either logically or physically by dependencies.

When a node provides resources and executes processes belonging to a cluster group, the node is said to "own" the cluster group and all of its resources. A node in a cluster may own different shared cluster disks at different times. However, each shared physical disk and all its logical disks are owned by only one cluster node at a time.

A failover occurs when one or more cluster resources in a cluster group fail and MSCS migrates the troubled group from one node to another. Ownership of the group is then transferred to the new node.

When installing MSCS, you specify the network name of the cluster. This name becomes the name of the virtual servers you install on the cluster. A virtual server is an application server within a cluster group. Executable and other dependent files of a virtual server are installed on a shared cluster disk so that each node in MSCS can execute the programs. Clients connect to the virtual server through a cluster resource network name. The network name provides transparent access to the virtual server regardless of which cluster node currently owns the group.

If a failover takes place, ownership of the group, including the network name, is transferred to a new cluster node. The node restarts all member processes of the virtual servers in the group. Because client computers use the network name to connect to the virtual server, it does not matter that a different node now owns the group.

MSCS uses an intelligent middleware that resides between the operating system and the applications available to users.

MSCS consists of the following components:

- **Clustering software**: Clustering software enables the servers in a cluster to communicate with each other and schedule the allocation of resources between servers. Clustering software further consists of the following components:

- **Cluster Service**: The Cluster Service runs on each cluster server and manages communication between cluster servers.
- **Resource Monitor**: The Resource Monitor manages communication between the Cluster Service and server resources.

- **Cluster Administrator**: Cluster Administrator allows users to manage and configure cluster servers. The Cluster Administrator provides several setup wizards for installing applications and services on the cluster.

The following table lists some of the commonly used terms in a clustering environment:

Term	Definition
Node	A server that is a member of a cluster.
Resource	A hardware or software component such as a printer, an IP address, or a disk that is available in a cluster.
Failover or Failback	The process of moving resources from a failed server to an available server in a cluster.
Group	A combination of resources that are managed as a unit of failover. The combination is also called Resource Group or Failover Group.
Active / Active	A state in which multiple nodes in a cluster contain resources or applications that can actively service clients.
Active / Passive	A state in which one node in a cluster actively services clients while the other node is idle.
Heartbeat	A repetitive communication sent back and forth between the cluster nodes at specific intervals. If the signal is not returned, the failover process begins.

1.2 Why Monitor MSCS?

Any enterprise needs to provide uninterrupted server availability in an organization and probe the exact causes for server downtime.

MSCS ensures that resources and applications existing on a failed server are restarted or available on other nodes in the cluster. The downtime may be momentary but business operations can continue uninterrupted.

Monitoring MSCS helps provide *high availability*, *fail-back*, *manageability,* and *scalability* for most of the mission-critical applications and resources.

- *High availability* ensures that resources such as IP addresses and disk drives are automatically transferred from a failed node to another online node.
- *Fail-back* ensures that resource availability is automatically re-balanced when a failed server comes back online.

- *Manageability* ensures that you can manually balance the server workloads by moving applications and data between servers on a cluster.
- *Scalability* ensures that additional nodes can be added to the cluster to meet future enterprise needs.

Clustering does not solve downtime problems external to the system, such as security breaches or incorrect system configuration. Clustering must ideally be implemented if server downtime is caused by hardware failures, such as loss of a server or corrupted memory.

In a non-clustered environment, mission-critical system failure can result in loss of revenue, data, and customers.

1.3 How AppManager Can Help

AppManager for Microsoft Cluster Server provides several Knowledge Scripts designed to give you a comprehensive view of how MSCS performs. The capabilities of the Knowledge Scripts in the MSCS category include:

- Monitoring Microsoft Windows Event Log entries created by Microsoft Cluster Server.
- Detecting when a cluster resource group is not online and attempting to bring it online automatically.
- Detecting when the cluster group owner changes.
- Verifying the status of a cluster node, network, or a network interface.
- Verifying whether the cluster resource is online and attempting to bring it online automatically.
- Detecting when the cluster resource owner changes.

You can set thresholds that specify the boundaries of optimal performance, and instruct AppManager to raise events when those thresholds are breached.

2 Installing AppManager for Microsoft Cluster Server

This chapter provides installation instructions and describes system requirements for AppManager for Microsoft Cluster Server (MSCS).

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

2.1 System Requirements

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

AppManager for MSCS has the following system requirements:

Software/Hardware	Version
NetIQ AppManager installed on the AppManager repository (QDB) computers, on the MSCS computers you want to monitor (agents), and on all console computers	8.0.3, 8.2, 9.1, 9.2, 9.5, or later One of the following AppManager agents are required ◆ AppManager agent 7.0.4 with hotfix 72616 or later ◆ AppManager agent 8.0.3, 8.2, 9.1, 9.2, 9.5, or later
Microsoft Windows operating system on the agent computers	One of the following: ◆ Windows Server 2019 ◆ Windows Server 2016 ◆ Windows Server 2012 R2 ◆ Windows Server 2012 ◆ Windows Server 2008 R2 ◆ Windows Server 2008 (32-bit and 64-bit) ◆ Windows Server 2003 R2 (32-bit and 64-bit)
Microsoft Cluster Server on the agent computers	Supports 32-bit and 64-bit systems
AppManager for Microsoft Windows module installed on the AppManager repository (QDB) computer, on the computers you want to monitor (agents), and on all console computers	7.6.170.0 or later

Software/Hardware	Version
Microsoft SQL Server Native Client 11.0	11.3.6538.0 or later
(for TLS 1.2 support)	**NOTE:** The SQL Server Native client can be installed from this Microsoft download link.

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

2.2 Installing the Module

Run the module installer on the cluster computers you want to monitor (agents) to install the agent components, and run the module installer on all console computers to install the Help and console extensions.

Access the `AM70-MSCS-7.x.x.0.msi` module installer from the `AM70_MSCS_7.x.x.0` self-extracting installation package on the AppManager Module Upgrades & Trials page.

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

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

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

You can install the module manually, or you can use Control Center to deploy the module on a remote computer where an agent is installed. For more information, see Section 2.3, "Deploying the Module with Control Center," on page 15. However, if you do use Control Center to deploy the module, Control Center only installs the *agent* components of the module. The module installer installs the QDB and console components as well as the agent components on the agent computer.

To manually install the module:

1. Double-click the module installer `.msi` file.
2. Accept the license agreement.
3. Review the results of the pre-installation check. You can expect one of the following three scenarios:
    - *No AppManager agent is present*. In this scenario, the pre-installation check fails, and the installer does not install agent components.
    - *An AppManager agent is present, but some other prerequisite fails*. In this scenario, the default is to not install agent components because of one or more missing prerequisites. However, you can override the default by selecting **Install agent component locally**. A

missing application server for this particular module often causes this scenario. For example, installing the AppManager for Microsoft SharePoint module requires the presence of a Microsoft SharePoint server on the selected computer.

- *All prerequisites are met*. In this scenario, the installer will install the agent components.

4 To install the Knowledge Scripts into the QDB:

4a Select **Install Knowledge Scripts** to install the repository components, including the Knowledge Scripts, object types, and SQL stored procedures.

4b Specify the SQL Server name of the server hosting the QDB, as well as the case-sensitive QDB name.

5 (Conditional) If you use Control Center 7.x, run the module installer for each QDB attached to Control Center.

6 (Conditional) If you use Control Center 8.x or later, run the module installer only for the primary QDB. Control Center automatically replicates this module to secondary QDBs.

7 Run the module installer on all console computers to install the Help and console extensions.

8 Run the module installer on the MSCS computers you want to monitor (agents) to install the agent components.

9 Ensure the MSCS service is running. For more information, see Section 2.6, “Post-installation Considerations,” on page 17.

10 ***If you have not discovered MSCS resources***, run the Discovery_MSCS Knowledge Script on all agent computers where you installed the module. For more information, see Section 2.7, “Discovering Microsoft Cluster Server Resources,” on page 17.

11 To get the updates provided in this release, upgrade any running Knowledge Script jobs. For more information, see Section 2.8, “Upgrading Knowledge Script Jobs,” on page 18.

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

2.3 Deploying the Module with Control Center

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

2.3.1 Deployment Overview

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

To deploy the module on an agent computer:

1. Verify the default deployment credentials.
2. Check in an installation package. For more information, see Section 2.3.2, “Checking In the Installation Package,” on page 16.
3. Configure an e-mail address to receive notification of a deployment.
4. Create a deployment rule or modify an out-of-the-box deployment rule.
5. Approve the deployment task.
6. View the results.

2.3.2 Checking In the Installation Package

You must check in the installation package, `AM70-MSCS-7.x.x.0.xml`, before you can deploy the module on an agent computer.

To check in a module installation package:

1. Log on to Control Center using an account that is a member of a user group with deployment permissions.
2. Navigate to the **Deployment** tab (for AppManager 8.x or later) or **Administration** tab (for AppManager 7.x).
3. In the Deployment folder, select **Packages**.
4. On the Tasks pane, click **Check in Deployment Packages** (for AppManager 8.x or later) or **Check in Packages** (for AppManager 7.x).
5. Navigate to the folder where you saved `AM70-MSCS-7.x.x.0.xml` and select the file.
6. Click **Open**. The Deployment Package Check in Status dialog box displays the status of the package check in.

2.4 Silently Installing the Module

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

```
msiexec.exe /i "AM70-MSCS-7.x.x.0.msi" /qn
```

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

To get the updates provided in this release, upgrade any running Knowledge Script jobs. For more information, see Section 2.8, “Upgrading Knowledge Script Jobs,” on page 18.

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

```
/L* "AM70-MSCS-7.x.x.0.msi.log"
```

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

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

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

Windows authentication:

```
AM70-MSCS-7.x.x.0.msi /qn MO_B_QDBINSTALL=1 MO_B_MOINSTALL=0 MO_B_SQLSVR_WINAUTH=1
MO_SQLSVR_NAME=SQLServerName MO_QDBNAME=AM-RepositoryName
```

SQL authentication:

```
AM70-MSCS-7.x.x.0.msi /qn MO_B_QDBINSTALL=1 MO_B_MOINSTALL=0 MO_B_SQLSVR_WINAUTH=0
MO_SQLSVR_USER=SQLLogin MO_SQLSVR_PWD=SQLLoginPassword
MO_SQLSVR_NAME=SQLServerName MO_QDBNAME=AM-RepositoryName
```

2.5 Permissions for Running MSCS Knowledge Scripts

AppManager for Microsoft Cluster Server requires that the NetIQ AppManager Client Resource Monitor (netiqmc) and the NetIQ AppManager Client Communication Manager (netiqccm) agent services have the following permissions:

- Ability to log on as a service
- Membership in the Domain Admin Group

By default, the setup program configures the agent to use the Windows Local System account.

To update the agent services:

1. Start the Services Administrative Tool. You can open the Administrative Tools folder in the Control Panel.
2. Right-click the **NetIQ AppManager Client Communication Manager** (netiqccm) service in the list of services, and select **Properties**.
3. On the Logon tab, specify the appropriate account to use.
4. Click **OK**.
5. Repeat steps 2 through 4 for the **NetIQ AppManager Client Resource Monitor** (netiqmc) service.
6. Restart both services.

2.6 Post-installation Considerations

To successfully discover Microsoft Cluster Server resources, ensure that the Cluster Server service is running.

To verify whether the service is running:

1. On all agent computers where you have installed the module, navigate to Control Panel, double-click **Administrative Tools**, and then double-click **Services**.
2. Verify that the **Status** column displays **Started** for the Cluster Server service.
3. If the service is not running, right-click the service and select **Start**.

2.7 Discovering Microsoft Cluster Server Resources

Use the Discovery_MSCS Knowledge Script to discover Microsoft Cluster Server (MSCS) configuration and resources. You can run this script on any node in the cluster. Only discover actual cluster nodes. You should not attempt to add or discover virtual servers

Every time you add a new node to the cluster, you must install the AppManager agent on the new node and run the Discovery_MSCS Knowledge Script to discover it. After you discover the new cluster node, the new node appears in the TreeView pane of the Operator Console or in the Control Center Console.

Run Discovery_MSCS on MSCS server objects. By default, this script runs once for each computer.

2.7.1 Discovery Parameters

Set the Values tab parameters as needed:

Description	How to Set It
Raise event when discovery succeeds? (y/n)	This script always raises an event when the job fails for any reason. In addition, you can set this parameter to **y** to raise an event when the job succeeds. The default is n.
Event severity when discovery succeeds	Set the event severity level, from 1 to 40, to reflect the importance of an event in which discovery succeeds. The default is 25 (blue event indicator).
Event severity when discovery fails	Set the event severity level, from 1 to 40, to reflect the importance of an event in which discovery fails. The default is 5 (red event indicator).
Event severity when discovery partially succeeds	Set the event severity level, from 1 to 40, to reflect the importance of an event in which discovery returns some data but also generates warning messages. The default is 10 (red event indicator).
Event severity when discovery is not applicable	Set the event severity level, from 1 to 40, to reflect the importance of an event in which the target computer does not have MSCS installed. The default is 15 (yellow event indicator).

2.7.2 Example of Using this Script

If you discover multiple nodes that are part of the same cluster, you may see what appear to be duplicate entries in the application view. For example, assume you have a SQL Server cluster with the computers `LOB01` and `LOB02`. Both of these computers are displayed in the Master view. After you run the SQL discovery to discover the cluster, `LOB01` and `LOB02` display the resource object `SQL Server:LOSLOBOS_SQL` in the Master view. `LOSLOBOS_SQL` represents the virtual server.

When you discover clustered applications such as Exchange and SQL Server, the discovered objects use the virtual server name, for example, `LOSLOBOS_SQL`. When you run Discovery_MSCS, the discovered objects use the physical node name. For example, after running Discovery_MSCS on the physical nodes and virtual server names described in this example, the Master view should display `MSCS Server:LOB01` under `LOB01`, and `MSCS Server:LOB02` under `LOB02`.

2.8 Upgrading Knowledge Script Jobs

If you are using AppManager 8.x or later, the module upgrade process now *retains* any changes you might have made to the parameter settings for the Knowledge Scripts in the previous version of this module. Before AppManager 8.x, the module upgrade process *overwrote* any settings you might have made, changing the settings back to the module defaults.

As a result, if this module includes any changes to the default values for any Knowledge Script parameter, the module upgrade process ignores those changes and retains all parameter values that you updated. Unless you review the management guide or the online Help for that Knowledge Script, you will not know about any changes to default parameter values that came with this release.

You can push the changes for updated scripts to running Knowledge Script jobs in one of the following ways:

- Use the AMAdmin_UpgradeJobs Knowledge Script.
- Use the Properties Propagation feature.

2.8.1 Running AMAdmin_UpgradeJobs

The AMAdmin_UpgradeJobs Knowledge Script can push changes to running Knowledge Script jobs. Your AppManager repository (QDB) must be at version 7.0 or later. In addition, the repository computer must have hotfix 72040 installed, or the most recent AppManager Repository hotfix. To download the hotfix, see the AppManager Suite Hotfixes page.

Upgrading jobs to use the most recent script version allows the jobs to take advantage of the latest script logic while maintaining existing parameter values for the job.

For more information, see the Help for the AMAdmin_UpgradeJobs Knowledge Script.

2.8.2 Propagating Knowledge Script Changes

You can propagate script changes to jobs that are running and to Knowledge Script Groups, including recommended Knowledge Script Groups and renamed Knowledge Scripts.

Before propagating script changes, verify that the script parameters are set to your specifications. You might need to appropriately set new parameters for your environment or application.

If you are not using AppManager 8.x or later, customized script parameters might have reverted to default parameters during the installation of the module.

You can choose to propagate only properties (specified in the **Schedule** and **Values** tabs), only the script (which is the logic of the Knowledge Script), or both. Unless you know specifically that changes affect only the script logic, you should propagate the properties and the script.

For more information about propagating Knowledge Script changes, see the “Running Monitoring Jobs” chapter of the *Control Center User Guide for AppManager*.

2.8.3 Propagating Changes to Ad Hoc Jobs or Knowledge Script Groups

You can propagate the properties and the logic (script) of a Knowledge Script to ad hoc jobs started by that Knowledge Script. Corresponding jobs are stopped and restarted with the Knowledge Script changes.

You can also propagate the properties and logic of a Knowledge Script to corresponding Knowledge Script Group members. After you propagate script changes to Knowledge Script Group members, you can propagate the updated Knowledge Script Group members to associated running jobs. Any monitoring jobs started by a Knowledge Script Group member are restarted with the job properties of the Knowledge Script Group member.

To propagate changes to ad hoc Knowledge Script jobs or Knowledge Script Groups:

1. In the Knowledge Script view, select the Knowledge Script or Knowledge Script Group for which you want to propagate changes.
2. Right-click the script or group and select **Properties propagation** > **Ad Hoc Jobs**.

3 Select the components of the Knowledge Script that you want to propagate to associated ad hoc jobs or groups and click **OK**:

Select	To propagate
Script	The logic of the Knowledge Script.
Properties	Values from the Knowledge Script Schedule and Values tabs, such as schedule, monitoring values, actions, and advanced options. If you are using AppManager 8.x or later, the module upgrade process now *retains* any changes you might have made to the parameter settings for the Knowledge Scripts in the previous version of this module.

3 MSCS Knowledge Scripts

AppManager provides Knowledge Scripts for monitoring Microsoft Cluster Server. These scripts can monitor shared drives and other shared resources from active nodes.

The Knowledge Scripts in the MSCS category are designed to run on clustered servers. They use the Cluster Administrator API, which enables them to track shared resources better.

Run the MSCS Knowledge Scripts on a server group representing all the nodes in the Microsoft Cluster Server, or on at least two different nodes. By running them on multiple nodes, you covered in case of a failure on one node, such as a computer that disconnects from the network or has a system failure. If you ran all your MSCS Knowledge Scripts on that node, you would receive no events from them.

The ability to monitor resources from active nodes applies to all the Knowledge Scripts in the MSCS category.

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

Knowledge Script	What It Does
EventLog	Monitors Windows Event Log entries created by the Microsoft Cluster Server (entries that have ClusSvc as their Source in the System Log).
GroupDown	Detects when a cluster resource group is not online. Can attempt to bring that cluster group online automatically.
GroupOwnerChange	Detects whether the owner of a cluster group has changed.
HealthCheck	Checks whether a node, network, resource, group, or a network interface is down. Also checks whether the ownership of a group has changed.
NetInterfaceDown	Checks whether a cluster network interface is down.
NetworkDown	Checks whether a cluster network is down.
NodeDown	Checks whether a cluster node is down.
ResourceDown	Detects when a cluster resource is not online; attempts to bring that cluster resource online automatically.
ResourceOwnerChange	Detects whether the owner of a cluster resource has changed.

3.1 EventLog

Use this Knowledge Script to monitor and filter Microsoft Windows Event Log entries created by the Microsoft Cluster Server (entries that have **ClusSvc** as their Source in the System Log). This script tracks Windows event log entries that match a set of filtering criteria and notifies you when a log entry that meets the filtering criteria is generated during the monitoring interval.

This script works on an incremental basis, meaning it does not fully rescan the event log each time it runs, and all log entries that match the filtering criteria are returned in the event or data point detail message.

3.1.1 Resource Object

Microsoft Cluster Server

In Windows Server 2003 environments, run this script on only one node to avoid duplication of events. Running this script on multiple nodes results in multiple scannings of the same Event Log, which in turn results in duplicate events for the same Event Log entries.

3.1.2 Default Schedule

By default, this script runs every 30 minutes.

3.1.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event if log entries match your search criteria. The default is y. **NOTE:** The format for Event log entries in Windows Server 2008 differs slightly from the format of Event log entries in Windows Server 2003. This difference can affect the information displayed in an event message for events raised on failover clusters. Specifically, for Windows Server 2003, the Description in the event message is an alphabetic value derived from the ResourceName and ResourceGroup fields in the Event log. In Windows Server 2008, the Description may be a numeric value or may be blank, depending on the contents of the ResourceName and ResourceGroup fields.
Collect data?	Set to **y** to collect data for charts and reports. The default is n. If enabled, data collection returns the number of new event log entries, and the detailed message lists the log entries.

Description	How to Set It
Events in past N hours	Set this parameter to control checking for the first interval (after which checking is incremental): ◆ **-1** for all the existing entries ◆ **n** for the past N hours (8 for the past 8 hours, 50 for the past 50 hours, etc.) ◆ **0** for no previous entries (only search from this moment onward)
Monitor for error events?	Set to **y** to monitor the Event Log for error events. The default is y.
Monitor for warning events?	Set to **y** to monitor the Event Log for warning events. The default is y.
Monitor for information events?	Set to **y** to monitor the Event Log for information events. The default is y.
Monitor for success audit events?	Set to **y** to monitor the Event Log for success audit events. The default is y.
Monitor for failure audit events?	Set to **y** to monitor the Event Log for failure audit events. The default is y.
Filter the Event Category field for	To monitor for events in a particular category (for example Server or Logon), enter an appropriate search string. This script looks for matching entries in the Event Log Category field. Multiple strings can be entered separated by commas. The search string can contain criteria used to include entries, exclude entries, or both. Separate the include and exclude criteria with a colon (:). If you are specifying only include criteria, the colon is not necessary.
Filter the Event ID field for	To monitor for particular event IDs, enter an appropriate search string. This script looks for matching entries in the Event Log Event field. Multiple IDs and ranges can be entered separated by commas. For example: 1,2,10-15,202. The search string can contain criteria used to include entries, exclude entries, or both. Separate the include and exclude criteria with a colon (:). If you are specifying only include criteria, the colon is not necessary.
Filter the Event User field for	To monitor for events associated with a particular user, enter an appropriate search string. This script looks for matching entries in the Event Log's User field. Multiple strings can be entered separated by commas. The search string can contain criteria used to include entries, exclude entries, or both. Separate the include and exclude criteria with a colon (:). If you are specifying only include criteria, the colon is not necessary.
Filter the Event Computer field for	To monitor for events generated by a particular computer, enter an appropriate search string. This script looks for matching entries in the Event Log Computer field. Multiple strings can be entered separated by commas. The search string can contain criteria used to include entries, exclude entries, or both. Separate the include and exclude criteria with a colon (:). If you are specifying only include criteria, the colon is not necessary.

Description	How to Set It
Filter the Event Description field for	To monitor for events with a particular detail description or containing keywords in the description, enter an appropriate search string. This script looks for matching entries in the Event Log Description field. Multiple strings can be entered separated by commas. The search string can contain criteria used to include entries, exclude entries, or both. Separate the include and exclude criteria with a colon (:). If you are specifying only include criteria, the colon is not necessary.
Maximum number of entries per event report	Specify the maximum number of entries that can be recorded into each event's detail message before an event is raised. If this script finds more entries from the log than can be put into one event report, it raises multiple events to report all the outstanding entries in the log. The default is 30 entries.
Event severity level when log entries match search criteria	Set the event severity level, from 1 to 40, to indicate the importance of an event in which log entries match your search criteria. The default is 8 (red event indicator). You can adjust the severity depending on the log or type of event you are checking.
Event severity level when job fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the EventLog job fails unexpectedly. The default is 35 (magenta event indicator).

3.2 GroupDown

Use this Knowledge Script to detect whether a cluster resource group is online. This script raises an event if the cluster group is offline. You can set this script to automatically bring the cluster group online.

3.2.1 Resource Object

Microsoft cluster group

3.2.2 Default Schedule

The default interval for this script is **Every 10 minutes**.

3.2.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event if a cluster group is not online or if a cluster-related API failure occurs. The default is y.

Description	How to Set It
Collect data?	Set to **y** to collect data for charts and reports. The default is n. If enabled, data collection returns a value of: ◆ **100** if the cluster resource group is online. ◆ **50** if the group is partially online. ◆ **0** if the cluster group is off-line. ◆ **-1** if the cluster group cannot be found.
Auto-start cluster resource group?	Set to **y** to automatically start the cluster resource group. The default is y.
List of cluster groups to be excluded	To exclude cluster groups from monitoring: ◆ Specify the comma separated list of cluster groups to be excluded. ◆ If a cluster group name is found across multiple severs, then all the matching cluster groups will be excluded. ◆ If you want to exclude a cluster group only from a specific server, then specify the cluster group name along with the server name in the following format: `server name|group name`. Example: `SERVER1|Group1, SERVER2|Group2`
Event severity level when cluster group offline and auto-start fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and auto-start fails. The default is 5 (red event indicator).
Event severity level when cluster group offline and auto-start succeeds	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and auto-start succeeds. The default is 25 (blue event indicator).
Event severity level when cluster group offline and auto-start is set to n	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and you set the Auto-start cluster resource group? parameter to n. The default is 18 (yellow event indicator).
Event severity level when cluster-related API failure occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster-related API failure occurs. The default is 15 (yellow event indicator).

3.3 GroupOwnerChange

Use this Knowledge Script to detect whether the owner of a cluster group has changed. Changes in the ownership of a cluster group typically indicate a failover or failback operation has taken place. This script raises an event if the owner of the cluster group changes or if a cluster-related API failure occurs.

Both the event and data detail message indicate the previous cluster group owner and the new group owner.

3.3.1 Resource Object

Microsoft cluster group

3.3.2 Default Schedule

The default interval for this script is **Every 5 minutes**.

3.3.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event when cluster group ownership changes or if a cluster-related API failure occurs. The default is y.
Collect data?	Set to **y** to collect data for charts and reports. The default is n. If enabled, data collection returns a value of 0 if the cluster group owner has changed in the interval, or a value of 100 if the cluster group owner has stayed the same. The detail message indicates the previous group owner and the new group owner.
Event severity level when cluster group owner changes	Set the event severity level, from 1 to 40, to indicate the importance of an event in which cluster group ownership has changed. The default is 5 (red event indicator).
Event severity level when cluster-related API failure occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster-related API failure occurs. The default is 15 (yellow event indicator).

3.4 HealthCheck

Use this Knowledge Script to determine whether a Microsoft Windows cluster node, network, resource, group, or network interface is down. This script can also determine whether the ownership of a cluster group has changed.

3.4.1 Resource Objects

Microsoft cluster node, network, resource, group, and network interface object

3.4.2 Default Schedule

The default interval for this script is **Every 10 minutes.**

3.4.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Data Collection	
Collect data?	Select **Yes** to collect data for graphs and charts. The default is Yes.
Monitoring	

Description	How to Set It
Auto-start cluster resource?	Select **Yes** to automatically start a cluster resource that is down. The default is Yes.
Auto-start cluster group?	Select **Yes** to automatically start a cluster group that is down. The default is Yes.
List of resource to be excluded	To exclude resources from monitoring: ◆ Specify the comma separated list of resources to be excluded. ◆ If a resource name is found across multiple severs, then all the matching resources will be excluded. ◆ If you want to exclude a resource only from a specific server, then specify the resource name along with the server name in the following format: `server name|resource name`. Example: `SERVER1|Resource1, SERVER2|Resource3`
Event Notification	
Raise event if changes occur in a network, resource, or group?	Select **Yes** to raise separate events, one for each component, for changes that occur in Cluster Server components: network, resource, or group. The default is Yes.
Raise a single event?	Select **Yes** to raise one event that summarizes all changes that have occurred in all Cluster Server components: network, resource and group. The default is Yes.
Event severity level when cluster-related API failure occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster-related API failure occurs. The default is 15.
Event severity level when a node is down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a node is down. The default is 8.
Network	
Event severity level when network is down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the network is down. The default is 8.
Event severity level when network is partitioned	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the network is partitioned. The default is 9.
Event severity level when cluster network interface is down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the cluster network interface is down. The default is 8.

Description	How to Set It
Cluster Resources	
Event severity level when cluster resource is off-line and auto-start fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster resource is offline and auto-start fails. The default is 5.
Event severity level when cluster resource is off-line and auto-start is set to 'No'	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster resource is offline and you deselected the Auto-start cluster resource? parameter. The default is 18.
Event severity level when cluster resource is off-line and auto-start is successful	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster resource is offline and auto-start succeeds. The default is 25.
Cluster Group	
Event severity level when cluster group owner changes	Set the event severity level, from 1 to 40, to indicate the importance of an event in which cluster group ownership has changed. The default is 5.
Event severity level when cluster group is off-line and auto-start fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and auto-start fails. The default is 5 (red event indicator).
Event severity level when cluster group is off-line and auto-start is set to 'No'	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and you deselected the Auto-start cluster group? parameter. The default is 18 (yellow event indicator).
Event severity level when cluster group is off-line and auto-start is successful	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and auto-start succeeds. The default is 2 (blue event indicator).

3.5 NetInterfaceDown

Use this Knowledge Script to detect whether a cluster network interface is down. This script raises an event if the network interface is down or a cluster-related API failure occurs.

3.5.1 Resource Object

Microsoft cluster net interface object

3.5.2 Default Schedule

The default interval for this script is **Every 10 minutes**.

3.5.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event if a cluster network interface is down or when a cluster-related API failure occurs. The default is y.

Description	How to Set It
Collect data?	Set to **y** to collect data for charts and reports.The default is n. If enabled, data collection returns a value of 100 if the interface is up and a value of 0 if the interface is down.
Event severity level when cluster network interface down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster network interface is down. The default is 8 (red event indicator).
Event severity level when cluster-related API failure occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster-related API failure occurs. The default is 15 (yellow event indicator).

3.6 NetworkDown

Use this Knowledge Script to detect whether a cluster network is down. This script raises an event if the network is down or if a cluster-related API failure occurs.

3.6.1 Resource Object

Microsoft cluster network folder

3.6.2 Default Schedule

The default interval for this script is **Every 10 minutes**.

3.6.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event if the network is down, if the network has been partitioned, or when a cluster-related API failure occurs. The default is y.
Collect data?	Set to **y** to collect data for charts and reports.The default is n. If enabled, data collection returns a value of 100 if the cluster network is up and a value of 0 if the cluster network is down.
Event severity level when network down	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the network is down. The default is 8 (red event indicator).
Event severity level when network has been partitioned	Set the event severity level, from 1 to 40, to indicate the importance of an event in which the network has been partitioned. The default is 9 (red event indicator).
Event severity level when cluster-related API failure occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster-related API failure occurs. The default is 15 (yellow event indicator).

3.7 NodeDown

Use this Knowledge Script to detect whether a cluster node is down. This script raises an event if the node is down or when a cluster-related API failure occurs.

3.7.1 Resource Object

Microsoft cluster node

3.7.2 Default Schedule

The default interval for this script is **Every 10 minutes**.

3.7.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event if a node is down or if a cluster-related API failure occurs. The default is y.
Collect data?	Set to **y** to collect data for charts and reports.The default is n. If enabled, data collection returns a value of 100 if the cluster node is up and a value of 0 if the cluster node is down.
Event severity level when node down	Set the event severity level, from 1 to 40, to indicate the importance o an event in which a node is down. The default is 8 (red event indicator).
Event severity level when cluster-related API failure occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster-related API failure occurs. The default is 15 (yellow event indicator).

3.8 ResourceDown

Use this Knowledge Script to detect if a cluster resource is online. This script raises an event if the resource if offline or if a cluster-related API failure occurs. You can set this script to attempt to bring the cluster resource online automatically.

3.8.1 Resource Object

Microsoft cluster resource

3.8.2 Default Schedule

The default interval for this script is **Every 10 minutes**.

3.8.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event if a cluster group is offline or if a cluster-related API failure occurs. The default is y.
Collect data?	Set to **y** to collect data for charts and reports.The default is n. If enabled, data collection returns a value of 100 if the cluster resource is online and a value of 0 if the cluster resource is off-line.
Auto-start cluster resource?	Set to **y** to automatically start the cluster resource. The default is y.
Event severity level when cluster group off-line and auto-start fails	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and auto-start fails. The default is 5 (red event indicator).
Event severity level when cluster group off-line and auto-start succeeds	Set the event severity level, from 1 to 40, to indicate the importance o an event in which a cluster group is offline and auto-start succeeds. The default is 25 (blue event indicator).
Event severity level when cluster group off-line and auto-start set to no	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster group is offline and you have set the Auto-start cluster resource? parameter to n. The default is 18 (yellow event indicator).
Event severity level when cluster-related API failure occurs	Set the event severity level, from 1 to 40, to indicate the importance of an event in which a cluster-related API failure occurs. The default is 15 (yellow event indicator).

3.9 ResourceOwnerChange

Use this Knowledge Script to detect whether the owner of a cluster resource has changed. Changes in the ownership of a resource typically indicate a failover or failback operation has taken place. This script raises an event if the owner of the cluster resource changes or if a cluster-related API failure occurs.

Both the event and data detail message indicate the previous cluster resource owner and the new resource owner.

3.9.1 Resource Object

Microsoft cluster resource

3.9.2 Default Schedule

The default interval for this script is **Every 5 minutes**.

3.9.3 Setting Parameter Values

Set the following parameters as needed:

Description	How to Set It
Raise event?	Set to **y** to raise an event is cluster resource ownership changes or when a cluster-related API failure occurs. The default is y.
Collect data?	Set to **y** to collect data for charts and reports. The default is n. If enabled, data collection returns a value of 0 if the cluster resource owner has changed in the interval, or a value of 100 if the resource owner has stayed the same. The detail message indicates the previous resource owner and the new owner.
Event severity level when cluster resource owner changes	Set the event severity level, from 1 to 40, to indicate the importance of an event in which cluster resource ownership has changed. The default is 5 (red event indicator).
Event severity level for cluster-related API failure	Set the event severity level, from 1 to 40, to indicate the importance of an event in which cluster-related API failure occurs. The default is 15 (yellow event indicator).

A Monitoring Clustered Resources and Applications

This section explains how to configure AppManager monitoring support for clustered server applications and cluster resources if you are not using the MSCS Knowledge Scripts. It includes an overview of how to configure cluster monitoring and examples that illustrate the most common configuration scenarios.

A.1 Monitoring Clusters with AppManager

You can use AppManager to monitor Microsoft Exchange, Microsoft SQL Server, and Oracle RDBMS running as virtual servers in a Microsoft Cluster Server. You can monitor cluster resources using the AppManager for MSCS and AppManager for Microsoft Windows modules.

At this time, you can only monitor cluster resources in MSCS. Other clustering products, such as Veritas Cluster Server, are not supported.

However, not all Windows server applications are cluster-enabled. Currently:

- With the AppManager for MSCS module, the agent can monitor any virtual server on the cluster.
- For most modules, the agent monitors the application only on the local node.

For some applications, such as Microsoft Exchange Server, Microsoft SQL Server, and Oracle RDBMS, you can run application-specific Knowledge Scripts on the physical nodes that make up the cluster to monitor the virtual server. For these applications, monitoring is handled through the application-specific module, such as AppManager for Exchange 2008 and 2010 or AppManager for Microsoft SQL Server.

For cluster resources that are not linked to a particular application, monitoring is handled through the AppManager for MSCS and AppManager for Microsoft Windows modules, and the MSCS and NT Knowledge Scripts.

To perform any cluster monitoring, you must install the AppManager agent on the local (not shared) disk of each node in MSCS.

A.2 Planning to Monitor Cluster Resources

Before installing agents for cluster monitoring, you should prepare for the deployment. As with any AppManager-related planning, first evaluate the characteristics of your environment and your monitoring needs.

The questions you need to answer include:

- What resources do you need to monitor in MSCS?
- Which applications are installed as clustered applications?
- Which Knowledge Scripts will you need to run to effectively monitor a given cluster as a virtual server?
- How many physical nodes make up each MSCS?

- What resources have you defined for each cluster group?
- Are multiple instances of the application you want to monitor running on the same node?
- Is the cluster configured as an active/passive cluster or as an active/active cluster? If active/active, how many drive resources can independently fail from node to node?

Create a worksheet for tracking the information you gather, particularly if you are monitoring more complex cluster configurations. For example, if MSCS relies on six or eight physical nodes:

- The Cluster Name resource and Quorum Disk should be located in the same cluster group, thereby guaranteeing that they are active on the same node at any given time. Because the cluster stores resource information on the MSCS directory on the Quorum Disk, the cluster operation fails if you take the Quorum Disk offline.
- If the cluster uses multiple instances of the application server and is configured as an active/active cluster, setting up the AppManager agent on each node and properly configuring the jobs to handle failover on each node requires careful planning.

A.3 Installing on an Active Cluster Node

Before you install the AppManager agent, ensure that the virtual server is active on the cluster node where you are installing the agent and managed object. Typically, discovery of the virtual server can only take place when the virtual server is active on the agent computer.

In most cases, move the active virtual server to each node before installing the managed object in order to discover all of the disk resources associated with the cluster. Or, install AppManager on an inactive node and disable the option to perform discovery as part of the installation. After installation, you can then move the virtual server to the appropriate node and run the discovery job manually.

Because MSCS resources are fully cluster-aware, you do *not* need to fail over any resources in order to discover MSCS resources.

For more information about installing the agent on each cluster node, see the *Installation Guide for AppManager.*

A.4 Monitoring a Log File on a Shared Cluster Disk

This topic provides an example of how AppManager manages cluster resources. The simplest cluster monitoring might involve checking a log file on the shared cluster disk. For this type of monitoring, the Operator Console TreeView should include all of the physical nodes that comprise the cluster, but it should not include any virtual server names.

For this type of monitoring, you should place the physical nodes for a cluster in a unique server group. For example, if you have a cluster of three computers—`BAJA`, `SHARK`, and `CORTEZ`—that form the virtual server `SALESWEST`, create a server folder called `SALESWEST` and add these three physical nodes to the group. For information about adding computers and creating server groups, see the *Operator Console User Guide for AppManager*.

A.4.1 Adding and Discovering the Cluster Nodes

To monitor a cluster of Windows servers:

1. Install the AppManager agent and appropriate module on each of the computers that make up the Microsoft Cluster Server. At a minimum, install the AppManager for Microsoft Windows module.
2. In the Operator Console or Control Center, add the computers to the TreeView if they are not already displayed there. Do not add computers using virtual names, such as the name of the cluster.
3. On the TreeView menu, click **Create Server Groups**.
   - Type a group name for the cluster.
   - Select the physical nodes to be members of the group, then click **Add**.
   - Click **OK**.
4. In Microsoft Cluster Administrator, fail all resources over to the first node. This process allows you to display all cluster disk objects under the NT resource object.
5. Run the Discovery_NT Knowledge Script on the computer you selected in step 4.
6. Repeat steps 4 through 6 for each physical node in the Microsoft Cluster Server. For example, fail the cluster resources over to computer A and run the Discovery_NT Knowledge Script, then repeat the steps for computer B.

A.4.2 Selecting Monitoring Knowledge Scripts

If you are monitoring resources that reside physically on the clustered node, such as network interface cards, CPU, physical memory, power supplies, or voltage, you can run Knowledge Scripts on the physical nodes without any further configuration. However, to monitor shared resources such as shared logical drives, services, or applications, run the AMAdmin_SetResDependency Knowledge Script to identify the resource dependency for each node.

The SetResDependency Knowledge Script enables you to define the resources that a Knowledge Script requires in order to run. The agent verifies whether the resources are there. If the resources are not there, the job is skipped until the next scheduled iteration, which avoids unnecessary or duplicate error messages.

Before you set the resource dependency, review the cluster group to identify shared resources and which resources may failover between cluster nodes. Also verify whether the cluster is an active/passive cluster or an active/active cluster.

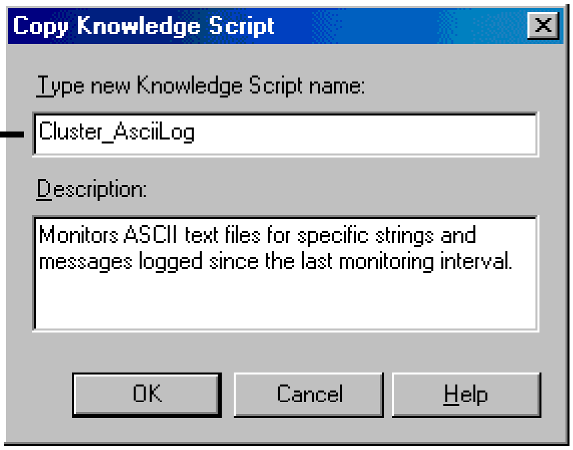
A.4.3 Setting Resource Dependency in an Active/Passive Cluster

In an active-passive cluster, an application or resource is only available on one node, the active node, at any given time. Because only one node is active, you need to configure only one custom Knowledge Script category to include your cluster resource jobs and one AMAdmin_SetResDependency job that identifies the cluster resource that is owned by the active node.

The following example uses the General_AsciiLog Knowledge Script, which monitors a log file on a drive you specify. For this example, this script monitors a log on a shared drive that is only owned by an *active* node in a cluster.

To monitor the shared logical disk in an active/passive cluster:

1. In the Operator Console or Control Center, right-click the General_AsciiLog Knowledge Script and select **Copy Knowledge Script**.
2. Rename the copy to create a new Knowledge Script category. You can use a prefix such as `Cluster` or the virtual server name, for example, `Cluster_AsciiLog` or `SALESWEST_AsciiLog`.


Type a unique prefix for the copy

Copy Knowledge Script

Type new Knowledge Script name: Cluster_AsciiLog

Description: Monitors ASCII text files for specific strings and messages logged since the last monitoring interval.

3. Repeat steps 2 and 3 for each additional monitoring Knowledge Script you want to use. For example, create a `Cluster_LogicalDiskBusy` Knowledge Script. Use the same prefix you used in renaming the AsciiLog Knowledge Script.
4. Run the AMAdmin_SetResDependency Knowledge Script on the server group you created for the cluster.
5. On the Values tab in the Knowledge Script Properties dialog box, set the following parameters:

Parameter	How to Set It
Knowledge Script category	This setting identifies which jobs should only run when the specified resource is owned by the current node. Otherwise, the job is inactive. Specify the Knowledge Script category that you want to be dependent on the resource you specify. For example, type `Cluster`. In this example, if you run this Knowledge Script on the computer `BAJA`, specify the `Cluster` job category and the dependency as the `M:` drive, the jobs you have created for the `Cluster` category will only run on the computer `BAJA` when it is the active node that owns the `M:` drive.
Required available resources	Type the name of a shared resource such as a shared logical disk. For example, `M:` In specifying a resource dependency, you should identify a single resource, whether it is a physical drive or service name, within the cluster group.

For more information about the AMAdmin_SetResDependency Knowledge Script, see the Help for the Knowledge Script.

A.4.4 Monitoring Clustered Resources

After you run the AMAdmin_SetResDependency Knowledge Script on each node in the Microsoft Cluster Server, you can run identical jobs on all of the nodes or on the server group you created for the cluster. For example, you can run the Cluster_AsciiLog Knowledge Script, and any other Knowledge Scripts that require the same logical drive, on the server group `SALESWEST`.

Only the active node sends events and collects data. In the Jobs list, the jobs running on the active node display a status of Running/Active, and the jobs on the passive node display a status of Running/Inactive. In the TreeView pane of the Operator Console, a small clock icon is displayed next to the “inactive” node.

If the cluster group or resource fails over, AppManager automatically begins monitoring the other node.

A.4.5 Using SetResDependency in an Active/Active Cluster

In an active/active cluster, monitoring shared logical drives can be somewhat more complex because each node can own one or more physical drives simultaneously. For example, you have two physical nodes, `BOSTON` and `ATLANTA`, that make up the cluster `EAST`. The computer `BOSTON` owns drives `E:` and `F:` and `ATLANTA` owns drive `G:`. A failover event on the `BOSTON` node can affect one or both drives, so `ATLANTA` may take ownership of just one additional drive or of all three drives.

If the drives are configured as independent group resources that can be owned by either node at any given time, you need to set up the dependency with this scenario in mind. In this case, because you have three different resources that can be owned by either node, you need to create three Knowledge Script categories to handle monitoring.

Using the General_AsciiLog Knowledge Script as an example, the following steps explain what you need to do.

To monitor the shared logical disk in an active/active cluster:

1. In the Operator Console or Control Center, right-click the **General_AsciiLog** Knowledge Script and select **Copy Knowledge Script**.
2. Rename the copy to create a new Knowledge Script category, for example, `ClusterE_AsciiLog`.
3. Repeat steps 2 and 3 to create two additional copies of the Knowledge Script, for example, `ClusterF_AsciiLog` and `ClusterG_AsciiLog`. You now have three copies of the `AsciiLog` Knowledge Script, one for each resource that may fail over:

   ```
   ClusterE_AsciiLog
   ClusterF_AsciiLog
   ClusterG_AsciiLog
   ```

4. Run the AMAdmin_SetResDependency Knowledge Script on the server group you created for the cluster. In the Properties dialog box:
   - Set **Knowledge Script category** to **ClusterE**.
   - Set **Required available resources** to be **E:**
5. Run the SetResDependency Knowledge Script on the server group a second time. In the Properties dialog box:
   - Set **Knowledge Script category** to **ClusterF**.
   - Set **Required available resources** to be **F:**

6 Run the SetResDependency Knowledge Script on the server group a third time. In the Properties dialog box:

- Set **Knowledge Script category** to **ClusterG**.
- Set **Required available resources** to be **G:**

7 Run all three AMAdmin_SetResDependency jobs on both cluster nodes.

8 After you configure the resource dependency, run all three cluster AsciiLog Knowledge Scripts on both cluster nodes. Before you start each of the cluster AsciiLog jobs, open the Knowledge Script Properties dialog box, click the **Objects** tab, and adjust the selected objects as follows:

Knowledge Script	Object
ClusterE_AsciiLog	Drive E:
ClusterF_AsciiLog	Drive F:
ClusterG_AsciiLog	Drive G:

As a general rule, change the selected objects by disabling the objects to which the Knowledge Script does not apply.

A.5 Monitoring Applications in an Active/Passive Cluster

Monitoring clustered applications such as SQL Server or Exchange Server in an active/passive environment is very similar to the monitoring example discussed in Section A.4.5, “Using SetResDependency in an Active/Active Cluster,” on page 37: checking an ASCII log file on a shared cluster disk in an active/active cluster environment. As with monitoring shared logical disks, the Operator Console or Control Center TreeView should include all of the physical nodes that compose the cluster. You should place the physical nodes for the cluster in a unique server group.

As an example of monitoring a clustered application in an active/passive cluster environment, assume you want to monitor SQL Server installed as a clustered application. For this example, the virtual server name is `VirtualSQLServer`, the physical nodes are `ROME` and `PARIS`, and the cluster’s shared disk is defined as drive `F:`.

To monitor SQL Server in an active/passive configuration:

1 Install the AppManager agent and the AppManager for Microsoft SQL Server module on both the `ROME` node and the `PARIS` node.

If necessary, run the Discovery_NT and Discovery_SQL Knowledge Scripts.

2 In the Operator Console or Control Center, create a server group in the Master view called `VirtualSQLServer` and add the `ROME` and `PARIS` servers to the group. For more information about creating server groups, see the *Control Center User Guide for AppManager*.

3 Run the AMAdmin_SetResDependency Knowledge Script on the server group for the cluster, for example `VirtualSQLServer`. In the Properties dialog box:

- Set **Knowledge Script category** to **SQL**.
- Set **Required available resources** to be **F:**

4 Click **OK** to run the job. By default, the job runs once on each computer in the server group to establish the dependency. You should only need to run the Knowledge Script again if you make changes to the cluster membership or the cluster resource.

A.5.1 Setting Values for Resource Dependency

The values you set in the Knowledge Script Properties dialog box identify jobs that should run only when the specified resource is owned by the current node. Otherwise, the job is inactive. Although using a drive letter as a cluster resource dependency is common, as shown in Section A.5, "Monitoring Applications in an Active/Passive Cluster," on page 38, you can set the dependency to be a required active *service* when monitoring clustered applications.

When monitoring a virtual SQL Server, for example, you can set the dependency based on the `mssqlserver$virtualservername` service rather than the logical drive. Run the AMAdmin_SetResDependency Knowledge Script on the computer `PARIS.` Specify the `SQL` job category and the dependency as the `MSSqlServer` service. SQL jobs run on the computer `PARIS` when it is the active node that owns the `MSSqlServer` service.

When specifying a resource dependency, identify a *single* resource, whether it be a physical drive or a service name, within the cluster group where the monitored application is installed.

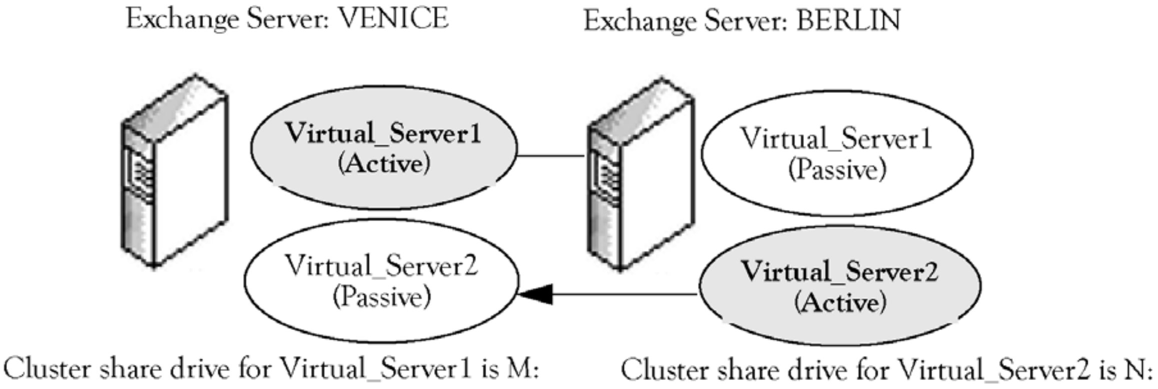
A.5.2 Selecting Knowledge Scripts to Run

Once you have established the resource dependency, you can monitor the cluster using SQL Knowledge Scripts. Run the SQL Knowledge Scripts to monitor the virtual SQL Server resources on the server group you created for the cluster.

A.6 Monitoring Applications in an Active/Active Cluster

In an active/active cluster environment, the servers in MSCS work independently of each other, each carrying its own share of processing load. If failure occurs on the hardware, operating system, or services on one of the independent servers, the failed server's processes are absorbed by the other cluster nodes.

For example, assume a relatively simple configuration with two Exchange 2003 servers with one virtual server instance that runs on each node. If failure occurs on either node, messaging services are automatically transferred to the backup node, which then handles all messaging until the failed node becomes available again.


Exchange Server: VENICE
Exchange Server: BERLIN
Virtual_Server1 (Active)
Virtual_Server1 (Passive)
Virtual_Server2 (Passive)
Virtual_Server2 (Active)
Cluster share drive for Virtual_Server1 is M:
Cluster share drive for Virtual_Server2 is N:

For example, if the active virtual server instance on `VENICE` fails, the messaging services are transferred to the backup node `BERLIN`. Because the node `BERLIN` is now handling the messaging for both `Virtual_Server1` and `Virtual_Server2`, the processing load is potentially much heavier than in an active/passive configuration, but messaging continues uninterrupted.

NOTE: Although the example assumes an Exchange cluster, most of the information can be applied to any active/active cluster configuration. Similarly, this example illustrates a simple configuration, with two physical nodes and two virtual servers. If your cluster has additional nodes and server instances, you can follow the basic strategy presented here, but configuring it to handle the additional nodes and instances is a bit more complex.

For AppManager to continue application-level monitoring when a virtual server fails over from one node to another, you need to configure the AMAdmin_SetResDependency Knowledge Script to identify when jobs should be active on each server in the Microsoft Cluster Server.

A.6.1 Setting the Resource Dependency for an Active/Active Cluster

In order for AppManager to transfer application monitoring between physical nodes if a failover event occurs, you must run each Knowledge Script you want to use on all of the physical nodes that may potentially host a virtual server. Because jobs are placed in standby mode based on the Knowledge Script category and a resource dependency, you need to create custom Knowledge Script categories for each virtual server in the Microsoft Cluster Server.

To create custom Knowledge Script categories and set resource dependency:

1. Select the Knowledge Scripts you want to run and identify the physical nodes and virtual server instances that you will be monitoring. For example, assume you want to run the Exchange_TopNSenders Knowledge Script on an active/active cluster with the two physical nodes:

   - `VENICE` with the active virtual server instance `Virtual_Server1`
   - `BERLIN` with the active virtual server instance `Virtual_Server2`

   Each active virtual server is configured to fail over to the backup node with `Virtual_Server1` dependent on drive `M:` (`VENICE` owns `M:`) and `Virtual_Server2` dependent on drive `N:` (`BERLIN` owns `N:`)

2. In the Operator Console or Control Center, copy and rename each script that you want to run so as to create a unique Knowledge Script category. For example, right-click the Exchange_TopNSenders Knowledge Script and select **Copy Knowledge Script**. Rename the script so that the category indicates the name of a virtual server:

   - `Virtual01_TopNSenders.qml`
   - `Virtual02_TopNSenders.qml`

   You need to do this for each Knowledge Script you want to run against the cluster and for each virtual server instance you want to monitor. Essentially each virtual server in the Microsoft Cluster Server needs its own set of Knowledge Scripts to run.

   NOTE: Create a unique category and server group for each virtual server being monitored.

3. In the TreeView pane, select one of the physical nodes for which you need to set resource dependency.

   During discovery, AppManager discovers both the physical nodes and the virtual servers associated with each node. The TreeView will display all the virtual servers associated with each physical node, not just the node where the virtual server is active. For example:

   `VENICE`

   - `Virtual_Server1` (active)

- Virtual_Server2 (inactive)

  BERLIN

- Virtual_Server1 (inactive)
- Virtual_Server2 (active)

4 Run the AMAdmin_SetResDependency Knowledge Script on the physical node you selected in step 3, for example, onto the server VENICE. The SetResDependency Knowledge Script enables you to define the circumstances when jobs should be inactive:

- In Exchange, a virtual server is only active on one node at a time. If you are monitoring an application that allows multiple virtual servers to be active on the same node at the same time, you need to create additional Knowledge Script copies for each virtual server and carefully identify the resource dependency for each.
- Inactive Knowledge Scripts are simply in stand-by mode on the backup node, waiting for a particular resource to appear, for example, a logical drive or a service that identifies an active instance. If the resource you specify becomes available because of a failover event, the SetResDependency job changes the Knowledge Scripts in the specified category to active mode, and jobs begin running on the backup node.

5 Using the example with the target server of VENICE, set the values in the Properties dialog box as follows:

Parameter	Description
Knowledge Script category	The category you created in step 2. For example, Virtual01 for the active virtual server on VENICE.
Required available resources	The drive that the active virtual server depends upon. For example, the drive M: on which the Virtual_Server1 instance relies.

6 Repeat steps 3 through 5 for each physical node in the Microsoft Cluster Server. For example, repeat the steps for the server BERLIN and set the Knowledge Script Properties as follows:

Parameter	Description
Knowledge Script category	The category you created in step 2. For example, Virtual02 for the active virtual server on BERLIN.
Required available resources	The drive that the active virtual server depends upon. For example, the drive N: that the Virtual_Server2 instance relies on.

7 Run all of the Knowledge Scripts you created in step 2 on all physical nodes in MSCS. If you established a server group for the cluster, run the jobs on the server group. For example, run Virtual01_TopNSenders and Virtual02_TopNSenders on both the VENICE and BERLIN nodes.

In the Operator Console, the job status is displayed as Running/Active or Running/Inactive, with Virtual01 scripts active on VENICE and inactive on BERLIN and Virtual02-related scripts active on BERLIN and inactive on VENICE.

If the virtual server Virtual_Server2 fails, the Exchange processes for Virtual_Server2 will fail over from BERLIN to VENICE. The server VENICE takes ownership of drive N: and AppManager recognizes this change through the SetResDependency script.

Upon failover, the status in the Job pane changes so that for the server VENICE:

- Virtual01_TopNSenders job would be Running/Active
- Virtual02_TopNSenders job would be Running/Active

And on BERLIN, the job status for both Virtual01_TopNSenders and Virtual02_TopNSenders is Running/Inactive.

Monitoring, automated actions, and data collection continue from the new physical node without interruption. Because data stream information is associated with the physical node, not the virtual server, graph data may appear disjointed.

A.7 Using Troubleshooter to Check Cluster Configuration

You can use the Troubleshooter tool to check the application monitoring for a virtual server.

To use the Troubleshooter:

1. In the Operator Console, select a cluster node.
2. On the TreeView menu, click **Troubleshooter**, click **Client Resource Monitor Info**, and then click **Application Monitoring Status**.
3. Check the resource dependencies settings. The report should look similar to the following example:

```
<RSC #0> :
        Name       =>  CLUSTER1
        JobCnt     =>  0
        Status     =>  Inactive
        ResDepend  =>  e:
        SvcDepend  =>
<RSC #1> :
        Name       => CLUSTER2
        JobCnt     =>  0
        Status     =>  Active
        ResDepend  =>  f:
```

A.8 Resetting Resource Dependencies

Normally, you establish the dependencies for your cluster only once, unless you make changes to the cluster membership or the cluster resources. However, if you shut down the NetIQ AppManager Client Resource Monitor service (NetIQmc) with the -o option, any resource dependencies you configured will be lost. If you use the -o option to restart the NetIQmc, then re-run the AMAdmin_SetResDependency Knowledge Script to reconfigure the resource dependencies.

To remove a previously configured resource dependency, run the AMAdmin_SetResDependency Knowledge Script on each node to be reconfigured, then specify the desired Knowledge Script category, such as SQL or Exchange. Leave the **Disks** and **Services** resource fields blank.