PowerShell Desired State Configuration nxNetworking Resources

Version: 1.0  
Documentation Date: September, 25 2015

nxDNSServerAddress

The **nxDNSServerAddress** resource in PowerShell Desired State Configuration (DSC) provides a mechanism to manage DNS Client settings on a Linux node.

### Syntax

|  |
| --- |
| nxDNSServerAddress <string> #ResourceName  {  Address[] = <string> #The DNS server address(es)  [ Ensure = <string> { Absent | Present } ]  [ AddressFamily = <string> { IPv4 | IPv6 } ]  } |

|  |  |
| --- | --- |
| Property | Description |
| Address | Specifies the address(es) of DNS servers that the computer should use for name resolution. |
| Ensure | Specifies whether the DNS server addresses exist in the configuration. Set this property to **Present** to ensure that the address exists, and set it to **Absent** to ensure that the address does not exist. |
| AddressFamily | Specifies whether the defined addresses are **IPv4** or **IPv6** addresses. |

### Example

|  |
| --- |
| Import-DSCResource -Module nxNetworking  Node $node {  nxDNSServerAddress DNSSettings{  Address = "10.25.25.11","10.26.26.11"  Ensure = "Present"  AddressFamily = "IPv4"  }  } |

nxIPAddress

The **nxIPAddress** resource in PowerShell Desired State Configuration (DSC) provides a mechanism to manage IP Address settings on a Linux node.

### Syntax

|  |
| --- |
| nxIPAddress <string> #ResourceName  {  IPAddress = <string>  InterfaceName = <string>  [ BootProtocol = <string> { Automatic | Static } ]  [ DefaultGateway = <string> ]  [ Ensure = <string> { Present | Absent } ]  [ PrefixLength = <string> ]  [ AddressFamily = { IPv4 | IPv6 }]  } |

|  |  |
| --- | --- |
| Property | Description |
| IPAddress | Specifies the address(es) of DNS servers that the computer should use for name resolution. |
| InterfaceName | Specifies the interface name (e.g. eth0) to configure the IP address for |
| BootProtocol | Specifies the boot protocol for the interface. Use **Automatic** for DHCP/autoconf and **Static** to specify the IP address. |
| DefaultGateway | Specifies the Default Gateway address for the interface |
| Ensure | Specifies whether the IP address should exist for the interface or not. Set this property to **Present** to ensure that the address exists, and set it to **Absent** to ensure that the address does not exist. |
| PrefixLength | Specifies the PrefixLength for the IP Address (CIDR notation). |
| AddressFamily | Specifies whether the defined addresses are **IPv4** or **IPv6** addresses. |

### Example

|  |
| --- |
| Import-DSCResource -Module nxNetworking  Node $node {  nxIPAddress Eth1Address{  IPAddress = "192.168.11.201"  InterfaceName = "eth1"  BootProtocol = "static"  DefaultGateway = "192.168.11.1"  Ensure = "Present"  PrefixLength = "24"  AddressFamily = "IPv4"  }  } |

nxFirewall

The **nxFirewall** resource in PowerShell Desired State Configuration (DSC) provides a mechanism to manage firewall rule settings on a Linux node.

### Syntax

|  |
| --- |
| nxFirewall <string> #ResourceName  {  Name = <string>  [ InterfaceName = <string> ]  FirewallType = <string> { IpTables | Ip6tables | yast | ufw | susefirewall2 | firewalld }  [ Protocol = <string> { tcp | udp | icmp } ]  [ Ensure = <string> { Present | Absent } ]  [ AddressFamily = <string> { IPv4 | IPv6 } ]  [ Access = <string> { Allow | Block }]  [ State = <string> { ESTABLISHED | RELATED | NEW } ]  [ Direction = <string> { INPUT | OUTPUT | FORWARD } ]  [ SourceHost = <string> ]  [ SourcePort = <string> ]  [ DestinationHost = <string> ]  [ DestinationPort = <string> ]  [ Position = <string> { top | after-top | before-end | end }]  } |

|  |  |
| --- | --- |
| Property | Description |
| Name | Specifies a name for the firewall rule |
| InterfaceName | Specifies the name of the interface the rule applies to. |
| FirewallType | Specifies the type of firewall on the Linux node to configure. Choices are: **IpTables**, **Ip6tables**, **yast**, **ufw**, **susefirewall2**, or **firewalld.** |
| Protocol | Specifies the protocol that the rule applies to. Choices are: **tcp**, **udp**, or **icmp.** |
| Ensure | Specifies whether the firewall rule should exist or not. Set this property to **Present** to ensure that the ruleexists, and set it to **Absent** to ensure that the rule does not exist. |
| AddressFamily | Specifies the address family (**IPv4** or **IPv6**) that the rule applies to |
| Access | Specifies whether to **Allow** or **Block** connections matching the rule parameters. |
| State | Specifies the state that the rule applies to. Choices are: **ESTABLISHED**, **RELATED**, or **NEW.** |
| Direction | Specifies the connection direction that the rule applies to. Choices are **INPUT, OUTPUT,** or **FORWARD.** |
| SourceHost | Specifies the source host that the rule applies to. |
| SourcePort | Specifies the source port that the rule applies to. |
| DestinationHost | Specifies the destination port that the rule applies to. |
| Position | Specifies the relative location in the firewall rule list to position the rule. Choices are **top**, **after-top**, **before-end**, or **end.** |

### Example

|  |
| --- |
| Import-DSCResource -Module nxNetworking  Node $Target.NodeName {  nxIPAddress Eth1{  InterfaceName = "eth1"  IPAddress = "192.168.12.101"  PrefixLength = "24"  BootProtocol = "static"  }  nxService FWD{  Name = "firewalld"  Enabled = $true  State = "Running"  Controller = "systemd"  }  nxFirewall WsManIn {    Name = "Allow wsman in"  interfaceName = "eth0"  FirewallType = "firewalld"  Ensure = "Present"  Access = "Allow"  Direction = "Input"  DestinationPort = "5986"  Position = "Before-End"    }    nxFirewall FWConfig {    Name = "Allow http in"  interfaceName = "eth0"  FirewallType = "firewalld"  Ensure = "Present"  Access = "Allow"  Direction = "Input"  DestinationPort = "80"  Position = "Before-End"    }    } |