This document is intended to provide a general outline of the steps that are required to configure MDaemon for high availability. Modifications may be required depending on differences in each specific IT infrastructure.

Businesses today rely heavily on networked applications to be productive. When applications fail or become unavailable to users, productivity will undoubtedly suffer. Furthermore, as applications become increasingly complex, they become much more difficult to troubleshoot when they fail. This can potentially result in extended periods of application downtime, which usually means lost productivity for users and, in some cases, revenue for an enterprise.

High Availability

One method to provide more resiliency to an application, and to extend its uptime for its users, is to provide a secondary server /application which will 'backup' the first in the event that it fails. In this way you can ensure that users experience no downtime in the event of an application failure. This also allows IT staff time to troubleshoot the primary application server without user disruption and can provide a means of allowing application maintenance to be carried out without impacting users. This is generically referred to as ‘High Availability’.

There are many options with regards to providing this sort of resiliency. These range from simple DNS tricks to hardware-based load balancing gear. The goal is to be able to provide a solution to recover a failed application in such a way that users are never impacted. This can be very complicated to deploy and can cost thousands of dollars to implement.

One method is to group multiple application servers supporting an application into what is known as a ‘Cluster’. This cluster appears to users as a single application server. The nodes that comprise the cluster are all able to, at any time, handle user requests. The cluster can also detect failures and shift the requests for an application to only the servers that are available to serve them, thus ensuring that the application is always available.

The solution described in this document demonstrates one method for deploying the MDaemon Email server in a networked cluster to provide fault tolerance and recovery of MDaemon services in the event of a server failure. It works like this: We have two servers both with MDaemon installed, one server being the primary and processing all user traffic and with a backup server standing by to take over request processing in the event of a failure of the primary. While there are several methods for accomplishing High Availability with MDaemon, most of these solutions require expensive Load Balancing hardware, complicated network configurations, and data synchronization tools to be effective. The following solution utilizes only software tools built into the Windows operating system.

For a clustered application deployment to be effective, the application configuration must provide the following things:

• Server failure detection – the cluster solution needs to be able to detect a failure
• Traffic Management – There needs to be a method to direct user requests only to servers which are ready and able to service those requests.
• Synchronized Data - All members that are part of a resilient application deployment must have replicas of the application data so that in the event they are required to handle user requests, they have the most up to date version of the application data to do so.
The solution described here accomplishes all the above requirements. It does this by making use of the following software tools:

- Microsoft’s Network Load Balancing (NLB) for traffic management.
- Microsoft’s PowerShell for failure detection and correction.
- Microsoft’s Distributed File System Replication (DFSR) for data synchronization.

Please keep in mind the above tools are in no way supported by Alt-N Technologies. They were merely used to facilitate the solution described in this document. Support for these tools may be provided by Microsoft.

**NOTE:** The solution outlined here was tested in a production environment using live user traffic and data. As mentioned above, it was tested using Microsoft’s DFSR for file synchronization between cluster nodes. There are several tools freely available for use in the synchronization of user data between multiple servers. DFSR however, provides near real-time synchronization of user data, which makes it uniquely suited to synchronizing the data between a set of busy Email servers.

**Requirements**

The following is a complete list of requirements needed to implement this solution in a network environment:

- Two Windows 2008R2 Servers (this solution was tested using physical servers but virtual servers could potentially be used). These servers should have the latest operating system patches and service packs applied. They should be configured to use the same LAN and IP Subnet
- MDaemon 14.5 or later installed and configured on both servers with the MDaemon Windows service start-up option set to ‘Manual’
- The Windows Network Load Balancing Feature installed and configured on both servers.
- A third NLB ‘control server’ for making changes to the NLB configuration on the cluster nodes. Because of the changes made to the networking stack on the servers, using one of the cluster nodes to configure the cluster can lead to unexpected results or even failed configurations and is not recommended by Microsoft. This server should also have the NLB cluster feature installed, although it does not actually participate in the NLB cluster
- Working Active Directory environment with a Windows Server 2003 or higher domain functional level.
- Microsoft’s Distributed File System Replication feature installed on both servers with the latest available DFSR updates and patches. Information on available DFSR patches can be found at the following link: [http://support2.microsoft.com/KB/968429](http://support2.microsoft.com/KB/968429). All patches and hotfixes listed in this link should be installed.
- For networks using Network Address Translation (NAT), a NAT rule to NAT outbound connections from both cluster hosts to the same public IP source address. This allows both cluster hosts to appear as the same host regardless of which cluster host is active and is needed when sending email messages. This is only required if NAT is being used for inbound connections to the cluster IP address.

**Overview**

The solution outlined here provides failover and redundancy of the MDaemon application by using a pair of MDaemon servers in a network cluster. The software which controls which server handles user requests is Microsoft’s Network Load Balancing feature. Traffic is sent to a ‘Cluster IP Address’, which is a virtual shared IP address used by the network to represent the current active cluster member. This address is shared by both servers and is used to connect to either server depending on current server availability. This is different from the IP address assigned to the server itself. Traffic
sent to the Cluster address is forwarded to the current active cluster member by the NLB service as if it was sent directly to that member.

This solution works in the context of there being a designated primary server and a designated backup server. The primary server is the server which normally takes all network traffic. It is also the server where all configuration changes would normally take place. The secondary server is the server which begins to take network traffic only in the event that the primary server has failed for some reason. Normally, it sits idle, with the MDaemon service stopped. When a service failure is detected on the primary server, the MDaemon services are started on the secondary server, and NLB begins to shift user requests to it instead of the Primary server.

Additionally, when the primary server comes back online, traffic is automatically shifted back to it. The idea is that the Primary server, if able, will always handle network requests.

Configuration changes to the MDaemon application would be made on the primary server. Because of the data synchronization used in this solution, those changes would be synchronized to the secondary server so that in the event it needed to begin processing traffic, it would have the most current configuration information.

Below is a high level diagram which illustrates the setup and layout of the described solution.

As you can see in the diagram, when traffic is sent to the cluster IP address, it is forwarded to the primary cluster member. In the event of a network failure, traffic is sent to the backup secondary cluster node as illustrated below.

This allows traffic to seamlessly transition from one server to another in the event of a network failure. The IP address clients are connecting to never changes, even though the traffic is sent to a different machine.
It is important to note that while Microsoft’s NLB can detect network failures, it has no ability to detect application level failures, such as could be experienced by an application crash or other software error. In situations where an application has crashed, but the network on an active NLB client is still functional, client requests would still be directed to that failed NLB node. This would result in client requests failing.

To correct for this, this solution uses a custom PowerShell script which is used to detect application level failures and to interact with the NLB Cluster Service to direct traffic to healthy nodes in the cluster. It does this by testing a specified set of TCP ports, and in the event of any of those ports failing, shifts traffic to a backup working cluster node. It is designed to run on a schedule on both servers in the NLB cluster. The script is built on the previously mentioned notion that there is a primary MDaemon server which should normally take all network traffic, and there is a backup or secondary MDaemon server which will take over if the primary has failed for some reason. This is an important point as it requires that the NLB clustering be configured in a very specific way for this script to work.

**Setup and Configuration**

After installing the server operating system and the MDaemon software, the basic steps needed to implement the solution described in this guide are as follows:

- Install the DFSR feature
- Configure the DFSR feature
- Install Microsoft’s Network Load Balancing feature
- Configure NLB for use with our MDaemon servers
- Add a scheduled task to run our failure detection script

To begin, we need to have both servers up and running with Windows 2008R2 installed. Each server should have the latest updates applied via Windows Update. Each server should be connected to the same LAN and have an IP address assigned in the same subnet.

**NOTE:** It is advised that the servers you use are new servers not currently being used in production. After following the steps outlined here, you should fully test this solution to ensure that you understand what it does and how it works. Additionally, once this solution has been deployed, the pair of MDaemon servers which comprise the cluster are locked together. Any change made to one server will be replicated to the other. Please keep this in mind when making configuration changes or stopping and starting MDaemon servers via the MDaemon UI. Also, when using this configuration, using IIS with WorldClient and MDaemon Remote Administration is not supported. Users will need to configure MDaemon to use MDaemon’s built-in webserver for use with WorldClient and MDaemon Remote Administration.

Next, MDaemon 14.5 or later should be installed on both servers. We need to determine which server will be designated primary and which server will be designated secondary. It is important to decide now which role each server will be assigned as this information will be used throughout the rest of this guide.

Now, we will want to fully configure MDaemon on the primary server. This would include any accounts or security settings required for MDaemon.

Next, we will want to install and configure DFSR to replicate configuration data from the primary server to the secondary. DFSR is a Microsoft technology developed initially to support the Distributed File System. It is designed to provide near-real-time two-way replication of large sets of files and is commonly used to synchronize file sets between branch offices. It uses a delta mechanism called Remote Differential Compression which allows it to only transfer the
‘changes’ to files instead of constantly copying whole files when something changes within the file. This makes it very suited for low bandwidth environments and makes it quite fast.

NOTE: DFSR should not be substituted for a regular backup rotation. It is advised that data is backed up as appropriate even with DFSR or some other synchronization solution deployed.

NOTE: As mentioned previously, DFSR requires a working Microsoft Active Directory to store its configuration data. If you do not have an operational Active Directory and have not yet installed and configured an Active Directory Domain Controller as well as the supporting DNS services, you will need to complete this procedure before proceeding. Configuring Active Directory is beyond the scope of this guide.

To install and configure DFSR for use with MDaemon, proceed through the following steps:

- First, we start Server Manager on the primary server (click Start, point to All Programs, point to Administrative Tools, and then click Server Manager). Then click on ‘Roles’ and click ‘Add Roles’. Then click ‘Next’.
- Place a check mark on ‘File Services’ and then click ‘Next’.
- On the next screen, click ‘Next’.
• On the Role Services screen, place a check box on ‘Distributed File System’ and uncheck ‘DFS Namespaces. Then click ‘Next’.

• On the confirmation screen, click the ‘Install’ button. Wait while the wizard finishes the installation of DFSR on your server.

• When the wizard is finished installing DFSR, ensure that installation was successful and click the close button.

You have now installed DFSR on your primary server. 

Repeat the above steps on your secondary server before continuing.
Once DFSR is installed on both servers, we need to configure it to replicate the data contained within MDaemon. This will include the MDaemon configuration data as well as the MDaemon user data. We do this by configuring a specific set of MDaemon folders for data replication. These folders will vary depending on the MDaemon features and applications installed on the server.

The following is a list of folders which need to be replicated when doing HA with an MDaemon installation:

<install drive>:\MDaemon\App
<install drive>:\MDaemon\Domains
<install drive>:\MDaemon\Users
<install drive>:\MDaemon\Public Folders
<install drive>:\MDaemon\Signatures
<install drive>:\MDaemon\WebAdmin
<install drive>:\MDaemon\WorldClient
<install drive>:\MDaemon\Digests
<install drive>:\MDaemon\Pem
<install drive>:\MDaemon\SpamAssassin

If you have installed SecurityPlus for MDaemon you should also replicate the following folder:

<install drive>:\MDaemon\SecurityPlus

DFSR uses what’s known as a ‘Replication Group’ to group synchronized folders under a single unified management entity. The folders that actually contain the data you want synchronized are known as ‘Replicated Folders’, and will appear under the replication group when they are created.

The following steps illustrate the process for creating a replication group and a set of replicated folders on your servers:

• On the Primary server, open Server Manager. In the left pane, expand Roles  File Services  DFS Management  Replication. Right click on ‘Replication’ and choose ‘New Replication Group’.
Choose ‘Multi-purpose Replication Group, and choose next.

On the next screen choose a name for the replication group, and optionally type a description in the description field. Make sure your Active Directory domain appears in the domain field. Your domain may already be pre-populated in this field. If not, you can choose the browse button to find your domain on the network, or type it in this field manually. Again this is the domain name of your Active Directory domain.

Next we need to add the servers which will participate in the replication group. These are the MDaemon servers which will be part of your HA group. We need to add at least two servers to continue. Click the add button. Type in the hostname of the server we designated as our Primary Server, and then click OK. Once added, click on the ‘Add’ button again to add the hostname of the designated Secondary server, and click OK. Once completed we should see our two MDaemon servers listed. Once both servers are listed, click next.
• In the next screen, make sure that the ‘Full Mesh’ option is selected. Click ‘Next’.

• Next we ensure that the ‘Replicate Continuously using specified bandwidth’ choice is selected and click next.

• We now need to choose the primary member of this replication pair. This is the member which holds the initial file set. Replication will be done based on the contents of this member’s files. Once we create a set of replicated folders (later in this document), decisions of what files are replicated to the Secondary member will be based on the contents of the replicated folders on the Primary member of this replication pair. We want to click on the drop down box and choose the hostname of the server we designated earlier as our Primary MDaemon Server. Then click ‘Next’.
We now choose the folders we want to replicate between DFSR replication group members. Refer to the list of folders we need to replicate at the beginning of this list of steps. First we click on ‘Add’ to add a replicated folder pair. Browse to the folder you want replicated. As an example we would add C:\MDaemon\App. We would click on ‘Add’, then in the dialog that comes up, we would browse to the folder at C:\MDaemon\App and click OK.

Notice we are using the existing permissions for the C:\MDaemon\App folder. Whatever permissions are on the folder on the Primary replication member, will be replicated to the secondary member. Repeat this process for each folder in the list at the beginning of these steps. When finished you should have something that looks like this:

Now click ‘Next’.
• The wizard will ask you to choose a partner that is part of the replication group that you want to replicate this folder with. It will also want you to designate the local path of where the data should be replicated to on the secondary replication partner. The secondary replication partner should already be listed in the window. Ensure that it is highlighted and click the ‘Edit button’. In the ‘Membership Status’ section, ensure that ‘Enabled’ is selected. Now click ‘Browse’ and browse to the same folder on the Secondary Server as on the Primary Server. In this example we want to choose C:\MDaemon\App. Ensure that the ‘Make the selected Replicated folder on this member Read-only’ box is unchecked and choose OK. Wait while the wizard configures the selected folder for replication. Once the wizard is done, click ‘Next’. Repeat these steps for each replicated folder you created in the previous step.

• Once you have repeated this process for each replicated folder you created, click ‘Create’.

• Once the wizard is finished you should now see a wizard confirmation page.
You have now successfully completed replication of data between your primary and secondary MDaemon servers. Click ‘Close’.

Configuring NLB

This next section details the steps needed to install and configure Microsoft’s Network Load Balancing feature to provide High Availability for the MDaemon Email server. Microsoft’s Network Load Balacing feature is a Layer 2 technology which provides network resiliency and redundancy to a Windows server and allows network traffic to be directed to a working server in the NLB cluster based on network status. It does this by manipulating the MAC/IP pairing of an IP address to direct layer 2 traffic to a working cluster node.

NLB has two basic modes of operation: Unicast or Multicast. Which NLB method you choose depends on your environment and each will offer different functionality within your network. The reason for this is in the way MAC addresses are assigned between the different modes. The different mode connectivity characteristics are outlined in the following table:

<table>
<thead>
<tr>
<th>NLB Mode</th>
<th>Number of NICs in Server</th>
<th>Functionality</th>
<th>Where is NLB Managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicast</td>
<td>Single</td>
<td>Communication to all network hosts in the same subnet and other subnets but not to other NLB Nodes</td>
<td>Control Server</td>
</tr>
<tr>
<td>Unicast</td>
<td>Dual</td>
<td>Communication to all network hosts in the same subnet and other subnets as well as other NLB nodes</td>
<td>Control server or either NLB node</td>
</tr>
<tr>
<td>Multicast</td>
<td>Single</td>
<td>Communication to all nodes in the subnet and to other NLB partners but not to hosts in other subnets</td>
<td>Control server or either NLB node</td>
</tr>
<tr>
<td>Multicast</td>
<td>Dual</td>
<td>Communication to all hosts in the subnets, all NLB partners, and all hosts outside the subnet</td>
<td>Control server or either NLB node</td>
</tr>
</tbody>
</table>

Additional information on NLB networking characteristics can be found at the following link: [http://technet.microsoft.com/en-us/library/cc782694(v=ws.10).aspx](http://technet.microsoft.com/en-us/library/cc782694(v=ws.10).aspx)

To begin we need to install the Microsoft Network Load Balancing feature. To do this, follow the steps below:
• Log into the Primary server with an account with administrative privileges. Start the Server Manager, and in the left pane, click on ‘Features’.

![Server Manager Features](image)

• Now Click on ‘Add Features’. In the window that pops up, scroll down until to find ‘Network Load Balancing’. Place a check mark in the box next to that option and then click next.

![Add Features Wizard](image)

• Next, we click the ‘Install’ button to start the install process.

![Confirm Installation Selections](image)
Now you should see a wizard confirmation like the one below.

You have now successfully installed Microsoft’s Network Load Balancing feature on your Primary Server. Next we click the ‘Close’ button and then close the Server Manager application. Complete this same process on the Secondary server.

At this point, we need to configure the two servers to work together in an NLB cluster. To do this we need to create a new cluster and assign both the Primary and Secondary servers to that cluster. An NLB cluster can be configured in such a way that all network traffic sent to the cluster IP address will be forwarded to a single cluster host unless that host becomes unavailable. In that instance, NLB will detect that the failed cluster host is no longer responding and will automatically start sending traffic to other cluster hosts. This allows for the transition of traffic from a host that has failed to one that is operational all without any user intervention. This is the way we will configure our cluster to operate.

**NOTE:** As mentioned earlier, it is recommended that you perform the following steps on a server other than either of the servers which will actually participate in the cluster. It has been observed that configuration errors can occur during the cluster creation process if the cluster is being created on a server which will be added to the cluster. This typically happens when configuring the cluster to use Unicast mode, but can happen in either mode. To avoid setup issues with NLB, it is advised to use a third, non-participating server to ensure a smooth NLB deployment when possible. In order to use a third server for NLB configuration, you will need to have Windows 2008 R2 and the NLB feature installed on this server before proceeding.

The following steps details how to configure an NLB cluster:

1. First, we need to create a new cluster. We start by launching the Network Load balancing Manager. Click on Start → Administrative Tools → Network Load Balancing Manager.
• Now in the left pane, right click on the ‘Network Load balancing Clusters’ node and choose ‘New Cluster’

![Image of NLB Manager]

• In the wizard that is shown, enter the IP or hostname of our Primary MDaemon server and click on ‘Connect’. Once the NLB Manager connects to the host, it will appear in the pane below. Once it is visible, click next.

![Image of NLB Cluster configuration]

• Now we need to choose the IP address we will use for the host in this NLB Cluster. In the list of IP addresses in the window (you may see multiple depending on your hosts NIC configuration), highlight the address you want the cluster to use for this host, ensure that the priority is set to ‘1’, and the ‘Initial Host state’ is set to ‘Started’. Then click ‘Next’.

• Next we need to choose the IP address of the cluster itself. This is the IP address that clients will use to connect to your MDaemon servers. You need to have an IP address in the same subnet as the IP addresses assigned to both cluster nodes. For example, if your Primary and Secondary MDaemon servers are assigned with 172.16.0.31/24 and 172.16.0.32/24 respectively, then your cluster IP would need to be an un-used IP address in the 172.16.0.0/24 range. Once this IP address has been determined, click on ‘Add’, ensure that ‘Add IPv4 Address’ is selected and type in the IP address and subnet mask you would like the NLB cluster to use. Then click ‘OK’ and then click ‘Next’.
• On the next screen, we need to choose a cluster operation mode. This is the method by which the cluster directs network traffic to the correct cluster node.

Ensure that the IP address we want to use for the NLB cluster is selected in the drop-down and that the ‘Cluster operation mode’ selected is ‘Multicast’. You can optionally type in the ‘Full Internet Name’ in the provided field. This would be something like ‘mail.domain.com’, however, this is not required. Then click ‘Next’.

![Cluster Configuration Screen]

• On the Port Rules screen we want to highlight the ‘Defined Port Rules’ and click ‘Edit’. Ensure that the port range is 0 to 65535, that the protocols are set to ‘Both’, and that ‘Single Host’ is selected, and then click ‘Ok’. Then click ‘Finish’.

![Port Rules Screen]

• Now that we have our cluster created, we need to add the Secondary server to the cluster configuration. We do this by right-clicking on the cluster IP address shown in the left pane of the NLB Manager window, and choose ‘Add host to cluster’.

![Add Host to Cluster Screen]
• Next we want to type in the IP address of the Secondary server and click ‘Connect’. Once the wizard connects to the host, it will appear in the pane below. Then click ‘Next’.

![Add Host to Cluster: Connect](image)

• Now we need to ensure that the priority is set to ‘2’ or higher, and that the initial host state is set to ‘Started’. Then click ‘Next’.

![Add Host to Cluster: Host Parameters](image)

• On the final wizard screen, simply click ‘Finish’.

![Add Host to Cluster: Port Rules](image)
We have now successfully created a new NLB cluster.

**NOTE:** It is recommended that you test the functionality of the cluster to ensure that failover works correctly in your environment. You should be able to send continuous pings to the cluster IP you configured in the steps above from another network host, disconnect the network connection on the primary NLB node, and still get ping responses from the cluster IP. If this does not work correctly in your environment, you should troubleshoot the issue before proceeding through the rest of this guide.

**Server Health Checking Script Setup**

Now that we have configured the servers to be part of an NLB cluster, we need to setup up our NLB health checking. As mentioned before, Microsoft’s NLB can only detect network failures of the cluster nodes in an NLB cluster. For example, a node that has a failed NIC card or has had its network cable unplugged. NLB does not have the ability to detect when the services fail.

For example, if a cluster node was the active node in the NLB cluster, and had an FTP server running on it, and that FTP server were to crash and stop functioning while the cluster node was still on the network, NLB would continue to send FTP traffic to that node. It cannot determine that the FTP server has crashed and it should no longer send FTP client traffic to it. To NLB, the node is still functioning normally because it is still reachable on the network. This scenario would result in FTP client connections still being sent to that node to fall dead as there is no longer an operational FTP service on that node to answer them.

To correct this, we have developed a script, which when run on a schedule, will test each cluster node’s offered services by making TCP connections to them and waiting for a response. If the service responds, then the script stops. If the service fails to respond, the script checks it again. If it fails a second time, it will interact with the NLB manager to shift traffic to the backup server automatically. In this way, we give NLB the ability to ‘detect’ service level failures and move traffic to an appropriately active server.

The script is written in Windows PowerShell. It’s designed to work with only two cluster members. Your servers first need to be configured to allow the script to execute. Then we need to try running the script from a PowerShell console to ensure it works properly. Once those tests succeed, we then need to set up the script to run at specified intervals. We will use the Windows Task Manager to do this. The following section outlines the steps needed to get this configured:

- First, we want to download the script file from the following site:

- Next, we want to configure the server to allow unsigned scripts to run. To do this, log into your primary server as an account with administrative privileges. Right-click on the Quick Launch PowerShell icon and choose ‘Run as Administrator’. Click ‘Yes’ to any UAC dialogs you see.
• Next, type the following command: Set-ExecutionPolicy RemoteSigned. Type ‘Y’ at the prompt and hit ‘Enter’.

![Image of PowerShell window with command input]

• Next, we need to test the script to make sure it runs properly. Find the directory where you downloaded the script and switch to it in the console window. To do this type cd <driveletter>:\<path>\to\<script>. For example if the script is located in C:\temp you would type ‘cd c:\temp’. In our example here, our script is located in the root of the C drive, so we just type ‘cd ‘ and hit ‘Enter’.

![Image of PowerShell window with directory change]

• Next we want to actually try to run the script. The basic syntax is as follows:

```
\hahealthchecking.ps1 –primary <hostname of primary server> -secondary <hostname of secondary server> -ports <TCP ports to check> -log –logfile <path to logfile> -alertaddress <email alert address> -fromaddress <email address to send from> -smtpport <TCP Port to use to connect to mailserver when sending an alert> -emailuser <emailusername> -emailpass <email password> -securealert <(optional)used to specify that the script use TLS when sending an alert E-mail>
```

The script uses various command line options which can be used to control how the script runs. These options are listed below:

- **-primary** – This tells the script which server it should consider as the server which normally takes network traffic. The script will always try and make this server operational.

- **-secondary** – This tells the script which server it should consider as the server which only takes network traffic in the event the primary server fails its health check during script execution.

- **-ports** – this is a comma separated list of ports the script should check on the MDaemon servers. For example ‘ports 80,443,25’

- **-log** – Turns on script logging. Requires a –logfile switch detailed next.
-logfile – Specifies the logfile and logfile path the script should write to. Example: -logfile C:\Logs\hascript.log

-alertaddress – Specifies an email address to send a message to in the event that the HA cluster is forced to failover. Example: -alertaddress admin@domain.com

-fromaddress – Specifies the Email address the message should come from. Example: -fromaddress haadmin@domain.com

-smtpport – Specifies the TCP port the script should use when trying to send the email. Example: -smtpport 25 This can be changed based on your needs.

-emailuser – Specifies the username to use to authenticate when sending the Email alert

-emailpass – Specifies the password to use to authenticate when sending the Email alert

-securealert – Specifies that the Email alert connection should be done using SSL/TLS

NOTE: When the above Email options are included, the script will connect to the Secondary node you specify on the command line to send the alert Email. Also, in addition to controlling the Active node in the NLB cluster, the script will also control the state of the MDaemon Windows service when it runs. It is advised that you set the Windows MDaemon service startup option to ‘Manual’ and let the script control the service state.

Here is a script execution example:

C:\\ hahealthchecking.ps1 –primary mail1.domain.com –secondary –mail2.domain.com –ports 25,80 –log –logfile C:\logs\hahealthchecking\hahealthchecking.log –alertaddress admin@domain.com –fromaddress haadmin@domain.com –smtpport 25 –emailuser admin@domain.com –emailpass Password –securealert

- We should now try and run this script in a PowerShell console to make sure it runs properly in the current environment and fix any issues before moving on to the next section.
If the script runs successfully you should see some output like the following:

```
Script Started at 09/25/2014 11:41:16
Starting Health Check....

***** Activenode is Primary *****

Checking the status of the MDaemon Server Service......
MDaemon Service on Primary Server is now running
Checking Service Ports....

mail1.domain.com: Port 25 is open
mail1.domain.com: Port 80 is open

Script completed 09/25/2014 11:41:18
```

We can see in the above output that our designated Primary MDaemon server is currently the Active Node in NLB, that the MDaemon service is started, and that the specified TCP ports responded to connection requests.

Now that the script seems to run successfully, we need to set the script to run on a schedule. We want to do this so that the script can run unattended. We also want to set the script up to run fairly often so that it can quickly recover from a server failure. The more times it runs, the sooner an application failure can be corrected.

**NOTE:** It is advised that this task be disabled if maintenance needs to be done on the MDaemon servers individually. Aside from failing over network traffic, it also does things like check that the MDaemon windows service is running and in some instances tries starting it if it’s not. This could lead to unexpected issues when trying to perform upgrades or maintenance on the MDaemon cluster nodes.

The script needs to be run under the context of a user account which has Administrative privileges on the server on which it runs. That user also needs to be granted the ‘Log on as Batch Job’ service right in the local security policy of the server. The steps needed to complete these tasks are detailed in the following section and will need to be repeated for each server in the NLB cluster:

- First we need to create a user account the script will run in the context as. This can be either a local or domain account. We can also use an already established account within the organization. The details of how to create user accounts are beyond the scope of this guide.
Once the user account is created, we need to assign it the correct rights and privileges. First we open the local policy manager by clicking on ‘Start’→‘Administrative Tools’→‘Local Security Policy’.

In the left pane, we need to expand ‘Local Security Polices’ and click on ‘User Rights Assignment’. In the right pane, scroll down until you see ‘Log on as Batch Job’. Double-click it and in the dialog box that pops up, click on ‘Add User or Group’ and then find your user account and click ‘OK’. Click ‘OK’ again. We can now close the Local Security Policy manager.
Next, we need to make sure that the user account that will be running the script is part of the local Administrators group. Open Server Manager, and in the left pane expand ‘Configuration’ and then expand ‘Local Users and Groups’. In the right pane, double-click on ‘Administrators’ and ensure that the user account we want to use is in the box. If it’s not click on ‘Add’ and find the user in the dialog that pops up. Once the user is selected, click ‘OK’ and then click ‘OK’ again.

Next we need to create a scheduled task that will run this script repeatedly. With Server Manager still open we want to expand ‘Task Scheduler’ in the left hand pane, right-click on ‘Task Scheduler Library’, and click ‘Create Basic Task’.

Type in the name of the task in the Name field and optionally add a description and click ‘Next’.
• Ensure that the trigger option selected is ‘Daily’ and click ‘Next’.

[Image of Task Trigger wizard with selected Daily trigger and next button]

• Set the ‘Recur every X days’ to 1 and click next.

[Image of Task Trigger wizard with Recur every 1 days]

• In the ‘Action’ section we want to ensure that the ‘Start a Program’ option is selected and then click ‘Next’.

[Image of Task Trigger wizard with Start a Program selected]
In the ‘Start a Program’ section, we want to type ‘powershell.exe’ in the Program/script field, and then in the ‘Add arguments (optional)’ field type in the command line script as we would run it in a PowerShell console without the ‘.’ in front of it. For example if we ran the script like this in a PowerShell console:

C:\HAHealthChecking.ps1 -primary mail1.domain.com -secondary mail2.domain.com -ports 25,80 -log -logfile C:\temp\script.log -alertaddress admin@domain.com -fromaddress haadmin@domain.com -smtpport 25 -emailuser user@domain.com -emailpass N0tF0rY0u! --securealert

We would put the following in the arguments section:

C:\HAHealthChecking.ps1 -primary mail1.domain.com -secondary mail2.domain.com -ports 25,80 -log -logfile C:\temp\script.log -alertaddress admin@domain.com -fromaddress haadmin@domain.com -smtpport 25 -emailuser user@domain.com -emailpass N0tF0rY0u! --securealert

Once this has been filled in, click ‘Next’.

Ensure that the ‘Open the Properties dialog for this task when I click finish’ check box is checked and then click ‘Finish’.
On the general tab, configure this task to run under the context of the user we added to the local administrators group in the previous section. Click on the ‘Change user or group’ button and select the user we added to the local administrators group in the previous section. Then click ‘OK’. Also, ensure that the ‘Run whether user is logged on or not’ option is selected, that the ‘Run with highest privileges’ is checked, and the ‘do not store password’ check box is unchecked. Ensure that the ‘Configure for:’ drop down is set to ‘Windows 7, windows Server 2008 R2’.

Click on the ‘Triggers’ tab. Double-click on the ‘Daily’ trigger. In the window that pops up, ensure that there is a check mark in the ‘Repeat task every:’ checkbox, that the drop down is set to 5 minutes and that the ‘for a duration of:’ drop down is set to 1 day. This is the setting that controls how often during the 1 day recurrence the task runs. Again, the more often the script runs, the quicker our HA servers will recover from a failure. If the task is set to repeat every 5 minutes, then it’s possible that a full 5 minutes could go by before a failure is detected and corrected. This may or may not be adequate for your environment. It is recommended that the ‘Stop task if it runs longer than:’ check box is checked and that the drop down is set to 30 minutes. This ensures that the task will not run indefinitely if there is a problem with its execution.

Click ‘Ok’ and then click ‘Ok’ again to close the Task Setup window. When you close this window, it should pop up a dialog window prompting you for the password of the user account the task will run as. Enter the password and then click ‘OK.’
We should now have a task configured to run our Health checking script on a schedule. Repeat this process on the Secondary server. Make sure that the task runtime on the Secondary server is offset from the Primary server to prevent the two servers from running the script at the same time.

If needed, we can test the newly created script by right-clicking on the task and clicking ‘Run’. Then we can check the logs to make sure it worked properly.

**Issues with Network Address Translation**

One last consideration for those who utilize Network Address Translation on their networks. When using NLB as outlined in this document, traffic needs to be sent to the cluster IP address we have configured for our cluster during the configuration process. All user traffic is sent to this IP and NLB will direct the traffic to the ‘real’ server within the NLB cluster. When the cluster nodes in an NLB cluster initiate connections, they do so NOT with the NLB cluster IP, but with their own local NIC IP address.

This presents an issue for users of NAT coupled with this HA solution and can cause problems with PTR lookup queries. Typically, users utilizing NAT would have inbound connections destined for a configured public IP, translated to that of the cluster IP address, which would then have NLB send the traffic to the appropriate Cluster node. Your forward and reverse DNS records would be configured with this public NAT IP address. But when these MDaemon servers send traffic outbound, they would do so using their local IP addresses. An outbound NAT rule would need to be configured to ensure that outbound connections from the cluster hosts to the Internet would have their source IP addresses translated to that of the cluster NAT public IP address. This way either host looks to the Internet as if it were the public cluster IP address regardless of which host is the active NLB node.

*In summary, this guide has provided one option for those wanting to run the MDaemon E-mail server in an HA configuration. There are many others but this one allows for High Availability with no additional cost for an Enterprise. This should help to provide redundancy and fault-tolerance to users of MDaemon and ensure that your MDaemon server is always available and online.*