

Checkmarx Application High Availability Cluster (Active-Active)

Contents

[Introduction](#)

[Prerequisites](#)

[Checkmarx HA cluster installation and configuration](#)

[Setting up ASP.NET State service - Enabling remote connectivity](#)

[Appendix I – Install and Configure MS NLB feature](#)

[Appendix II - Setup SSL connectivity in a cluster environment](#)

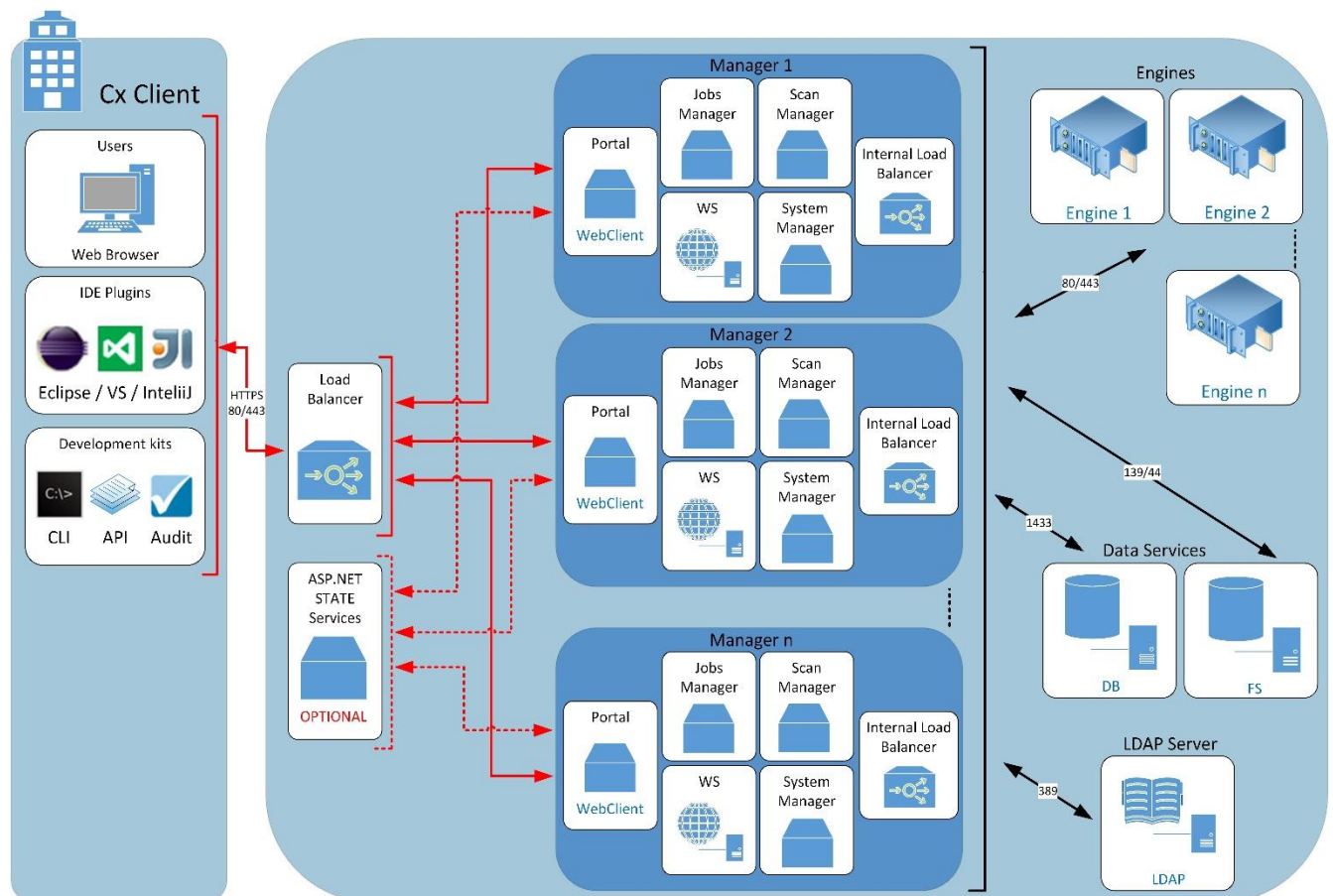
[Appendix III - Setup Single Sign On \(SSO\) in a cluster environment](#)

[Appendix IV - Generate machine key](#)

Introduction

This manual describes how to create a highly scalable Checkmarx Application cluster. The Checkmarx system HA can be based on MS NLB (Setting up and configuration can be found in [Appendix I](#)) or any other software/hardware NLB (MS/software NLB's are mostly used at test/stage env. while HW NLB is recommended for production env.) By using a NLB, application traffic can be load-balanced, or distributed, across multiple servers. The main purpose of using the NLB is its ability to enhance application availability by ensuring there is no single point of failure. For example, if one of the servers running the Checkmarx Application has a hardware failure or loss of network connectivity, the remaining servers running the Checkmarx Application could continue processing requests; minimizing application downtime. This manual doesn't describe how to create MS SQL and Network Storage Servers high availability. Customers will be responsible for high availability of MS SQL and Network Storage Servers.

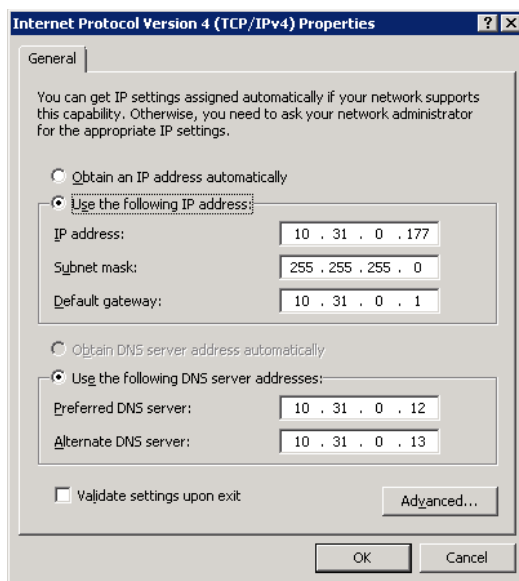
Checkmarx HA Cluster Architecture



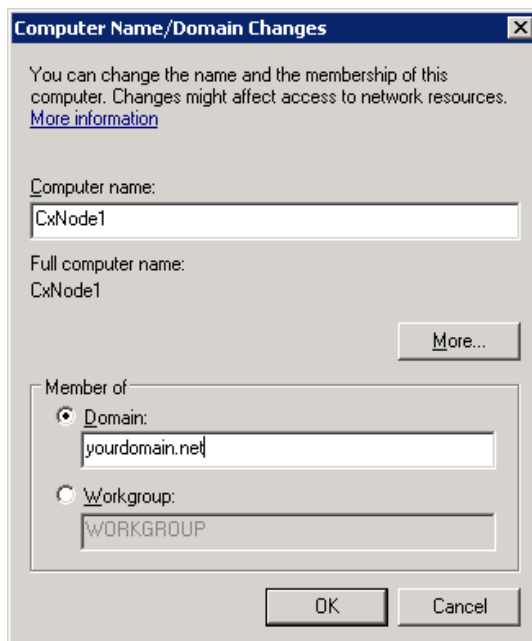
- The NLB server/device communicates with nodes inside the cluster and provides virtual IP to access the Checkmarx Application in the LAN.
- Checkmarx recommends to work with HW NLB (at production env.) in Stateless mode (No sticky session) and ASP.NET Session State service (installed on a machine that is not one of the Cx Nodes) in order to keep the session alive at all times. Working with NLB that is configured for sticky session is supported as well (with no need for ASP.NET Session State service) but that will cause a loss of session if one of the servers will be unavailable and users will have to re-initiate the session. If sticky session will be used the configuration at the LB will have to be for – “Application Generated Cookie Stickiness” with Cookie name = “ASP.NET_SessionId”
- ASP.NET Session State Service will be installed on Windows machine to share session data between nodes inside the cluster. The ASP.NET service may be installed on the NLB server (If MS NLB is used) or a 3rd server / SQL server / one of the NLB nodes.
- The CxNode contains the Checkmarx Application Instance. It includes the CxWebClient, CxWebInterface, CxJobsManager, CxScansManager, CxSystemManager components.
- The CxEngine is a dedicated engine server installed outside a cluster.
- The MS SQL Server is running the CxDB and CxActivity databases outside a cluster.
- The Network Storage is outside a cluster and is used to store files data.

Prerequisites

- All the nodes that are participating in the cluster must have same platform x64 and must run same OS version, at minimum Windows 2008 Standard Edition.
- Communication - End point <-> Load Balancer <-> Cx Manager servers **must** be done with same protocol and port.
- All nodes of cluster should be connected to the same network subnet and use static IP.



- All servers should be connected to network domains with unique DNS.



- All nodes of cluster should have read / write access to network storage.
- The Network storage can be mapped to a drive on each node (SAN) / UNC Paths are supported as well (NAS).
- All nodes of the cluster and the SQL server should have synchronized date and time.
- All nodes and engines must run the same Checkmarx version and hotfix
- Checkmarx system on all nodes of the cluster should be installed at same path on the servers
- All nodes must have same Checkmarx License (LOC, Supported Languages etc.).
- All nodes of the cluster should have access to Checkmarx databases running on MS SQL Server. If windows authentication is used, all nodes should run Checkmarx components under the same windows account (win services and application pools). In the case that SQL authentication is used, all nodes will use the same SQL user (defined in configuration file).
- ASP.NET service may be installed on the NLB server (If MS NLB is used) or a 3rd server / SQL server

Application Layer Gateway Service	PROVIDES S...		Individual	LOCAL SERVICE
Application Management	Processes i...		Manual	Local System
ASP.NET State Service	Provides s...	Started	Automatic	Network S...
Audio Service	Manages a...	Started	Automatic	Local System
AVC WatchDog	AVC Watch	Started	Automatic	Local System

- During installation Administrator privileges are required.
- If MS NLB will be used - NLB feature should be installed on all servers inside a cluster (CxNode and NLB server) prior to Checkmarx system installation on the nodes – for MS NLB installation instructions please see [Appendix I](#).

Checkmarx HA cluster installation and configuration

- All servers that are part of the cluster should be up and running before starting installation and configuration.
- Install the following Checkmarx Application components – Checkmarx “Web Portal” and “CxManager” on each node of the cluster.

- Configure and Verify Read/Write permissions for the Checkmarx Nodes for working with the Network Storage
- Open the CxDB SQL Database -> CxComponentConfiguration table -> Edit the values for the following keys - SOURCE_PATH (X2), EX_SOURCE_PATH, EngineScanLogsPath, REPORTS_PATH that are configured to CxManager’s local drive to the Network Storage mapped drive/UNC Path – All CxNodes will Read/Write from the same folders.
- Set up a shared location to write the Cx logs into.

Setup Checkmarx log file locations to shared storage -

Open each log4net configuration files in the installation:

JobsManager – <Checkmarx Installation Path>\Checkmarx Jobs Manager\bin\CxJobsManagerLog4Net.config

ScansManager – <Checkmarx Installation Path>\Checkmarx Scans Manager\bin\CxScansManagerLog4Net.config

SystemManager –<Checkmarx Installation Path>\Checkmarx System Manager\bin\ CxSystemManagerLog4Net.config

WebServices – <Checkmarx Installation Path>\Checkmarx Web Services\CxWebInterface\log4Net.config

RestAPI - <Checkmarx Installation Path>\Checkmarx Web RestAPI\CxRestAPI\Log4Net.config

WebPortal - <Checkmarx Installation Path>\CheckmarxWebPortal\Web\log4net.config

Open the files for editing and find both file tags. The first one is for the log file, and the second one is for the trace file. This is an example for the WebPortal log configuration.

```
<file value="..\..\Logs\WebClient\${COMPUTERNAME}\Portal.log" />
```

Change both to refer to the storage path you are using:

```
<file
value="//SharedStorage\MyShare\Logs\WebClient\${COMPUTERNAME}\Portal.
log" />
Ex:
\\XXXXXX\CXS_Prod_Data\CXS\Logs\WebClient\${COMPUTERNAME}\Logs\ScansM
anager\Trace\CxScanManagerAll.log
```

Repeat this step for all the following configuration files.

- Setup Checkmarx configurations keys (Only if ASP.NET Session State Service is used):

- Set session settings of CxWebClient to out of process mode – Open web.config file under <Checkmarx installation folder>\CheckmarxWebPortal\Web

- Find -

```
<sessionState mode="InProc" timeout="172800"/>
```

Change it to –

```
<sessionState mode="StateServer" stateConnectionString="tcpip=your-ASP.NET
State-Server:42424" cookieless="false" timeout="172800"/>.
```

Please replace – “your-ASP.NET–State-Server” with the IP address of the Server that holds that service.

- Add an explicit <machineKey> element to the: CxWebClient **web.config** file and CxRestAPI web.client on each node – (The value of <machineKey> on each of the nodes will be the <machineKey> generated on the ASP.NET State service server)

Open **web.config** file under

<Checkmarx installation folder>\CheckmarxWebPortal\Web

And (from version 8.1.0)

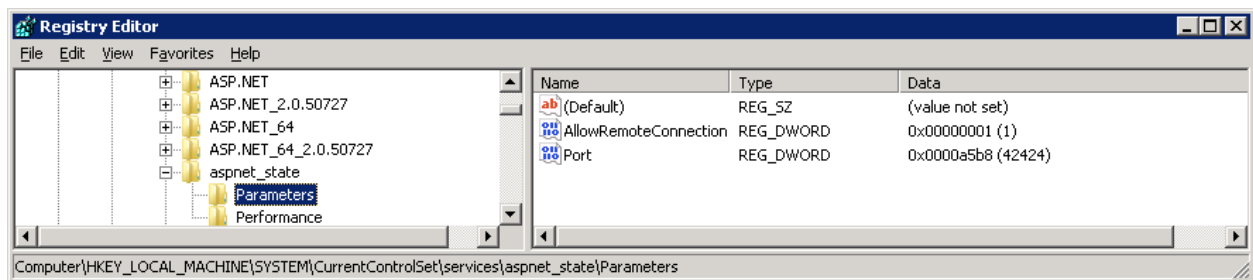
<Checkmarx installation folder>\Checkmarx Web RestAPI\CxRestAPI

on each node and find </system.web>, paste generated code inside the <system.web> section.

See [Appendix IV](#) for instructions - how to generate a <machineKey> element.

Setting up ASP.NET State service - Enabling remote connectivity

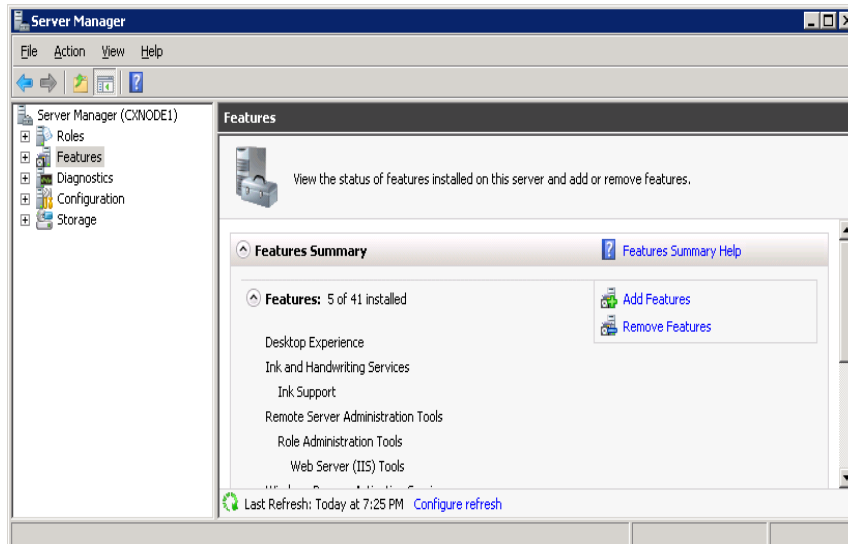
To allow remote connections it is required to set under the registry hive - HKLM\SYSTEM\CurrentControlSet\Services\aspnet_state\Parameters\AllowRemoteConnection the key to a value of '1'. After changing the **AllowRemoteConnection** key value, you'll have to restart ASP.NET state service for the change to take effect. Also make sure your firewall allows connectivity to the ASP.NET state service (TCP 42424 by default).



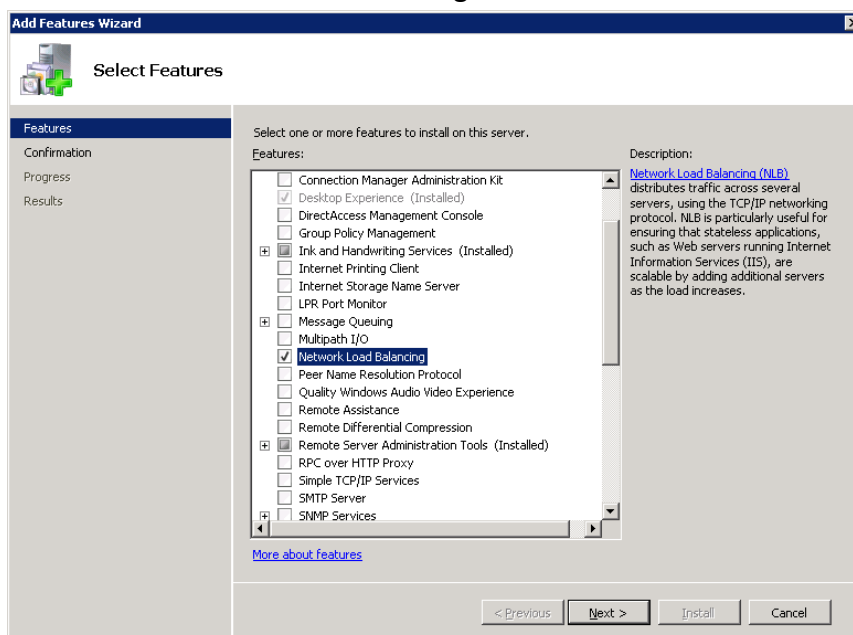
Appendix I

Install and configure MS NLB feature

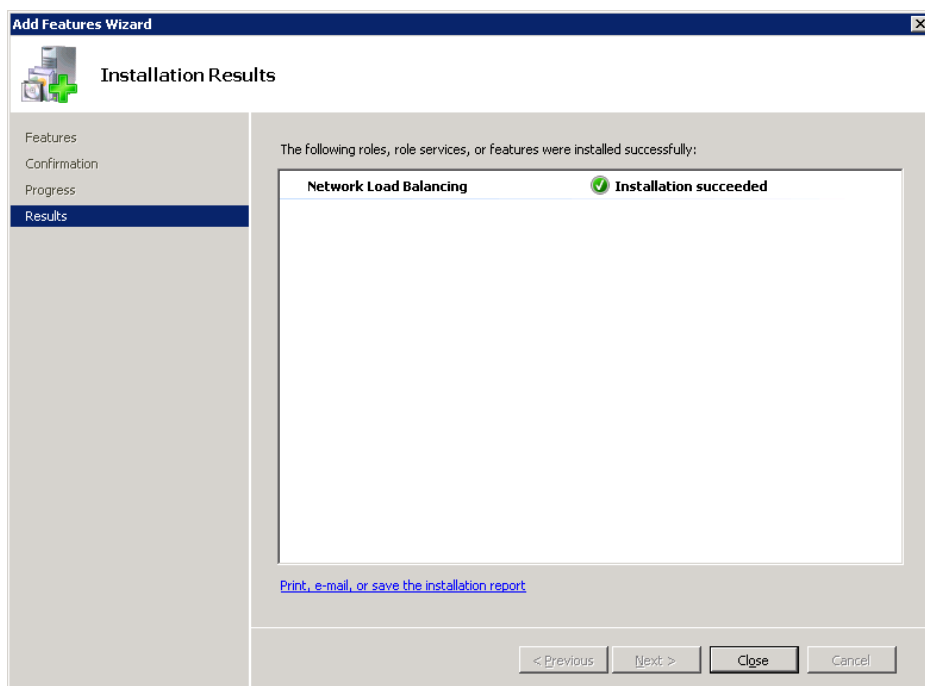
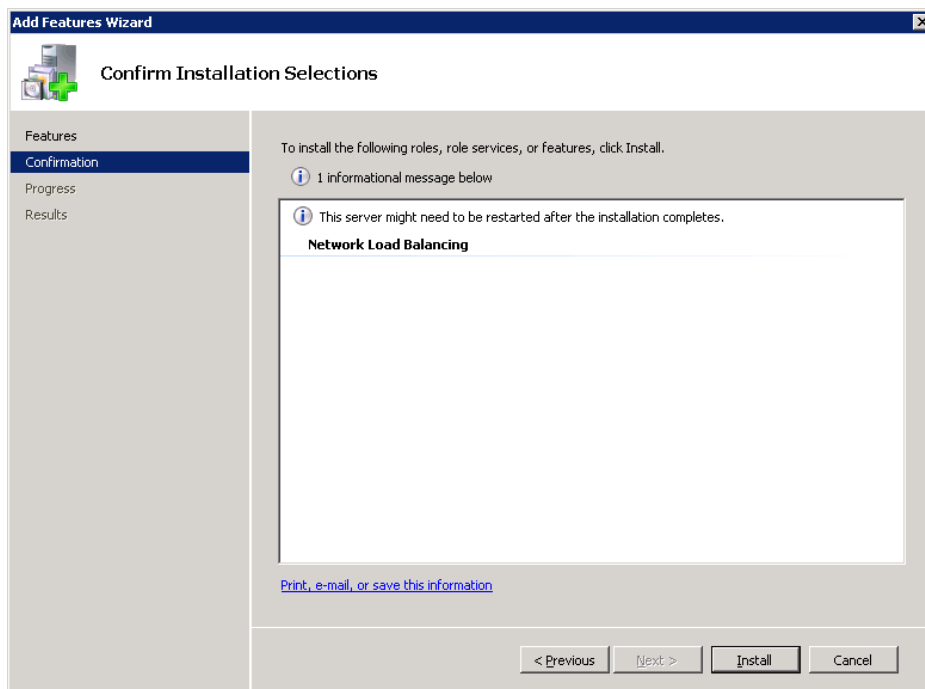
- Open a “Server Manager” and choose “Features”



- Select “Network Load Balancing” and click next.

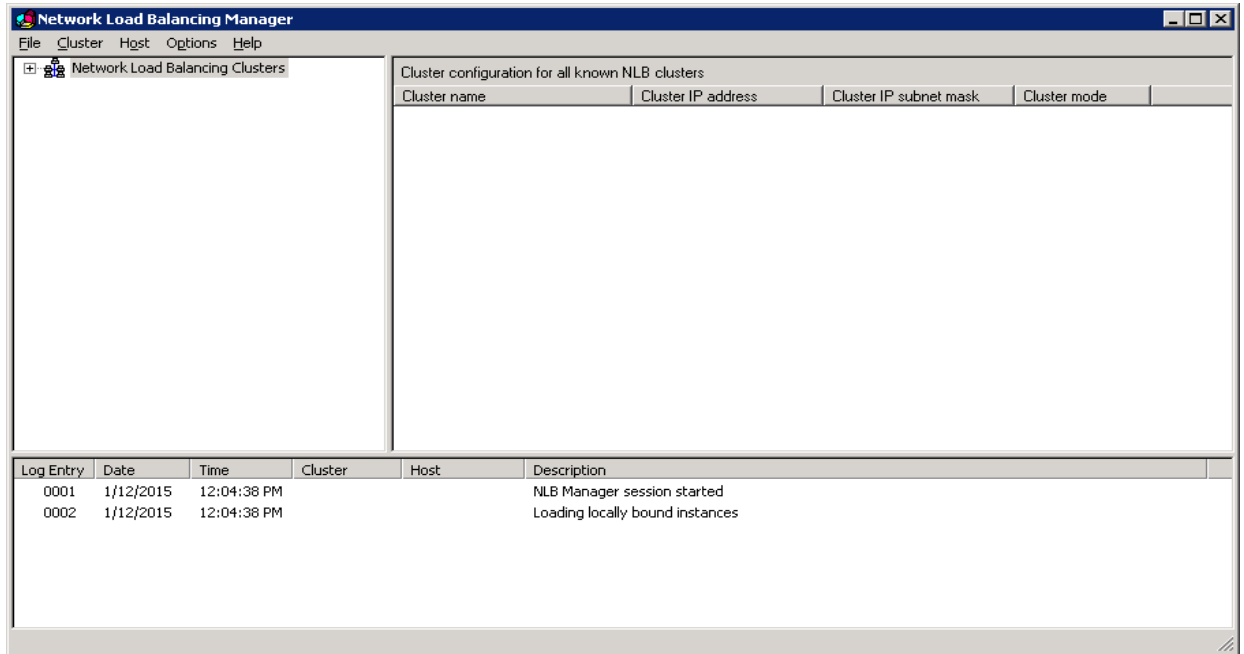


- Click install and wait to installation complete.

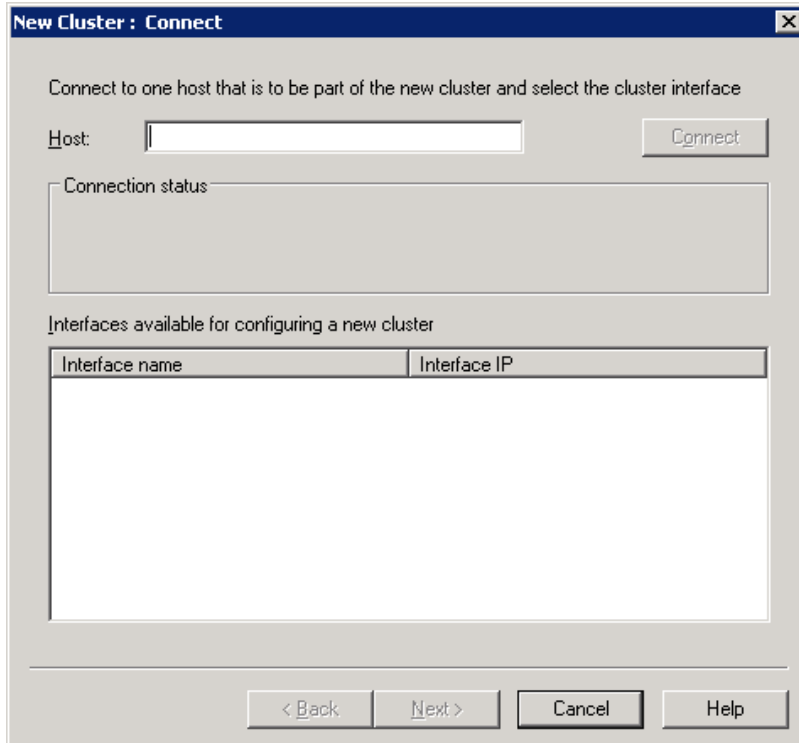


Configuring NLB server

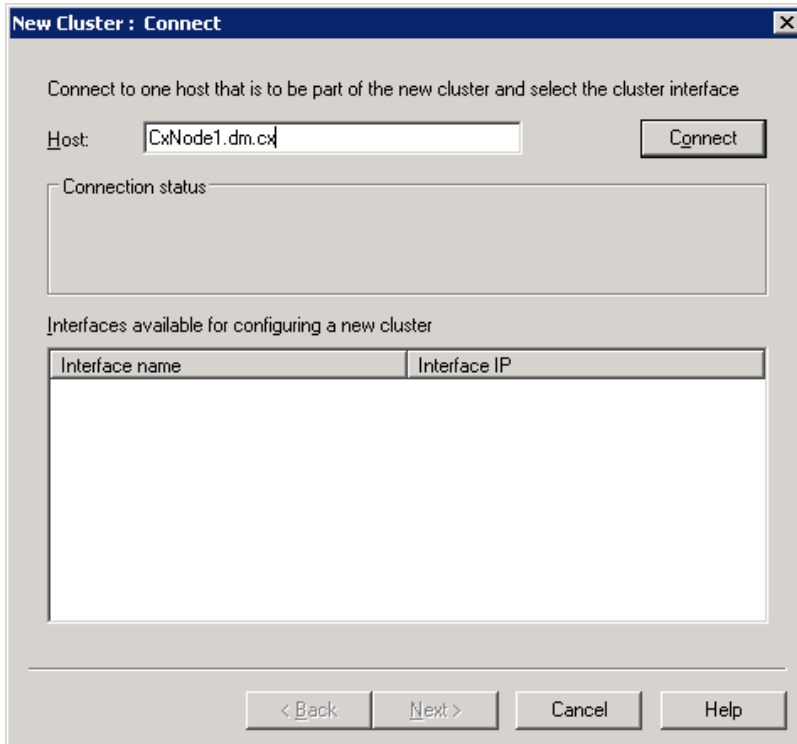
- Connect to NLB server and open "Network Load Balancing Manager". (Administrative Tools -> Network Load Balancing Manager)



- Create new cluster. (Menu Cluster-> New)



- Enter CxNode1 DNS and click “Connect”.



New Cluster : Connect

Connect to one host that is to be part of the new cluster and select the cluster interface

Host:

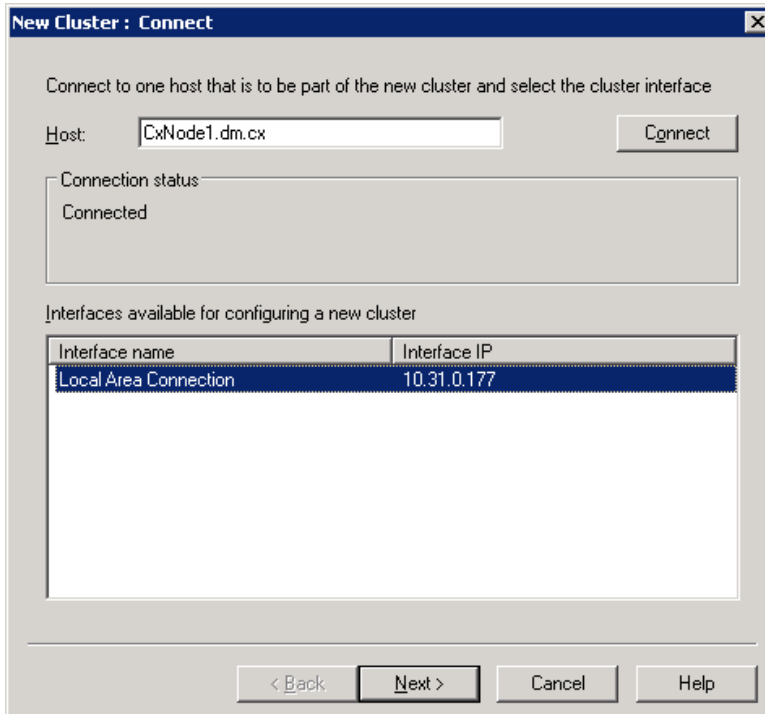
Connection status

Interfaces available for configuring a new cluster

Interface name	Interface IP
----------------	--------------

< Back Next > Cancel Help

- Select the available interface name and click “Next”.



New Cluster : Connect

Connect to one host that is to be part of the new cluster and select the cluster interface

Host:

Connection status

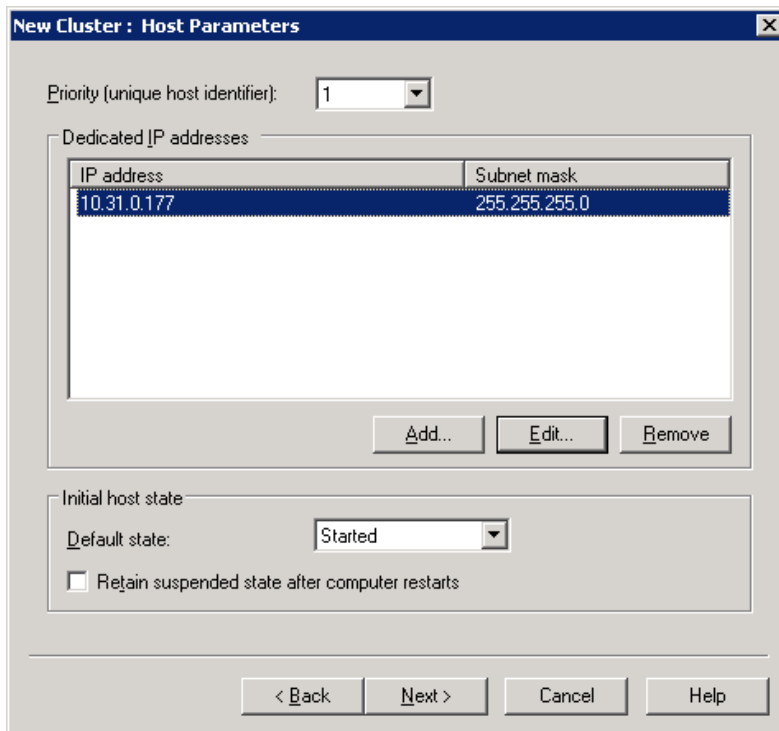
Connected

Interfaces available for configuring a new cluster

Interface name	Interface IP
Local Area Connection	10.31.0.177

< Back Next > Cancel Help

- Verify host parameters and click “Next”.



New Cluster : Host Parameters

Priority (unique host identifier):

Dedicated IP addresses

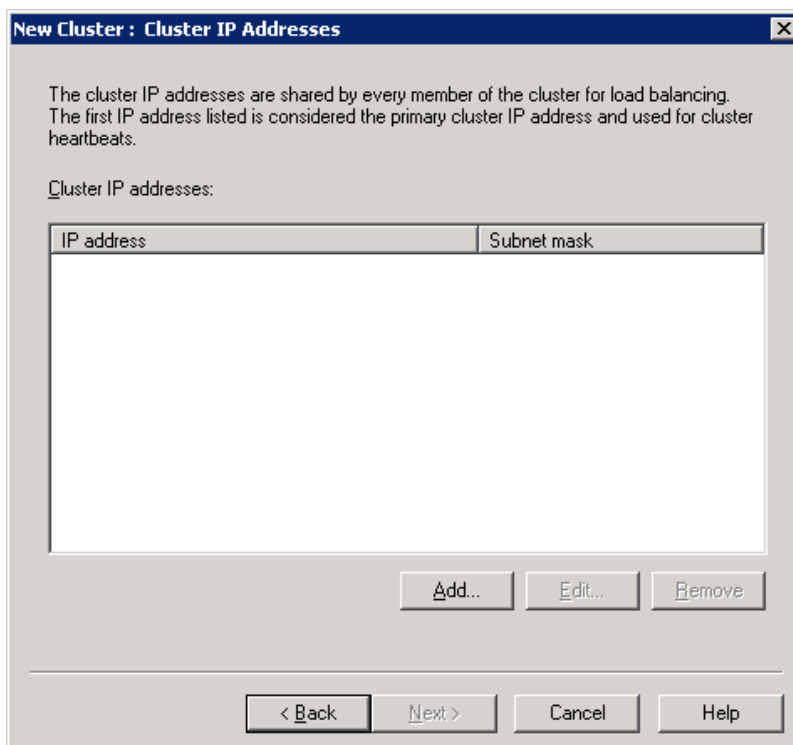
IP address	Subnet mask
10.31.0.177	255.255.255.0

Initial host state

Default state:

☐ Retain suspended state after computer restarts

- Click “Add” to add the cluster IP address.



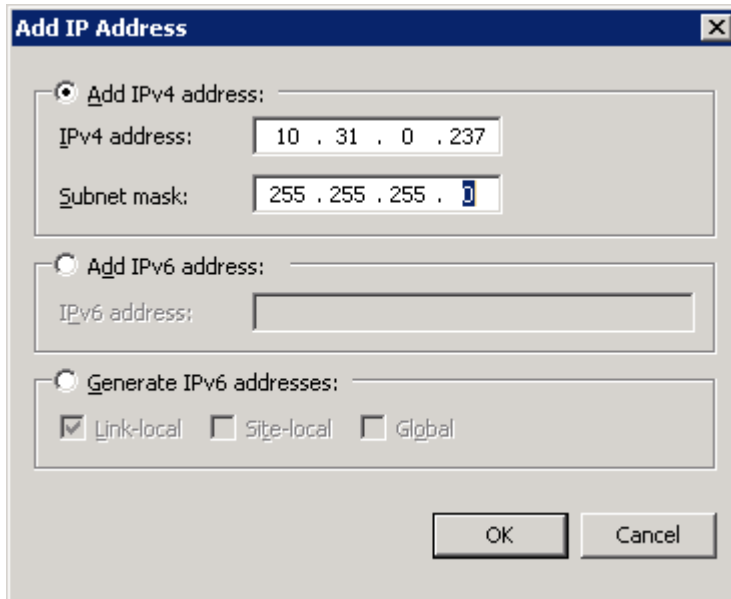
New Cluster : Cluster IP Addresses

The cluster IP addresses are shared by every member of the cluster for load balancing. The first IP address listed is considered the primary cluster IP address and used for cluster heartbeats.

Cluster IP addresses:

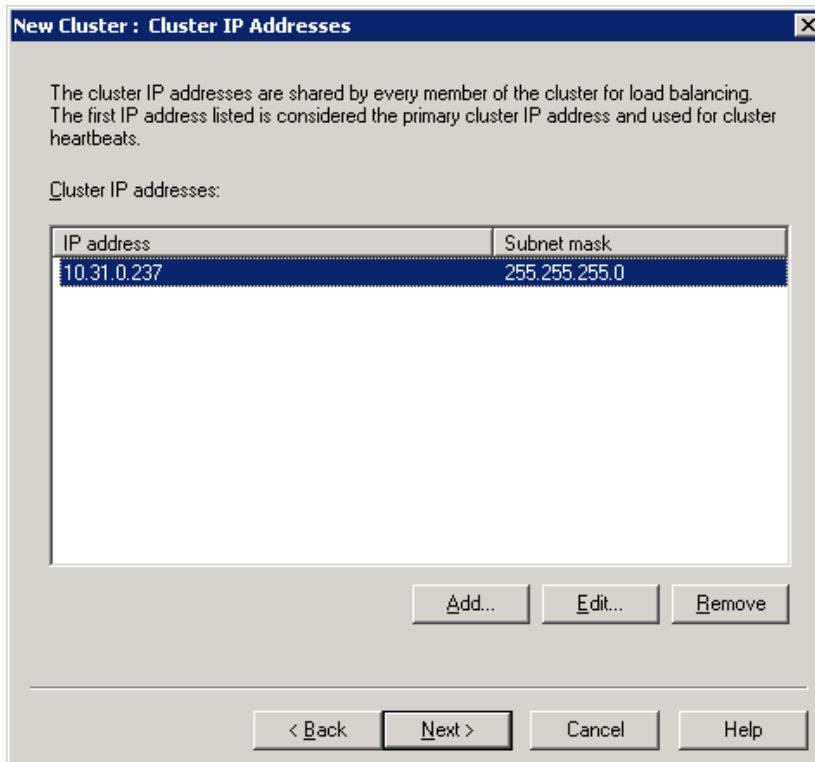
IP address	Subnet mask
------------	-------------

- Enter the Cluster IP and subnet and click “Ok”. Verify that the IP is not assigned to other machine in the network. This IP will be used as a “virtual” IP of the HA cluster and the Checkmarx Application will be available via this IP address.



The "Add IP Address" dialog box has a title bar with a close button. It contains three radio button options: "Add IPv4 address:", "Add IPv6 address:", and "Generate IPv6 addresses:". The "Add IPv4 address:" option is selected. Below it are two text input fields: "IPv4 address:" with the value "10 . 31 . 0 . 237" and "Subnet mask:" with the value "255 . 255 . 255 . 0". The "Add IPv6 address:" option is unselected, and its corresponding "IPv6 address:" field is empty. The "Generate IPv6 addresses:" option is also unselected, and below it are three checkboxes: "Link-local" (checked), "Site-local" (unchecked), and "Global" (unchecked). At the bottom right are "OK" and "Cancel" buttons.

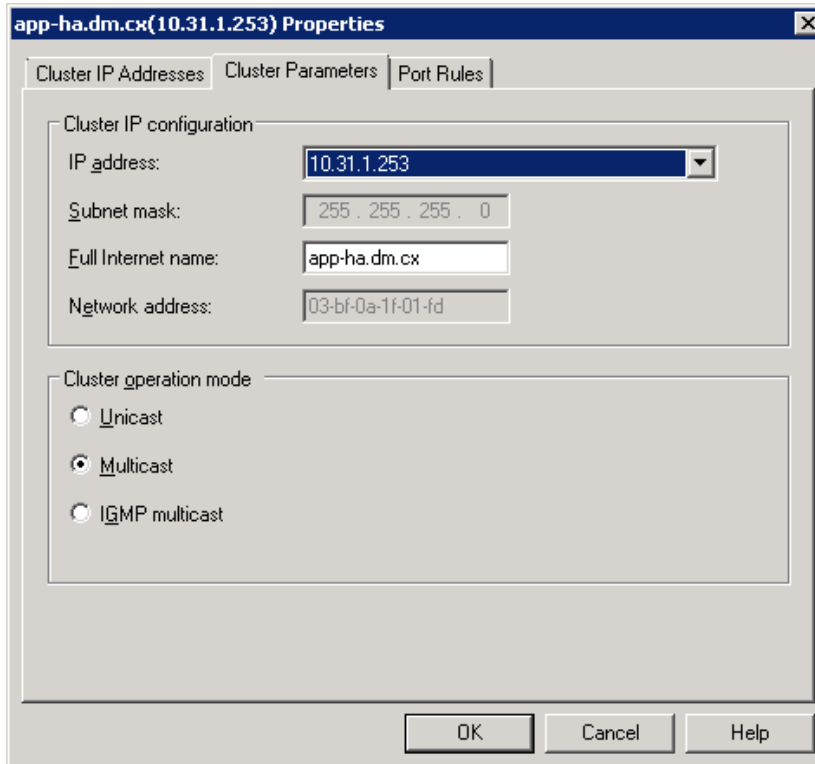
- View the cluster IP and click “Next”



The "New Cluster : Cluster IP Addresses" dialog box has a title bar with a close button. It contains a text area with the following text: "The cluster IP addresses are shared by every member of the cluster for load balancing. The first IP address listed is considered the primary cluster IP address and used for cluster heartbeats." Below this is a label "Cluster IP addresses:" followed by a table. The table has two columns: "IP address" and "Subnet mask". The first row is highlighted in blue and contains the values "10.31.0.237" and "255.255.255.0". Below the table are three buttons: "Add...", "Edit...", and "Remove". At the bottom are four buttons: "< Back", "Next >", "Cancel", and "Help".

IP address	Subnet mask
10.31.0.237	255.255.255.0

- Set the Full Internet name and Multicast mode for the cluster. Click “Next”.
 (The DNS entry should be added manually to domain’s DNS server)



app-ha.dm.cx(10.31.1.253) Properties

Cluster IP Addresses | **Cluster Parameters** | Port Rules

Cluster IP configuration

IP address: 10.31.1.253

Subnet mask: 255.255.255.0

Full Internet name: app-ha.dm.cx

Network address: 03-bf-0a-1f-01-fd

Cluster operation mode

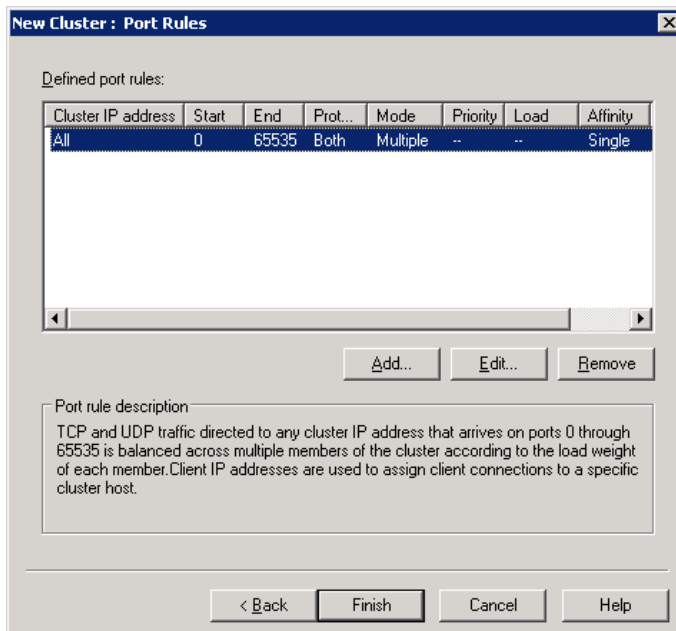
☐ Unicast

☒ **Multicast**

☐ IGMP multicast

OK Cancel Help

- Click “Edit” to set port rules of the cluster.



New Cluster : Port Rules

Defined port rules:

Cluster IP address	Start	End	Prot...	Mode	Priority	Load	Affinity
All	0	65535	Both	Multiple	--	--	Single

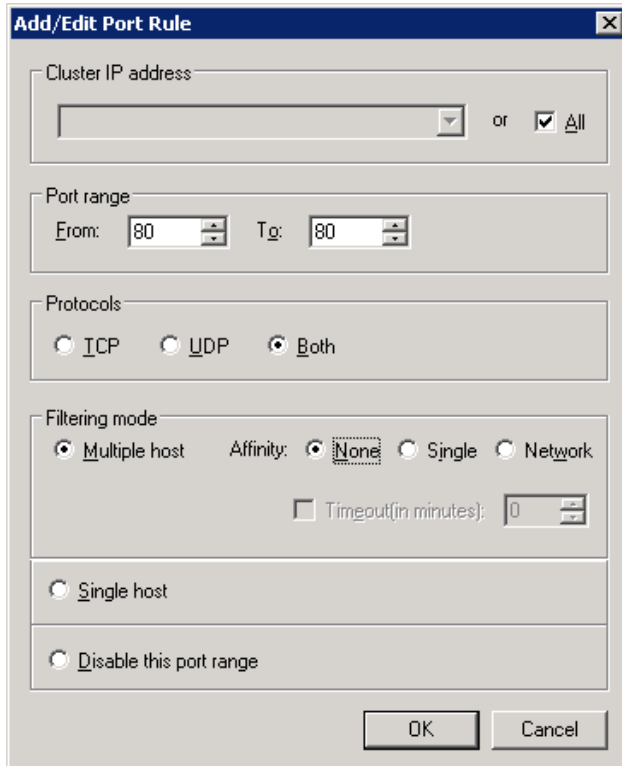
Add... Edit... Remove

Port rule description

TCP and UDP traffic directed to any cluster IP address that arrives on ports 0 through 65535 is balanced across multiple members of the cluster according to the load weight of each member. Client IP addresses are used to assign client connections to a specific cluster host.

< Back Finish Cancel Help

- Edit port range - Enter the port that is configured for IIS listening (binding) on CxNode servers. (80 is default). Set Affinity parameter to “None” and click “Ok.”



Add/Edit Port Rule

Cluster IP address: or ☒ All

Port range: From: To:

Protocols: ☐ TCP ☐ UDP ☒ Both

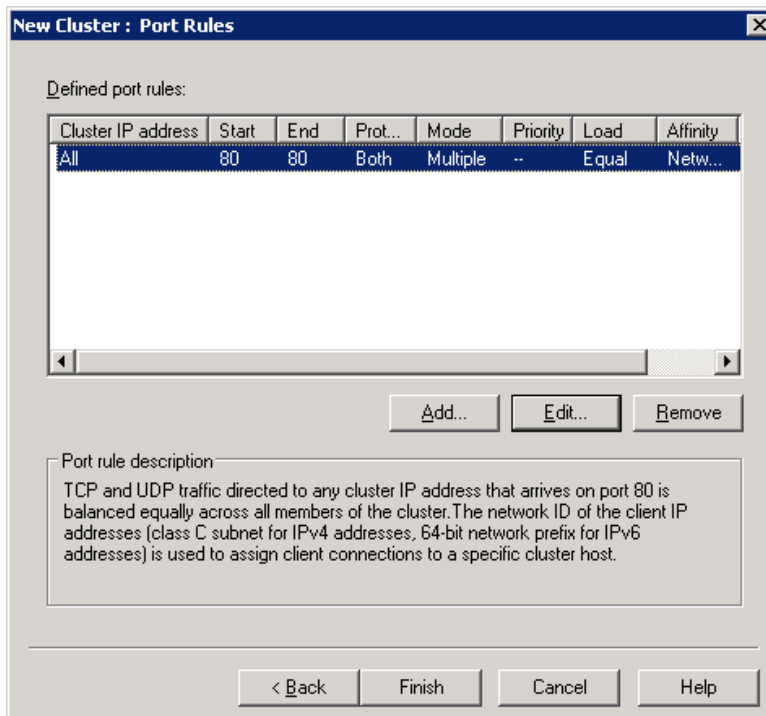
Filtering mode: ☒ Multiple host Affinity: ☒ None ☐ Single ☐ Network
☐ Timeout(in minutes):

☐ Single host

☐ Disable this port range

OK Cancel

- Click “Finish” to create a cluster.



New Cluster : Port Rules

Defined port rules:

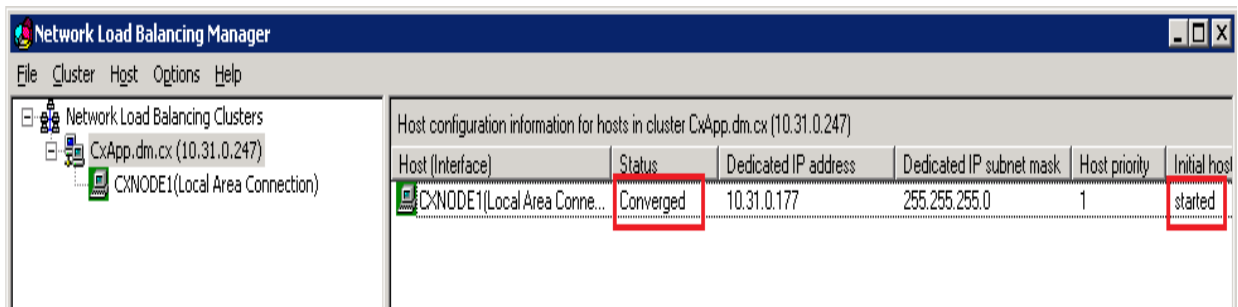
Cluster IP address	Start	End	Prot...	Mode	Priority	Load	Affinity
All	80	80	Both	Multiple	--	Equal	Netw...

Add... Edit... Remove

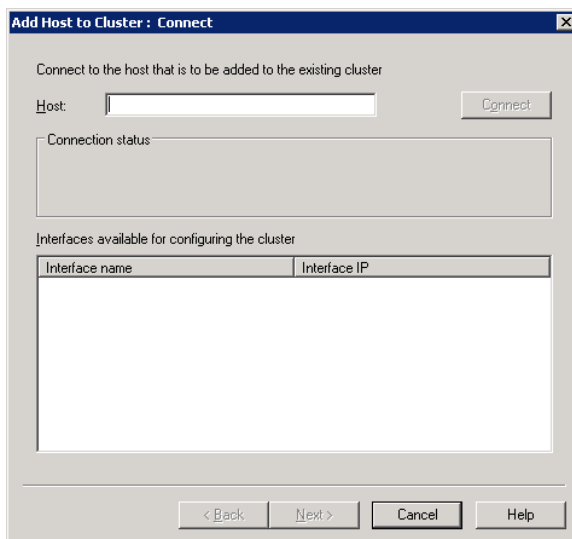
Port rule description:
 TCP and UDP traffic directed to any cluster IP address that arrives on port 80 is balanced equally across all members of the cluster. The network ID of the client IP addresses (class C subnet for IPv4 addresses, 64-bit network prefix for IPv6 addresses) is used to assign client connections to a specific cluster host.

< Back Finish Cancel Help

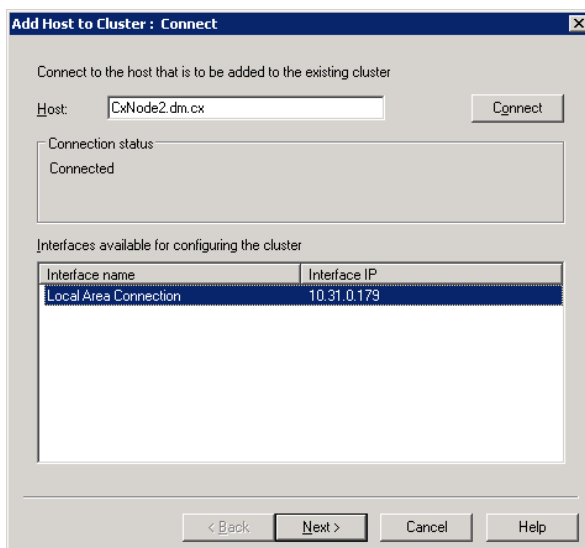
- Cluster created and running.



- Add an additional node to the cluster. Menu -> Cluster -> Add Host



- Enter the DNS of second node and click "Connect".



- Click "Next"

Add Host to Cluster : Host Parameters

Priority (unique host identifier):

Dedicated IP addresses

IP address	Subnet mask
10.31.0.179	255.255.255.0

Initial host state

Default state:

☐ Retain suspended state after computer restarts

- Click "Finish".

Add Host to Cluster : Port Rules

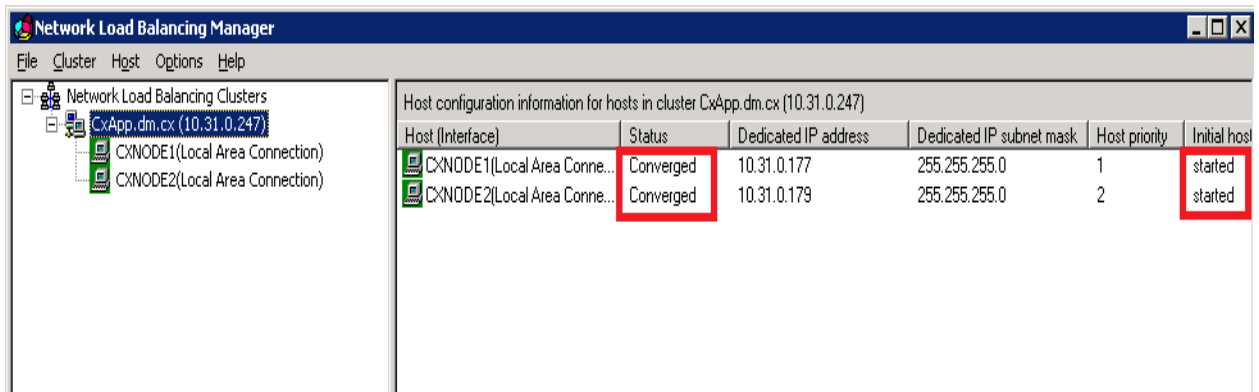
Defined port rules:

Cluster IP address	Start	End	Prot...	Mode	Priority	Load	Affinity
All	80	80	Both	Multiple	--	Equal	Netw...

Port rule description

TCP and UDP traffic directed to any cluster IP address that arrives on port 80 is balanced equally across all members of the cluster. The network ID of the client IP addresses (class C subnet for IPv4 addresses, 64-bit network prefix for IPv6 addresses) is used to assign client connections to a specific cluster host.

- Validate that the new node is added to cluster and is running.



- Browse to Checkmarx portal using cluster IP or DNS <http://yourcluster/cxwebclient/>

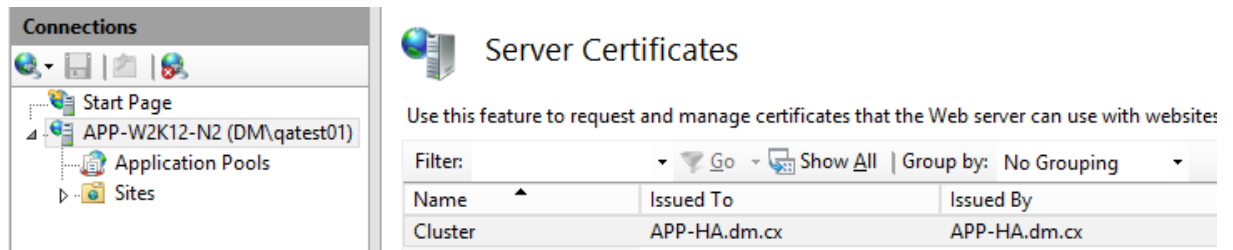
Known limitations

- Change server's DNS name participating in cluster requires manual reconfiguration of cluster and nodes restart.

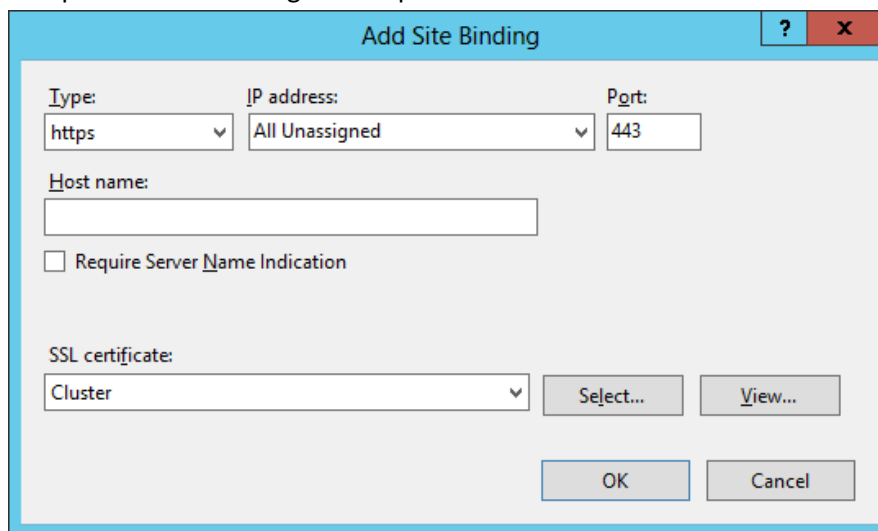
Appendix II

Setup SSL connectivity in a cluster environment

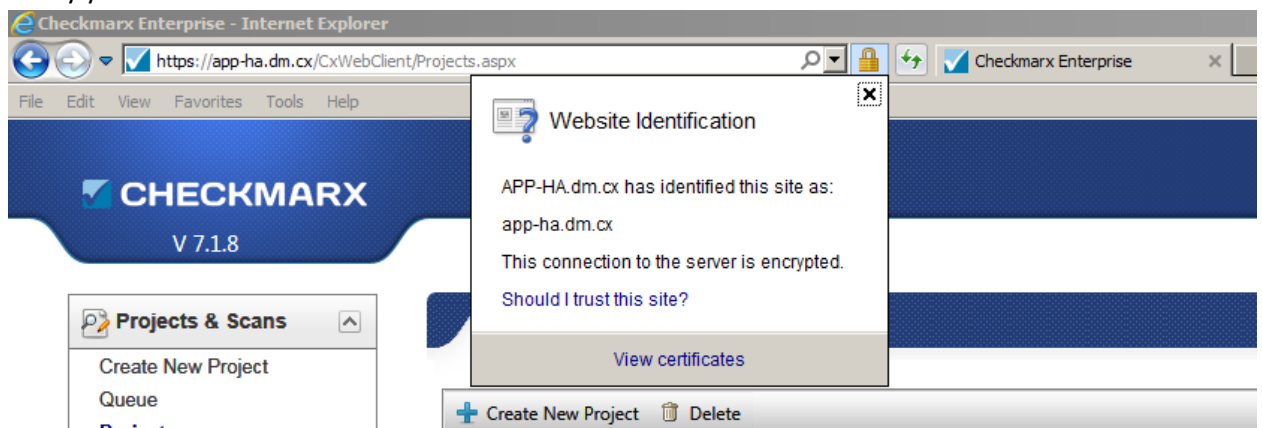
- Create a new certificate for the cluster
- Export the above certificate to a .pfx file
- Import the above certificate on each node (IIS Management -> Server Certificates -> Import)



- Setup the HTTPS binding with imported certificate.



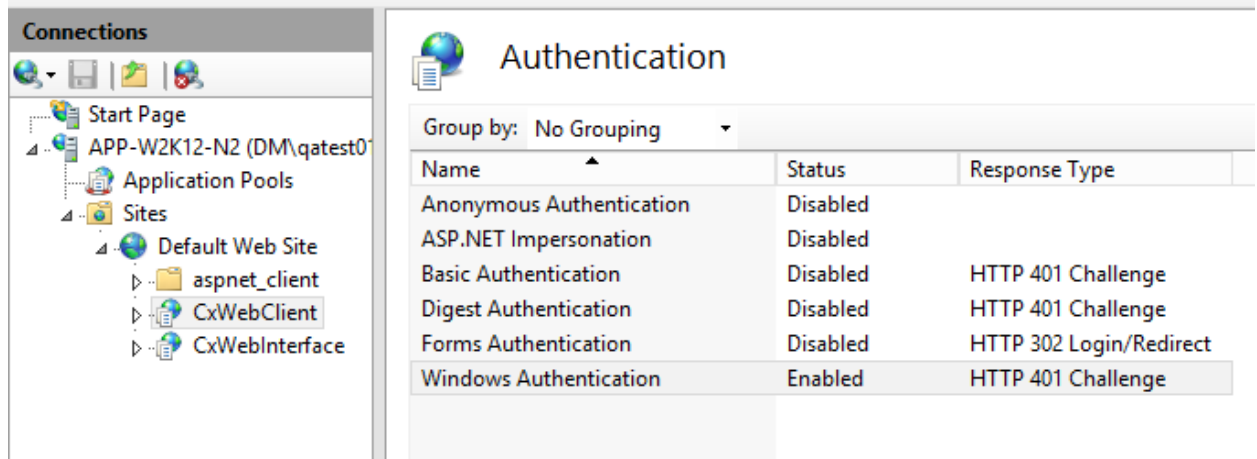
- Verify your clients trust the certificate



Appendix III

Setup Single Sign On (SSO) in a cluster environment

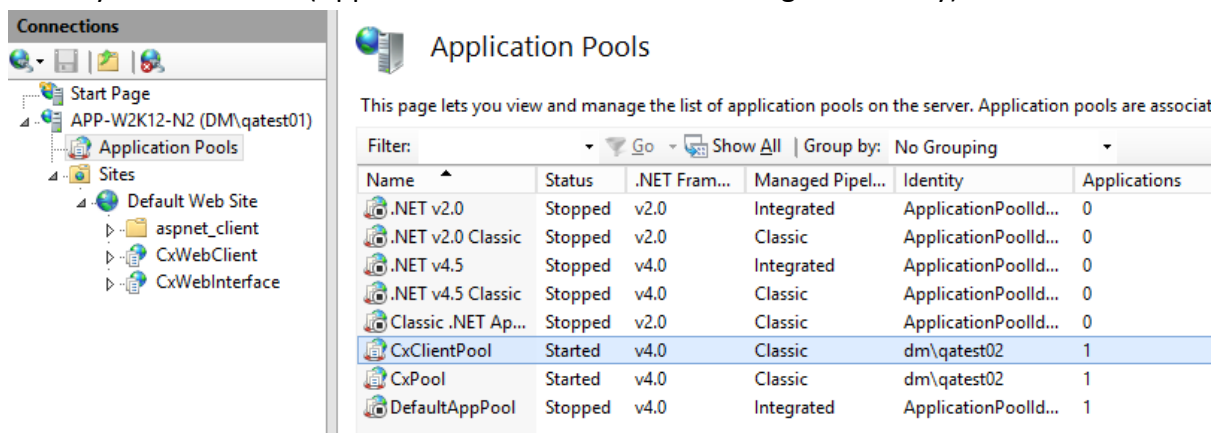
- Enable windows authentication and disable anonymous authentication of CxWebClient and CxWebInterface on each node



The screenshot shows the IIS Manager console with the 'Connections' tree on the left. The 'Authentication' settings for the selected website are displayed on the right. The 'Group by' dropdown is set to 'No Grouping'.

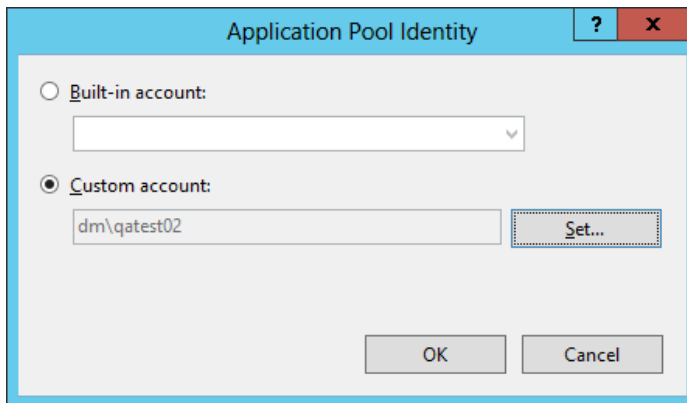
Name	Status	Response Type
Anonymous Authentication	Disabled	
ASP.NET Impersonation	Disabled	
Basic Authentication	Disabled	HTTP 401 Challenge
Digest Authentication	Disabled	HTTP 401 Challenge
Forms Authentication	Disabled	HTTP 302 Login/Redirect
Windows Authentication	Enabled	HTTP 401 Challenge

- Set UseSSOLogin key to true in the WebPortal web.config file –
 <Checkmarx installation folder>\CheckmarxWebPortal\Web.config
- Set the application pools (CxClientPool and CxPool) to run under domain user identity on each node. (Application Pool -> Advanced Settings -> Identity)

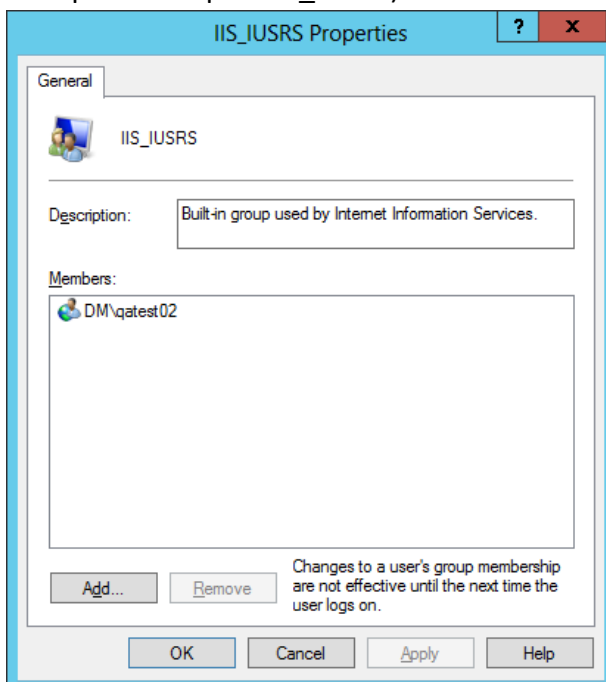


The screenshot shows the IIS Manager console with the 'Connections' tree on the left. The 'Application Pools' settings are displayed on the right. The page title is 'Application Pools' and the description is 'This page lets you view and manage the list of application pools on the server. Application pools are associated with websites and are used to run web applications.' The 'Filter' dropdown is set to 'Show All' and the 'Group by' dropdown is set to 'No Grouping'.

Name	Status	.NET Fram...	Managed Pipel...	Identity	Applications
.NET v2.0	Stopped	v2.0	Integrated	ApplicationPoolId...	0
.NET v2.0 Classic	Stopped	v2.0	Classic	ApplicationPoolId...	0
.NET v4.5	Stopped	v4.0	Integrated	ApplicationPoolId...	0
.NET v4.5 Classic	Stopped	v4.0	Classic	ApplicationPoolId...	0
Classic .NET Ap...	Stopped	v2.0	Classic	ApplicationPoolId...	0
CxClientPool	Started	v4.0	Classic	dm\qatest02	1
CxPool	Started	v4.0	Classic	dm\qatest02	1
DefaultAppPool	Stopped	v4.0	Integrated	ApplicationPoolId...	1



- Add the application pools identity to IIS_IUSRS group on each node. (Local Users and Groups -> Groups-> IIS_IUSRS)



- Edit the %SystemRoot%\System32\inetsrv\config\applicationHost.config file on each node.
 - Find the `<location path="Default Web Site/CxWebInterface">` and edit the security section under `<system.webServer>` as following


```
<security>
  <authentication>
    <anonymousAuthentication enabled="false" />
    <windowsAuthentication enabled="true" useAppPoolCredentials="true"
      useKernelMode="true" />
  </authentication>
</security>
```
 - Find the `<location path="Default Web Site/CxWebClient">` and edit the security section under `<system.webServer>` as following

```
<security>
  <authentication>
    <anonymousAuthentication enabled="false" />
    <windowsAuthentication enabled="true" useAppPoolCredentials="true"
      useKernelMode="true" />
  </authentication>
</security>
```

- Register SPN for the cluster (Domain Administrator permissions are required)

Open command prompt and run the following command

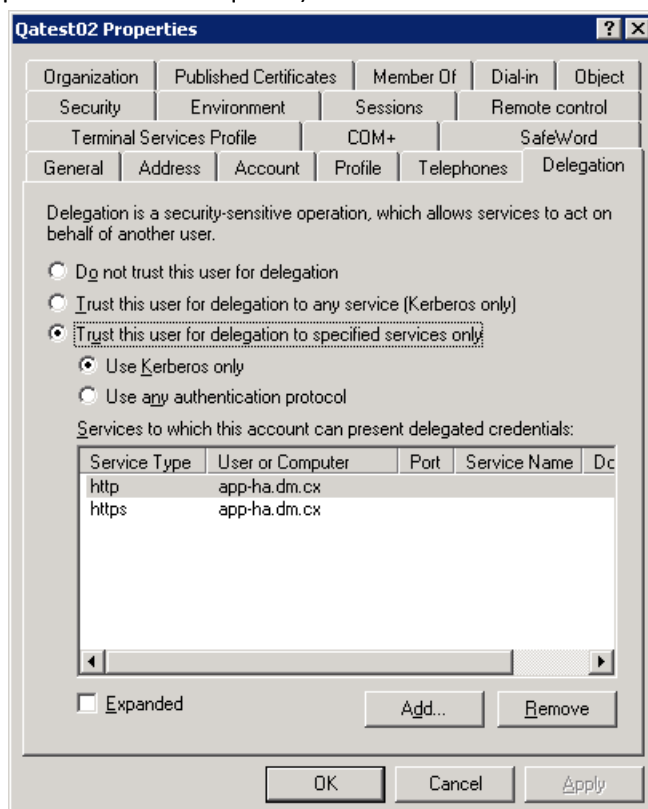
setspn useraccount -a http/your_cluster_full_qualified_domain_name

Note:

- Replace `useraccount` with the account name that is running application pools on the cluster nodes.
- If https is used, it should be registered as following

setspn useraccount -a https/your_cluster_full_qualified_domain_name

- Enable delegation for application pool account to your cluster SPN (Domain Administrator permissions are required)



Appendix IV

• Generate machine key

- Open Windows PowerShell window.
- Copy & Paste the following script

```
# Generates a <machineKey> element that can be copied + pasted into a Web.config file.
function Generate-MachineKey {
    [CmdletBinding()]
    param (
        [ValidateSet("AES", "DES", "3DES")]
        [string]$decryptionAlgorithm = 'AES',
        [ValidateSet("MD5", "SHA1", "HMACSHA256", "HMACSHA384", "HMACSHA512")]
        [string]$validationAlgorithm = 'HMACSHA256'
    )
    process {
        function BinaryToHex {
            [CmdletBinding()]
            param($bytes)
            process {
                $builder = new-object System.Text.StringBuilder
                foreach ($b in $bytes) {
                    $builder = $builder.AppendFormat([System.Globalization.CultureInfo]::InvariantCulture, "{0:X2}", $b)
                }
                $builder
            }
        }
        switch ($decryptionAlgorithm) {
            "AES" { $decryptionObject = new-object System.Security.Cryptography.AesCryptoServiceProvider }
            "DES" { $decryptionObject = new-object System.Security.Cryptography.DESCryptoServiceProvider }
            "3DES" { $decryptionObject = new-object System.Security.Cryptography.TripleDESCryptoServiceProvider }
        }
        $decryptionObject.GenerateKey()
        $decryptionKey = BinaryToHex($decryptionObject.Key)
        switch ($validationAlgorithm) {
            "MD5" { $validationObject = new-object System.Security.Cryptography.HMACMD5 }
            "SHA1" { $validationObject = new-object System.Security.Cryptography.HMACSHA1 }
            "HMACSHA256" { $validationObject = new-object System.Security.Cryptography.HMACSHA256 }
            "HMACSHA385" { $validationObject = new-object System.Security.Cryptography.HMACSHA384 }
            "HMACSHA512" { $validationObject = new-object System.Security.Cryptography.HMACSHA512 }
        }
        $validationKey = BinaryToHex($validationObject.Key)
        [string]::Format([System.Globalization.CultureInfo]::InvariantCulture,
            "<machineKey decryption=\"{0}\" decryptionKey=\"{1}\" validation=\"{2}\" validationKey=\"{3}\" />",
            $decryptionAlgorithm.ToUpperInvariant(), $decryptionKey,
            $validationAlgorithm.ToUpperInvariant(), $validationKey)
    }
}
```

- Run **Generate-MachineKey** without parameters to generate a <machineKey>