# Configuring High Availability on the Cisco CSR 1000v

High Availability refers to the ability to establish redundancy of networking functionality and configuration data between two peer routers.



# Information About High Availability on the Cisco CSR 1000v



## Overview of High Availability for the CSR 1000v on Microsoft Azure

You can deploy the Cisco CSR 1000v between the front-end and back-end subnets. The Cisco CSR 1000v represents a single point of failure for access to back-end resources. For example, you can configure two Cisco CSR 1000vs and connect them in parallel between the front and back end subnets. Each of these Cisco CSR 1000vs is known a peer router.

The routing table for the back-end subnet contains entries that point to the next hop router, which is one of the two Cisco CSR 1000vs. The routing protocol that is configured on the Cisco CSR 1000v determines the path of the downstream traffic.

The peer Cisco CSR 1000v routers communicate with one another over a tunnel using the Bi-directional Forwarding Detection (BFD) protocol. The BFD generates an event if connectivity between the peer routers is lost. This event causes the active Cisco CSR 1000v to update the entries in the route table to point to itself

as the default router. The routing table controls the upstream traffic of the Cisco CSR 1000v. An active router is the next-hop router for either an individual route or all routes in the user-defined route table.

In cloud environments, it is common for virtual networks to implement a simplistic mechanism for routing, which is based on a centralized route table. You can create multiple route tables. Each route table has a subnet that is assigned, which is a source of route information. In Microsoft Azure, the route table is populated automatically and includes one or more individual routes depending upon the network topology. You can configure the routes in the route table. You can also configure using the GUI on the Microsoft Azure portal web site, entering Azure CLI commands, or by using the programmatic APIs (for example, a REST API).

Using a centralized route table for a subnet allows a pair of Cisco CSR 1000v routers to operate in a redundant fashion. You can deploy two Cisco CSR 1000vs can be deployed in the same virtual network and have interfaces that are directly connected to subnets within the virtual network. You can add routes to the route table to point to one of the two redundant CSR 1000vs. So at any given time, one of the two CSRs is the next hop router for a subnet. This router is called the active router for the subnet and the other (peer) router is the passive router.

A subnet has a centralized route table, which allows two Cisco CSR 1000v routers to operate in a redundant mode. You can deploy two Cisco CSR 1000vs in the same virtual network with their interfaces that are directly connected to subnets in the virtual network. You can add routes to the route table to point to one of the two redundant CSR 1000vs. At any given time, one of the two Cisco CSR 1000v's serves as the next hop router for a subnet. This router is called the active router for the subnet. The peer router is referred to as the passive router. The “active” Cisco CSR 1000v is the next hop for a given route destination.

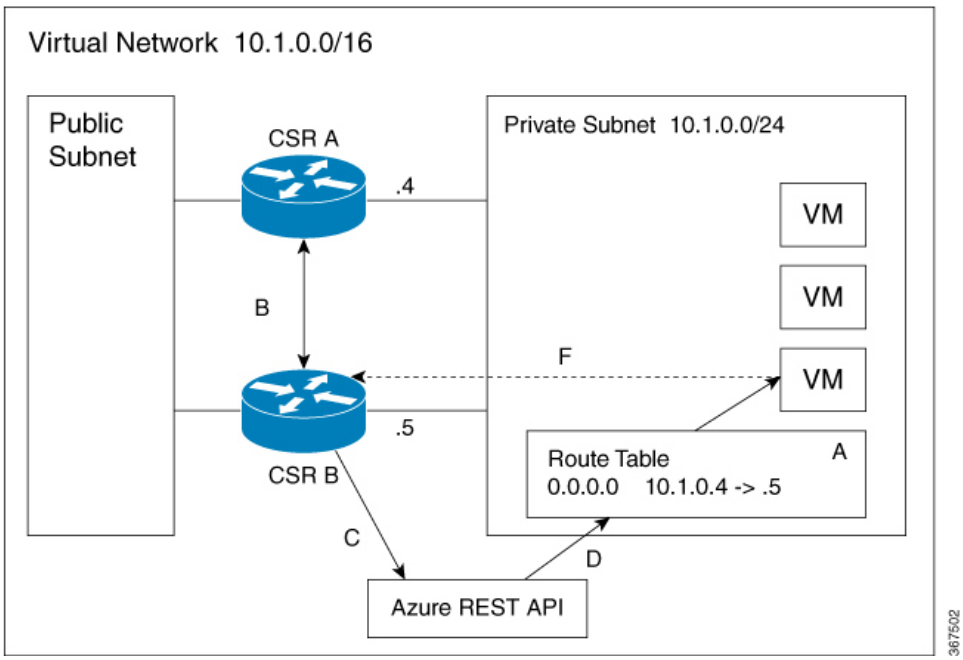
Failure detection is a mechanism that is used by a Cisco CSR 1000v to detect whether its peer Cisco CSR 1000v is operating properly. The Bidirectional Failure Detection (BFD) protocol is used. An IP tunnel is created between the two peer routers. Each router periodically sends a BFD protocol message to the other router. If one router fails to receive a BFD message from its peer for some period, it concludes the peer router has failed.

If the active router fails, the route table for the subnet can be dynamically updated to change the next hop address for one or more routes so that they refer to the passive router. If the peer router detects the failure of the active router, the peer Cisco CSR 1000v router uses the programmatic API to update the route table entries.

**Note** The Microsoft Azure route table updates may take up to one minute to take effect and may cause a delay in the High Availability failover.

For a route table entry, configure which of the two Cisco CSR 1000v routers is the primary router. The other router is the passive router if it is configured as a secondary router. By default, all routes are configured as secondary.

Consider the simple network diagram that is shown in the following figure. The topology in this figure is an example of 1-for-1 redundancy. For more information, see the Redundancy Topologies, on page 3 section.




The Private Subnet has an address block of 10.1.0.0/24. CSR A and CSR B provide a redundant path for traffic leaving this leaf subnet. The subnet has a route table that provides the route information to virtual machines which are attached to the Private Subnet. Initially the default route in the route table records the IP address of the next hop router as 10.1.0.4 (CSR A). All traffic leaving the subnet goes through CSR A. CSR A is currently the active router for the default route. Then, CSR A fails and CSR B detects the failure because it stops receiving BFD protocol messages from CSR A. CSR B writes to the route table via a REST API to change the default route to the interface of CSR B on the 10.1.0.0/24 subnet, which is IP address 10.1.0.5. CSR B becomes the active router for the default route.

| Step | Description |
|---|---|
| A | CSR A with address 10.1.0.4 is the active router for the 15.0.0.0 network. |
| B | CSR A fails. CSR B detects the failure using the BFD protocol. |
| C | CSR B uses an HTTP request to the Azure REST API. |
| D | Azure updates the 15.0.0.0 route in the user-defined route table to the IP address of CSR B. |
| E | Virtual machines see the route table update. |
| F | Packets from the virtual machines are now directed to CSR B. |

# Redundancy Topologies

Two different redundancy topologies are supported:

**1-for-1 redundancy topology** If both of the Cisco CSR 1000v routers have a direct connection to the same subnet, the routers provide 1-for-1 redundancy. All the traffic that is intended for a Cisco CSR 1000v only goes to the active router. The active router is the next-hop router for a subnet. The other router is the passive router for all the routes.

**Load sharing topology** In this topology, both the Cisco CSR 1000v routers have direct connections to different subnets within the same virtual network. Traffic from subnet A goes to router A and traffic from subnet B goes to router B. Each of these subnets is bound to different route tables. If router A fails, the route table for subnet A is updated. Instead of router A being the next hop, the route entry is changed to router B as the next hop. If router B fails, the route table for subnet B is updated. In the same manner if router B fails, the route table now includes router A as the next hop router.

# Redundancy Nodes

A redundancy node is a set of configuration parameters that specifies an entry in a route table. The next hop of a route is updated when an active router fails. Configuring a redundancy node requires the following information:

***Table 1: Redundancy Node Parameters***

| Parameter | Description |
|---|---|
| Route Table | The route may include the following details:<br>• Name or identifier for the table<br>• The region or group in which the table was created<br>• Identifier for the creator/owner of the table<br>• (Optional) Individual Route. If an individual route is not specified, the redundancy node represents all of the routes in the route table. |
| Credentials | Authentication for the Cisco CSR 1000v, which gives it the authorization to update entries in the route table. Each cloud provider may handle the process of obtaining and specifying the credentials differently. |
| Next Hop | Next hop address that is written to the routing table entry when a trigger event occurs. It is usually the next hop address of the interface for the Cisco CSR 1000v, on the subnet that is being protected. |
| Peer Router | Identifies the redundant router that forwards traffic for this route, after a failure occurs on this router. |
| Router Role | Identifies whether the redundancy node serves in a primary or secondary role. This is an optional parameter. If not specified, the router role defaults to a secondary role. |

# Event Types

The high availability feature recognizes and responds to three types of events:

**Peer Router Failure**

When the peer route fails, it is detected as a Peer Router Failure event. In response to this event, the event handler writes the route entry with the next hop address that is defined in the redundancy node. To enable this event to be generated, configure the BFD protocol to a peer router and associate the BFD peer with one or more redundancy nodes.

**Revert to Primary Router**

After a router recovers from a failure, the Revert to Primary Router event occurs. The purpose of the event is to ensure that in the route table entry for the redundancy node has this router defined as the primary router.

**Redundancy Node Verification**

Use the Redundancy Node verification event to verify the ability of the event handler to execute functions. The event handler detects a Redundancy Node verification event and reads the route entry that is specified by the redundancy node. The event handler writes the same data back to the route entry. The Redundancy Node verification event is triggered by executing a script (manually or programatically). For further information about verification events, see User-Defined Triggers, on page 32 in the Advanced Programming for High Availability on Microsoft Azure, on page 32 section.

# New and Updated Information About High Availability

The first version of high availability in the Azure cloud—HA Version 1—was introduced in Cisco IOS XE Everest 16.5.1.

**Note** The second version of high availability --HA Version 2-- is available using Cisco IOS XE Fuji 16.9.2 or later. HA Version 2 supports several new features, and a new configuration and deployment mechanism. It is recommended that you migrate to HA version 2, as support for HA version 1 will be removed from a future IOS release.

The following functions are introduced in HA Version 2:

### Active–Active Operation

You can configure both Cisco CSR 1000vs to be active simultaneously, which allows for load sharing. In this mode of operation, each route in a route table has one of the two routers serve as the primary router and the other router serves as a secondary router. To enable load sharing, take all the routes and split them between the two Cisco CSR 1000vs.

### Reversion to Primary CSR After Fault Recovery

You can designate a Cisco CSR 1000v as the primary router for a given route. While this Cisco CSR 1000v is up and running, it is the next hop for the route. If the Cisco CSR 1000v fails, the peer Cisco CSR 1000v takes over as the next hop for the route, maintaining network connectivity. When the original router recovers from the failure, it reclaims ownership of the route and is the next hop router.

### New Configuration and Deployment Mechanism

In HA version 2, the implementation of HA has been moved out of the Cisco IOS XE code and runs in the guestshell container. For further information on the guestshell, see the "Guest Shell" section in the Programmability Configuration Guide. In HA version 2, the configuration of redundancy nodes is performed in the guestshell using a set of Python scripts.

### Authentication by Microsoft Managed Service Identity

The Azure Active Directory(AAD) must authenticate a Cisco CSR 1000v to access and update route tables in the Azure network. (In HA version 1, authentication is achieved by creating a Cisco CSR 1000v application in Azure Active Directory (AAD) to represent the Cisco CSR 1000v.)

Microsoft has a Managed Service Identity (MSI) service that automates the creation of an application for a virtual machine. For more information on MSI, see: https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview. HA version 2 uses the MSI service to authenticate the Cisco CSR 1000v. You do not need to manually create the Cisco CSR 1000v application.

**Note** HA version 2 also continues to support authentication in Azure Active Directory (AAD).

### User-Supplied Scripts

The guestshell is a container in which you can deploy your own scripts. HA Version 2 exposes a programming interface to user-supplied scripts, so you can write scripts that can trigger both failover and reversion events. You can develop your own algorithms and triggers to control which Cisco CSR 1000v provides the forwarding services for a given route.

# Migrating from High Availability Version 1 to Version 2

Choose one of the following deployment options for High Availability(HA) on Microsoft Azure, on Cisco IOS XE Fuji 16.9.x:

- HA Version 1—continues to be supported in Cisco IOS XE Fuji 16.9.x. However, this will be deprecated in all releases later than Cisco IOS XE Fuji 16.9.x.
- HA Version 2 with Redundancy Node Configuration in Cisco IOS XE. Configuration steps using Cisco IOS XE CLI commands is supported in Cisco IOS XE Fuji 16.9.1 and provides access to the new features in HA version 2. This allows you to continue using your existing redundancy node configurations. However, this deployment option will be deprecated in all releases later than Cisco IOS XE Fuji 16.9.x.
- HA Version 2 with Redundancy Node Configuration in the guestshell. Configuration steps using guestshell-based Python scripts is supported in Cisco IOS XE Fuji 16.9.1. This is the preferred deployment method. If you currently use Redundancy Node Configuration in Cisco IOS XE, we recommend that, during the period of time that you are using Cisco IOS XE Fuji 16.9.x., you migrate to using Redundancy Node Configuration in the guestshell.

**Note** By default, in Cisco IOS XE Fuji 16.9.x, the Cisco CSR 1000v on Microsoft Azure runs HA Version 1. To run HA Version 2, you must manually install the "csr_azure_ha" package in the guestshell.

### Differences in High Availability across Cisco IOS XE Releases

The following table shows some of the differences between running high availability in various IOS releases.

*Table 2: Support of HA Functions for Different Cisco IOS XE Releases*

| Aspect | Cisco IOS XE 16.5.x to IOS XE 16.8.x | Cisco IOS XE 16.9.x | Cisco IOS XE 16.10.x or later |
|---|---|---|---|
| HA Version 1 | Yes | Yes | No |
| HA Version 2 | No | Yes | Yes |
| Redundancy node configuration in Cisco IOS XE | YES | Yes | Yes |
| Redundancy node configuration in guestshell | No | Yes | Yes |
| Revert back to primary router after recovery | No | Yes | Yes |
| CSR 1000v authentication by Azure Active Directory | Yes | Yes | Yes |
| CSR 1000v authentication by Managed Service Interface | No | Yes | Yes |
| User scripts to update routes | No | Yes | Yes |

### Comparison in the Configuration of HA Version 1 and HA Version 2

The following configuration procedures are unchanged between HA Version 1 and HA Version 2:

- Configuring a Tunnel Between Cisco CSR 1000v Routers
- Configuring EIGRP over Virtual Tunnel Interfaces, on page 9

For configuration of HA Version 1, see Configuring High Availability Version 1 for the CSR 1000v on Microsoft Azure, on page 33.

# Configuring High Availability for the CSR 1000v on Microsoft Azure



# Configuring High Availability in Cisco IOS XE

## Configuring IOX and the Guestshell on Cisco IOS XE

The following Cisco IOS XE configuration shows the commands that are required to access the guestshell. You need not configure these prerequisites as they are included automatically in the startup-config file.

However, if you are upgrading the Cisco IOS XE, the configuration is not applied automatically. In this case, you must configure the prerquisites manually.

```
iox
ip nat inside source list GS_NAT_ACL interface GigabitEthernet1 vrf GS overload
ip route vrf GS 0.0.0.0 0.0.0.0 GigabitEthernet1 192.168.35.1 global
interface VirtualPortGroup0
 vrf forwarding GS
 ip address 192.168.35.101 255.255.255.0
  ip nat inside
 no mop enabled
 no mop sysid
ip access-list standard GS_NAT_ACL
 permit 192.168.35.0 0.0.0.255
app-hosting appid guestshell
 app-vnic gateway1 virtualportgroup 0 guest-interface 0
  guest-ipaddress 192.168.35.102 netmask 255.255.255.0
 app-default-gateway 192.168.35.101 guest-interface 0
 name-server0 8.8.8.8
```

An Event Manager Applet detects that the guestshell reaches the "up" state. The applet then performs an action - it installs the guestshell package by issuing the command: `pip install csr_azure_guestshell~=1.1 --user` . If the guestshell package is not installed, or you want to ensure you have the latest package installed, then you can manually install the package by issuing the `pip install csr_azure_guestshell~=1.1 --user` command at the guestshell prompt. See Installing the csr_azure_guestshell Package, on page 12.

**Note** Configure each Cisco CSR 1000v to use a DNS server. In this example, the Cisco CSR 1000v uses the DNS server in the "name-server 8.8.8.8" command. Refer to the IP Addressing: DNS Configuration Guide.

## Configuring a Tunnel Between Cisco CSR 1000v Routers

Configure an IPsec tunnel or a VxLAN tunnel between Cisco CSR 1000v routers as shown in the following two configuration examples. A tunnel uses either the EIGRP or the BGP routing protocol. The tunnel uses Bidirectional Forwarding Detection (BFD) to detect peer failures.

**Configuration of an IPsec Tunnel**

```
crypto isakmp policy 1
 encr aes 256
 authentication pre-share
crypto isakmp key cisco address 0.0.0.0
!
!
crypto ipsec transform-set uni-perf esp-aes 256 esp-sha-hmac
 mode tunnel
!
!
crypto ipsec profile vti-1
 set security-association lifetime kilobytes disable
 set security-association lifetime seconds 86400
 set transform-set uni-perf
 set pfs group2
!
!
interface Tunnel1
 ip address 192.168.101.1 255.255.255.252
 load-interval 30
 tunnel source GigabitEthernet1
 tunnel mode ipsec ipv4
 tunnel destination 23.96.91.169
 tunnel protection ipsec profile vti-1
 bfd interval 500 min_rx 500 multiplier 3
```

**Note** We recommend that you specify a BFD interval of 500 ms or more. You can increase the BFD timers to account for the varying latency in different regions.

**Configuration of a VxLAN Tunnel**

```
interface Tunnel100
 ip address 192.168.101.1 255.255.255.0
 shutdown
 bfd interval 500 min_rx 500 multiplier 3
 tunnel source GigabitEthernet1
 tunnel mode vxlan-gpe ipv4
 tunnel destination 40.114.93.164
 tunnel vxlan vni 10000
```

**Note** We recommend that you specify a BFD interval of 500 ms or more. You can increase the BFD timers to account for the varying latency in different regions.

## Configuring EIGRP over Virtual Tunnel Interfaces

Configure EIGRP over the virtual tunnel interfaces using the following steps.

**Note** Other than using EIGRP, which is the protocol used in the following steps, you also have the option of using either BGP, or OSPF.

**Before you begin**

Configure either a VxLAN or IPsec tunnel between the Cisco CSR 1000v routers.

**SUMMARY STEPS**

1. **router eigrp** *as-number*
2. **network** *ip-address subnet-mask*
3. **bfd all-interfaces**
4. **end**
5. **show bfd neighbors**

**DETAILED STEPS**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **router eigrp** *as-number*<br>**Example:**<br>`Device(config)# router eigrp 1` | Enables the EIGRP routing process and enters router configuration mode. |
| **Step 2** | **network** *ip-address subnet-mask*<br>**Example:**<br>`network 192.168.101.0 0.0.0.255` | Share the network of the tunnel using EIGRP. |
| **Step 3** | **bfd all-interfaces**<br>**Example:**<br>`Device(config-router)# bfd all-interfaces` | Enables BFD globally on all interfaces that are associated with the EIGRP routing process. |
| **Step 4** | **end**<br>**Example:**<br>`Device(config-router)# end` | Exits router configuration mode and returns the router to privileged EXEC mode. |
| **Step 5** | **show bfd neighbors**<br>**Example:**<br>`Device# show bfd neighbors`<br><br>`IPv4 Sessions`<br>`NeighAddr LD/RD RH/RS State Int`<br>`192.168.101.2 4097/4097 Up Up Tu100` | Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered. |

## Configuring BFD Binding Between Routers

**Before you Begin**

Configure a tunnel between Cisco CSR 1000v routers and configure EIGRP or BGP over the tunnel interface.

**Procedure**

```
redundancy
cloud provider azure 2        << 2 is the node index
bfd 172.13.1.2
```

The cloud provider can be `azure` (commercial cloud) or `azusgov` (U.S. Government cloud). The node index matches the redundancy node that is configured in the guestshell. The IP address in the **bfd** command is that of the peer Cisco CSR 1000v.

**Note** If you make a mistake when you configure the BFD peer address, delete the address entry by executing the `no bfd peer 172.13.1.2` command. Then, enter the correct BFD peer IP address.

Ensure that you do no not run the `no cloud provider azure 2` command, as this command deletes the entire redundancy node. If you delete the entire redundancy node, the redundancy node configured in guestshell with index 2 is also deleted.

## Configuring the Console Timeout

When you start an SSH session to the Cisco CSR 1000v, ensure that you do not configure the terminal VTY timeout as infinite - do not configure: **exec-timeout** 0 0. Use a non-zero value for the timeout; for example, **exec-timeout** 4 0 (this command specifies a timeout of four minutes and zero seconds). The reason why the **exec-timeout** 0 0 command causes an issue is as follows: Azure enforces a timeout for the console idle period of between 4 and 30 minutes. When the idle timer expires, Azure disconnects the SSH session. However, the session is not cleared from the point of view of the Cisco CSR 1000v, as the timeout was set to infinite (by the **exec-timeout** 0 0 configuration command). The disconnection causes a terminal session to be orphaned. The session in the Cisco CSR 1000v remains open indefinitely. If you try to establish a new SSH session, a new virtual terminal session is used. If this pattern continues to occur, the number of allowed simultaneous terminal sessions is reached and no new sessions can be established. In addition to configuring the exec-timeout command correctly, it is also a good practice to delete idle virtual terminal sessions using the commands that are shown in the following example:

```
CSRA# show users
Line User Host(s) Idle Location
2 vty 0 cisco idle 00:07:40 128.107.241.177
* 3 vty 1 cisco idle 00:00:00 128.107.241.177

CSRA# clear line 2
```

If the workarounds in the preceding scenarios are ineffective, as a last resort, you can restart the Cisco CSR 1000v in the Azure portal.

# Configuring High Availability in the Guest Shell

## Installing the csr_azure_guestshell Package

To allow the Cisco CSR 1000v instances to use Microsoft Azure resources and services, install the csr_azure_guestshell Python package. This package is automatically installed in the guestshell when a Cisco CSR 1000v instance is created on the Azure cloud.

However, if this package is not automatically installed, you can do so by performing the following procedure:

**Before you Begin**

Download the csr_azure_guestshell Python package from pypi.org. Ensure that you are not on the SUDO mode when you use the guestshell Python package.

**Procedure**

```
Router# guestshell enable      << enable the guestshell
Router# guestshell                        << enter the guestshell
(guestshell) pip install csr_azure_guestshell~=1.1 --user
```

## Installing the csr_azure_ha Package

**Before you Begin**

Download the csr_azure_guestshell Python package from www.cisco.com to ensure that the Cisco CSR 1000v is running the latest version of the package. The package is not automatically installed in the guestshell; it must be be explicitly installed so that you can run HA version 2.

**Procedure**

```
guestshell enable
guestshell
pip install csr_azure_ha~=1.0 --user            << execute this command in the guestshell
```

## Setting the Path Environment Variable

Update the path environment in the guest shell to specify the location of the configuration scripts.

**Procedure**

Run the following command in the guestshell:

```
source ~/.bashrc
```

## Configuring Redundancy Nodes

Create and modify redundancy nodes using a set of Python scripts. See "Creating a Redundancy Node" and further sections below. Python scripts use parameters that are shown in the following tables:

- Redundancy Node Parameters: required by Python scripts to create, modify, or delete redundancy nodes.
- AAD Redundancy Node Parameters: required by Python scripts if you are authenticating the Cisco CSR 1000v using an application that is defined in the Azure Active Directory.

*Table 3: Redundancy Node Parameters*

| Parameter Name | Switch | Description |
|---|---|---|
| Node index | -i | Index that is used to uniquely identify this node. Valid values: 1–255. |
| Cloud provider | -p | Specifies the type of Azure cloud: azure, azusgov, or azchina. |
| Subscription ID | -s | Azure subscription id. |
| Resource group name | -g | Name of the resource group that contains the route table. |
| Route table name | -t | Name of the route table to be updated. |
| Route | -r | IP address of the route to be updated in CIDR format. Can be IPv4 or IPv6 address.<br>If a route is unspecified, then the redundancy node is considered to apply to all routes in the routing table of type "virtual appliance". |
| Next hop address | -n | IP address of the next hop router. Use the IP address that is assigned to this CSR 1000v on the subnet which utilizes this route table. Can be an IPv4 or IPv6 address. |
| Mode | -m | Indicates whether this router is the primary or secondary router for servicing this route. Default value is secondary. |

*Table 4: AAD Redundancy Node Parameters*

| Parameter Name | Switch | Description |
|---|---|---|
| Tenant ID | -d | Identifies the AAD instance. |
| Application ID | -a | Identifies the application in AAD. |

| Parameter Name | Switch | Description |
|---|---|---|
| Application key | -k | Access key that is created for the application.<br>**Note** For High Availability version 2, the key must be URL unencoded. For High Availability version 1, the key must be URL encoded. |

## Creating a Redundancy Node

Run the following script to create a redundancy node and add it to the database.

**create_node.py** { *switch value* } [...[{ *switch value* }]

*switch*—See: Configuring Redundancy Nodes.

A valid redundancy node must have the following parameters configured:

- Node index
- Cloud provider
- Subscription ID
- Resource group name
- Route table name

**Example**

```
create_node.py -i 10 -p azure -s b0b1a9e2-4444-4ca5-acd9-bebd1e6873eb -g ds-rg -t
ds-sub2-RouteTable
```

```
-r 15.0.0.0/8 -n 192.168.7.4
```

If successful, the script returns a value of zero.

## Setting Redundancy Node Parameters

If you need to change the value of parameters in an existing redundancy node, run the following script.

**set_params.py** { *switch value* } [...[{ *switch value* }]

**Example**

```
set_params.py -i 10 -r 15.0.0.0/16 -n 192.168.7.5
```

The index parameter (`-i`) is required. This command sets the values of any specified parameters. If the specified parameter has already been defined for the redundancy node, then the value of the parameter is updated.

When a node index value of zero is specified, the values that are provided by the command for the specified parameters are treated as the default values for these parameters.

If successful, the script returns a value of zero.

## Clearing Redundancy Node Parameters

If you want to clear the value of specified parameters, for an existing redundancy node, run the following script.

**clear_params.py** -i *value* { *value* } [...[{ *switch* *value* }]

**Example**

In this example, the clear_params script clears both the route and next hop address parameters.

```
clear_params.py -i 10 -r -n
```

Only the index parameter is required. The values of any additional specified parameters are cleared.

If you specify a node index value of zero, the parameter values that are provided are treated as the default values for the parameters.

If successful, the script returns a value of zero.

## Show Node

Run the following script to display the parameter values of an existing redundancy node.

**show_node.py** { -i *value* }

**Example**

```
show_node.py -i 10
```

-i specifies the index of the redundancy node (0–255).

If the script is successful, it displays the parameters of the specified node. If the script is unsuccessful, it displays an error message.

## Delete Node

Run the following script to delete an existing redundancy node.

**delete_node.py** { -i *value* }

**Example**

```
delete_node.py -i 10
```

-i specifies the index of the redundancy node (0–255).

The `-i` parameter must be specified. The node is expected to exist in the database. If the client tries to delete a node that is not in the database, an error is not generated. If successful, the script returns a value of zero and it displays the parameters of the specified node. If the script is unsuccessful, a non-zero return value is produced and an error message is written to the log file.

If the script is successful, it displays an error message.

**Note** The database of redundancy nodes is maintained on a virtual disk that is allocated to the guestshell. If you issue the following Cisco IOS XE command: **guestshell destroy**, then the virtual disk is deleted and all node configurations are lost.

If you want to recover from the loss of a deleted virtual disk, manually perform the following steps:

1. Enable the guestshell.
2. Install the "csr_azure_guestshell" Python package.
3. Install the "csr_azure_ha" Python package.
4. Run Python scripts to reconfigure all of the redundancy nodes.

# Configuring High Availability in Microsoft Azure

## Before You Begin

Before configuring High Availability for CSR 1000v on Microsoft Azure, you require:

- A virtual network setup in Microsoft Azure with two subnets.
- Two Cisco CSR 1000v VMs.
- Licenses for each Cisco CSR 1000v:

  (Cisco IOS XE Everest 16.6.1 or later) Enable the AX or SEC license.

  (Cisco IOS XE Everest 16.5.1 or earlier) Enable the AX license.

## Methods for Configuring Microsoft Azure

The following methods can be used for configuring Microsoft Azure:

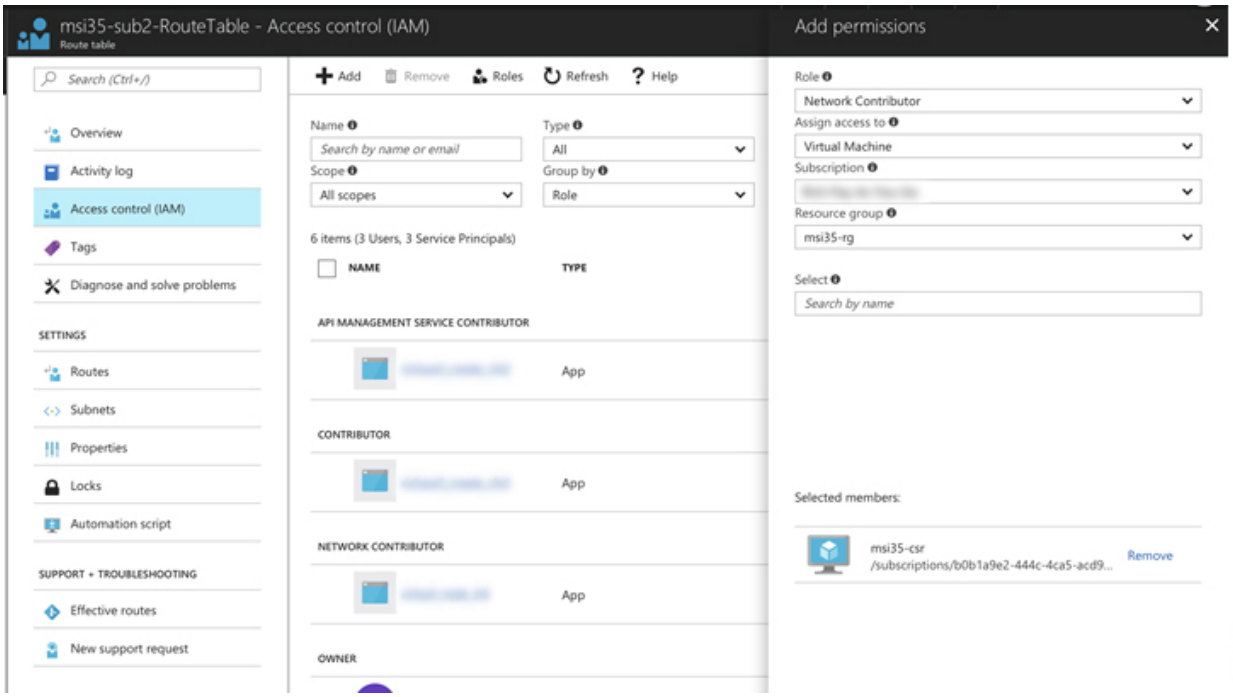
- Microsoft Azure CLI commands
- Powershell commands
- Microsoft Azure Portal https://portal.azure.com/

## Configuring IAM for the Route Table

This section explains how you can configure the route table of a subnet to allow a VM to modify the Cisco CSR 1000v route table.

**Step 1** To add an application into an existing network, in the **All resources** pane, choose a private side subnet in the left pane; for example, "subnet2-CSR-RouteTable".

**Example:**




**Step 2** Select "Access control (IAM)" and click **+Add**.

**Step 3** In the "Role" textbox, choose **Network contributor**.

**Step 4** In the "Assign access to" checkbox, select "Virtual Machine".

**Step 5** Enter your subscription ID.

**Step 6** Enter the Resource group.

**Step 7** A list of your Cisco CSR 1000v appears in the list of machines under "Selected members". Select the VM for which you want to assign permissions to change the route table.

**Note** If you expect to see a Cisco CSR 1000v under "Selected members" but it is not displayed, then check that you entered a valid subscription ID, resource group and also check that your Cisco CSR 1000v is running.

**Step 8** Click **Save**.

# Configuring the Network Security Group

If you have a network security group attached to interface NIC0 of the router, configure an inbound and outbound security rule for the network security group. Each rule must allow ports 4789 and 4790 to pass BFD protocol traffic. To set up a security rule on a group, see: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/nsg-quickstart-portal.

# Configuring an Application in Azure Active Directory

In HA version 2, you need not explicitly create an application in the Azure Active Directory and grant it permission to access the route tables. An application representing the CSR 1000v is automatically created in the Azure Active Directory via the Managed Service Identity service. However, you can continue to use a

manually configured application (as used in HA Version 1). For more information, see Create an Application in a Microsoft Azure Active Directory, on page 33.

# Configuring High Availability for the CSR 1000v on Microsoft Azure: Examples

## Single Route Table and Two Secondary Routers—Example

In this example, the two peer routers are connected to the same subnet. One of the two routers is active for the specified redundancy node. Since both redundancy nodes are configured in secondary mode, the active router does not change after the recovery of a failed CSR.

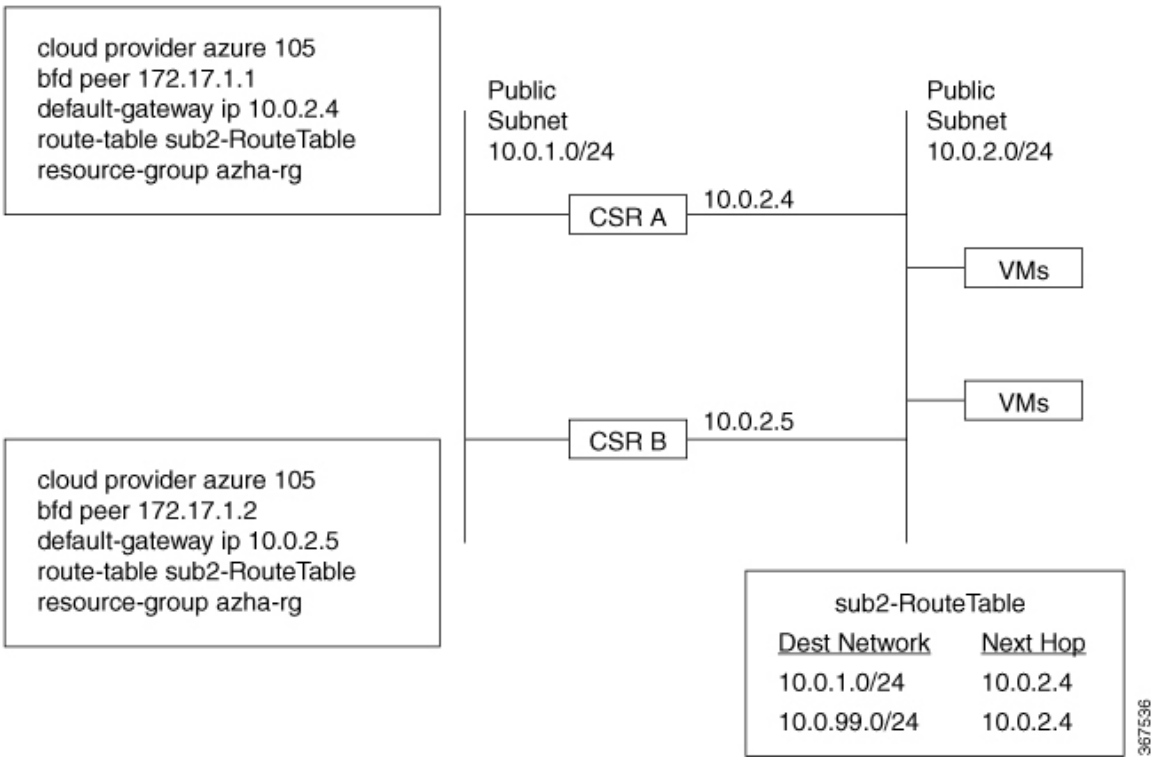



*Table 5: CSR A Node Parameters*

| Parameter | Value |
|---|---|
| Node index | 105 |
| BFD Peer | 172.17.1.1 |
| Next Hop IP | 10.0.2.4 (IP address of CSR A) |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |

*Table 6: CSR B Node Parameters*

| Parameter | Value |
|---|---|
| Node Index | 105 |
| BFD Peer | 172.17.1.1 |
| Next Hop IP | 10.0.2.5 (IP address of CSR B) |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |

**Note** The route name was not configured for either node, which causes all the routes in the sub2-RouteTable to be updated with the Next Hop IP after a peer failure event.

## Single Route Table, One CSR Primary

In this example, the two peer routers are connected to the same subnet. One of the two routers is active for the specified redundancy node. CSR A is configured to be the primary router for this subnet. In the absence of a failure of CSR A, it is the active router for all routes in the route table. If CSR A fails, CSR B temporarily becomes the active router. After CSR A recovers from the failure, it becomes the active router.

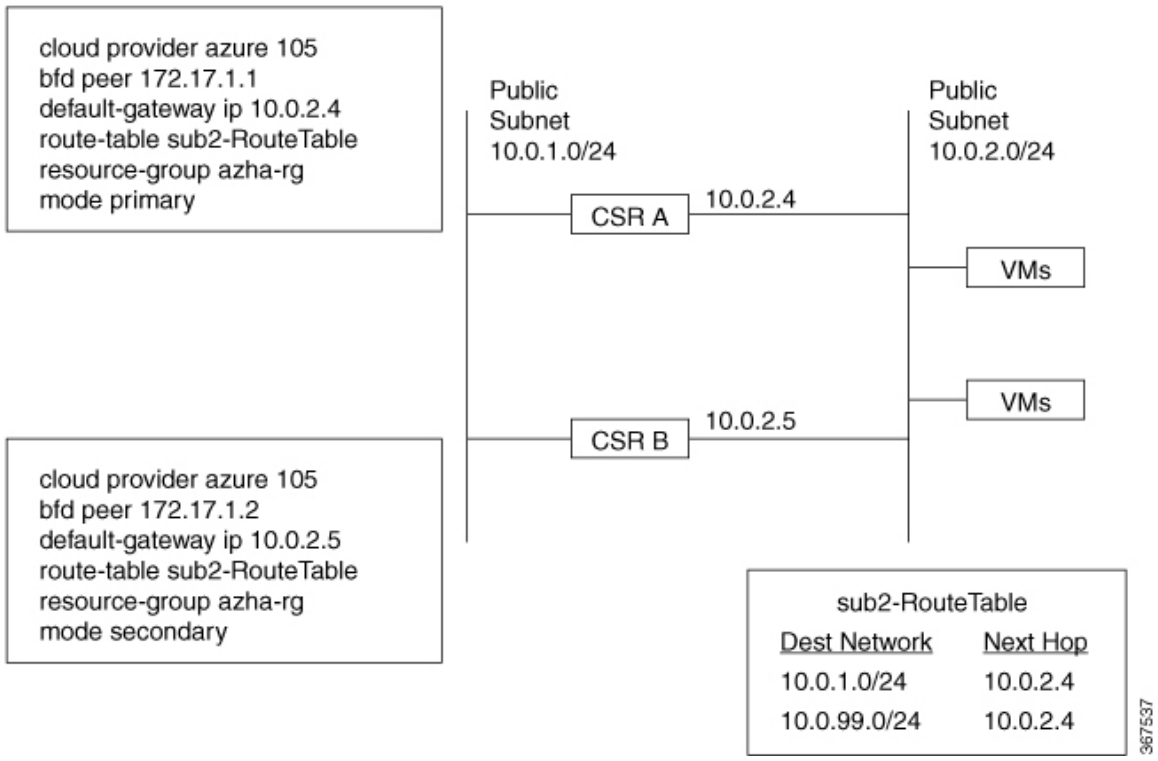


*Table 7: CSR A Node Parameters*

| Parameter | Value |
|---|---|
| Node index | 105 |
| BFD Peer | 172.17.1.2 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.4 (IP address of CSR A on the subnet) |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |
| Mode | Primary |

*Table 8: CSR B Node Parameters*

| Parameter | Value |
|---|---|
| Node index | 105 |
| BFD Peer | 172.17.7.1 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.5 (IP address of CSR A on the subnet) |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |
| Mode | Secondary |

## Single Route Table, Nodes for Individual Routes, Both CSRs Secondary

In this example, the two peer routers are connected to the same subnet. The route table contains two routes which get split between the two CSR 1000vs. All nodes are configured in secondary mode.

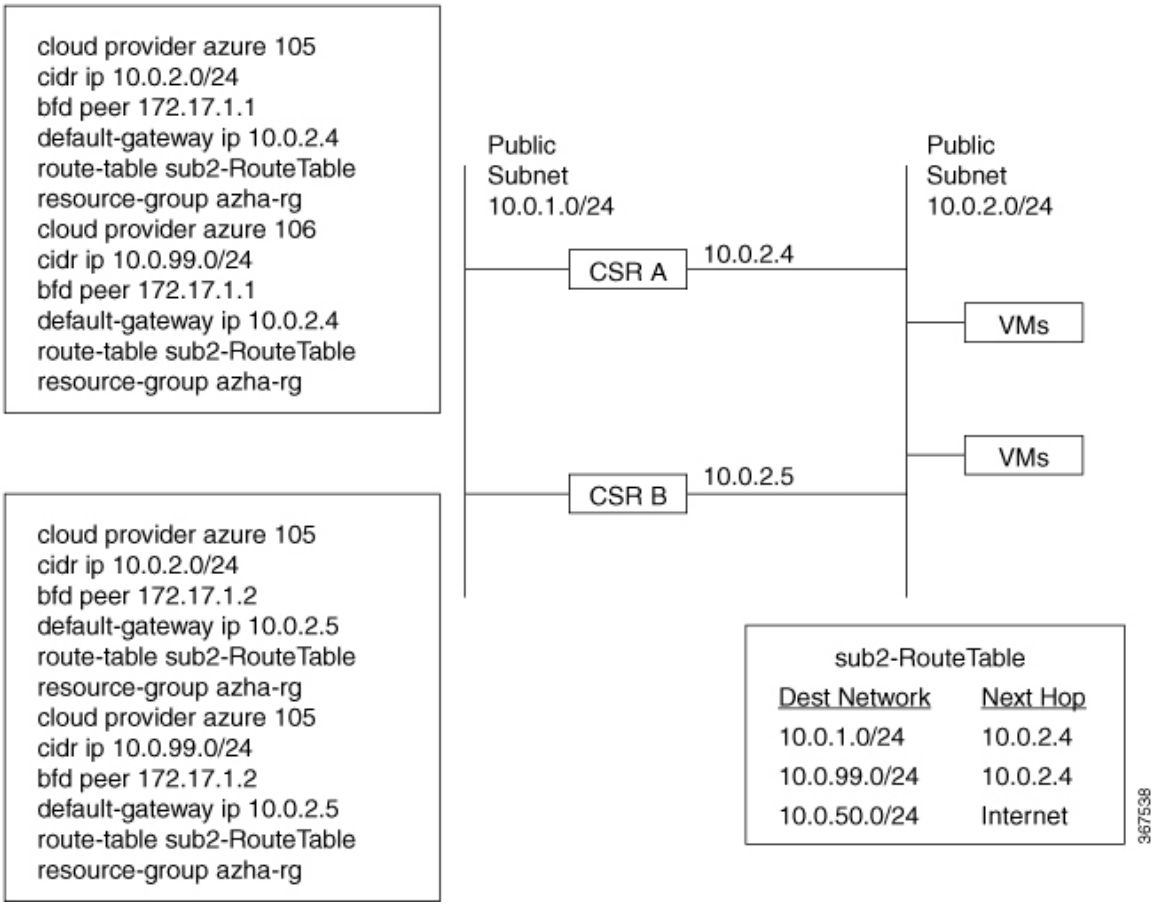



*Table 9: CSR A Node Parameters (First node)*

| Parameter | Value |
|---|---|
| Node index | 105 |
| BFD Peer | 172.17.7.1 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.5 (IP address of CSR A on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |
| Mode | Primary |

*Table 10: CSR A Node Parameters (Second node)*

| Parameter | Value |
|---|---|
| Node index | 106 |

| Parameter | Value |
|---|---|
| BFD Peer | 172.17.7.1 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.5 (IP address of CSR A on the subnet) |
| Route | 16.0.0.0/8 |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |
| Mode | Primary |

*Table 11: CSR B Node Parameters (First node)*

| Parameter | Value |
|---|---|
| Node index | 205 |
| BFD Peer | 172.17.7.1 (Address of CSR A on the tunnel) |
| Next Hop IP | 10.0.2.5 (IP address of CSR B on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |
| Mode | Secondary |

*Table 12: CSR B Node Parameters (Second node)*

| Parameter | Value |
|---|---|
| Node index | 206 |
| BFD Peer | 172.17.7.1 (Address of CSR A on the tunnel) |
| Next Hop IP | 10.0.2.5 (IP address of CSR B on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | sub2-RouteTable |
| Resource Group | azha-rg |
| Mode | Secondary |

## Two Route Tables, One CSR Primary

In this example, the two peer routers are connected to two different subnets: CSR A on subnet A(10.0.2.0/24) and CSR B on subnet B (10.0.3.0/24). CSR A is the primary router on subnet A (with IP address 10.0.2.4). CSR B is the primary router on subnet B (with IP address 10.0.4.5).

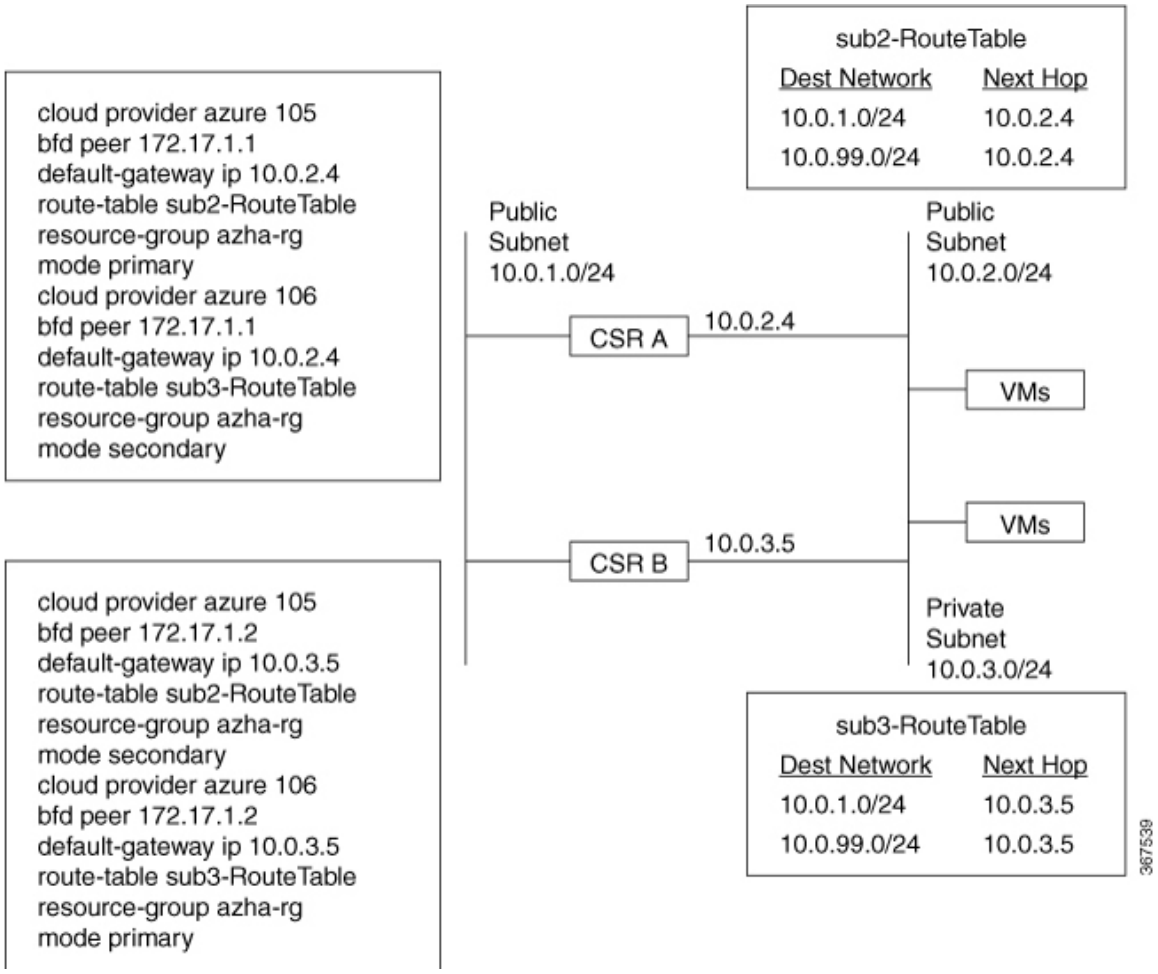



*Table 13: CSR A Node Parameters (First Node)*

| Parameter | Value |
|---|---|
| Node index | 105 |
| BFD Peer | 172.17.7.2 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.4 (IP address of CSR A on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | subA-RouteTable |
| Resource Group | azha-rg |

| Parameter | Value |
|---|---|
| Mode | Primary |

*Table 14: CSR A Node Parameters (Second Node)*

| Parameter | Value |
|---|---|
| Node index | 106 |
| BFD Peer | 172.17.7.2 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.4 (IP address of CSR A on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | subA-RouteTable |
| Resource Group | azha-rg |
| Mode | Primary |

*Table 15: CSR A Node Parameters (Third Node)*

| Parameter | Value |
|---|---|
| Node index | 205 |
| BFD Peer | 172.17.7.2 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.4 (IP address of CSR A on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | subB-RouteTable |
| Resource Group | azha-rg |
| Mode | Secondary |

*Table 16: CSR A Node Parameters (Fourth Node)*

| Parameter | Value |
|---|---|
| Node index | 206 |
| BFD Peer | 172.17.7.2 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.2.4 (IP address of CSR A on the subnet) |
| Route | 16.0.0.0/8 |
| Route Table | subB-RouteTable |
| Resource Group | azha-rg |

| Parameter | Value |
|---|---|
| Mode | Secondary |

*Table 17: CSR B Node Parameters (First Node)*

| Parameter | Value |
|---|---|
| Node index | 105 |
| BFD Peer | 172.17.7.1 (Address of CSR A on the tunnel) |
| Next Hop IP | 10.0.4.5 (IP address of CSR B on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | subB-RouteTable |
| Resource Group | azha-rg |
| Mode | Primary |

*Table 18: CSR B Node Parameters (Second Node)*

| Parameter | Value |
|---|---|
| Node index | 106 |
| BFD Peer | 172.17.7.1 (Address of CSR A on the tunnel) |
| Next Hop IP | 10.0.4.5 (IP address of CSR B on the subnet) |
| Route | 16.0.0.0/8 |
| Route Table | subB-RouteTable |
| Resource Group | azha-rg |
| Mode | Primary |

*Table 19: CSR B Node Parameters (Third Node)*

| Parameter | Value |
|---|---|
| Node index | 205 |
| BFD Peer | 172.17.7.1 (Address of CSR A on the tunnel) |
| Next Hop IP | 10.0.4.5 (IP address of CSR B on the subnet) |
| Route | 15.0.0.0/8 |
| Route Table | subA-RouteTable |
| Resource Group | azha-rg |

| Parameter | Value |
|---|---|
| Mode | Secondary |

*Table 20: CSR B Node Parameters (Fourth Node)*

| Parameter | Value |
|---|---|
| Node index | 206 |
| BFD Peer | 172.17.1.1 (Address of CSR B on the tunnel) |
| Next Hop IP | 10.0.4.5 (IP address of CSR A on the subnet) |
| Route | 16.0.0.0/8 |
| Route Table | subA-RouteTable |
| Resource Group | azha-rg |
| Mode | Secondary |

## Route Table Entry Types

The route tables in Microsoft Azure support different entry types. The entry type for a route can be one of the following: Virtual network gateway, Internet, or Virtual Appliance. The next hop address identifies a resource in the Azure network.

Routes with an entry type of Virtual network gateway or Internet do not have an explicit IP address for the next hop and are not supported by the High Availability feature.

(Cisco IOS XE Everest 16.6) When you configure High Availability on the Cisco CSR 1000v, all the routes within a route table must have an entry type of Virtual Appliance. These routes require an explicit IP address for the next hop.

(Cisco IOS XE Everest 16.7 or later) When you configure High Availability on the Cisco CSR 1000v, you can specify individual routes to be updated in the case of failure. Ensure that you configure each individual route as having an entry type of Virtual Appliance. If you configure a redundancy node that represents all of the entries in the route table, ensure that all of the routes have an entry type of Virtual Appliance.

# Verifying High Availability for the CSR 1000v on Microsoft Azure using Cisco IOS XE Commands

Enter the following command to verify that the tunnel interface is configured and enabled:

```
# show ip interface brief
                      IP-Address      OK? Method Status                Protocol
GigabitEthernet1      192.168.35.20   YES DHCP   up                    up
GigabitEthernet2      192.168.36.12   YES DHCP   up                    up
Tunnel1               172.17.1.1      YES NVRAM  up                    up
VirtualPortGroup0     192.168.35.101  YES NVRAM  up                    up
```

Use the following command to verify that neighboring Cisco CSR 1000v routers have established a BFD session:

```
# show bfd neighbors

IPv4 Sessions
NeighAddr                              LD/RD         RH/RS     State     Int
172.17.1.2                             4097/0        Down      Down      Tu1
```

Use the following command to check the binding between a redundancy node and the BFD peer:

```
# show redundancy cloud provider azure 100
Cloud HA: work_in_progress=FALSE
Provider : AZURE node 100
 State : idle
 BFD peer     = 172.17.1.2
 BFD intf     = Tunnel1
```

Use the following command to verify that the high availability configuration commands that are entered in the preceding sections appear in the running configuration:

```
# show running-configuration int Tunnel1
Building configuration...

Current configuration : 225 bytes
!
interface Tunnel1
 ip address 172.17.1.1 255.255.255.252
 bfd interval 500 min_rx 500 multiplier 3
 tunnel source GigabitEthernet1
 tunnel mode vxlan-gpe ipv4
 tunnel destination 40.117.153.111
 tunnel vxlan vni 10000
```

Save the configuration for future use, with the command: **copy running-configuration startup-configuration**

High availability depends upon several services to be running in guestshell. Enter the following command to verify that IOX is configured and running:

```
# show iox

Virtual Service Global State and Virtualization Limits:
Infrastructure version : 1.7
Total virtual services installed : 0
Total virtual services activated : 0
Machine types supported   : LXC
Machine types disabled    : KVM
Maximum VCPUs per virtual service : 1
Resource virtualization limits:
Name                        Quota     Committed     Available
------------------------------------------------------------------
system CPU (%)                 75             0            75
memory (MB)                  3072             0          3072
bootflash (MB)              20000             0          5745

IOx Infrastructure Summary:
---------------------------
IOx service (CAF)    : Running
IOx service (HA)     : Not Running
IOx service (IOxman) : Running
Libvirtd             : Running
```

Enter the following command to verify that the guest application is defined and running:

```
# show app-hosting list
```

```
App id                            State
-----------------------------------------------------
guestshell                        RUNNING
```

If the guestshell state shows as DEPLOYED in the output of the preceding command, then enable the guestshell using the following command:

```
# guestshell enable

Interface will be selected if configured in app-hosting
Please wait for completion
guestshell activated successfully
Current state is: ACTIVATED
guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully
```

# Verifying High Availability for the CSR 1000v on Microsoft Azure using Guestshell Commands

Enter the following command in Cisco IOS XE to enter the guestshell:

```
guestshell
```

Navigate to the home directory and find a subdirectory named "azure":

```
cd ~
ls
```

Verify that the authentication token service is running:

```
[guestshell@guestshell azure]$ systemctl status auth-token
  auth-token.service - Authentication Token service
Loaded: loaded (/etc/systemd/user/auth-token.service; enabled; vendor preset: disabled)
Active: active (running) since Wed 2018-06-13 19:56:00 UTC; 5min ago
Main PID: 36 (python)
CGroup: /system.slice/libvirtd.service/system.slice/auth-token.service
└─36 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_guestshell/TokenMgr...
```

If the service is not running, start the service by executing the following command:

```
[guestshell@guestshell azure]$ sudo systemctl start auth-token
```

Verify the high availability service is running:

```
 [guestshell@guestshell azure]$ sudo systemctl start waagent

 [guestshell@guestshell azure]$ systemctl status azure-ha
azure-ha.service - Azure High Availability service
Loaded: loaded (/etc/systemd/user/azure-ha.service; enabled; vendor preset: disabled) Active:
 active (running) since Wed 2018-06-13 19:56:00 UTC; 7min ago
Main PID: 29 (python)
CGroup: /system.slice/libvirtd.service/system.slice/azure-ha.service
├─ 29 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_ha/server/ha_serve...
└─103 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_ha/server/ha_serve...
```

If the service is not running, start it with the command:

```
[guestshell@guestshell azure]$ sudo systemctl start azure-ha
```

Check if the Python path has been set

```
[guestshell@guestshell azure]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py If this path is
not found, run this command in guestshell [guestshell@guestshell azure]$ source ~/.bashrc
```

Use the show_node script to display a node you have configured:

```
[guestshell@guestshell HA]$ show_node.py -i 1
Redundancy node configuration: index 1
routeTableName azha1-sub2-RouteTable

route  15.0.0.0/8 nextHop 192.168.35.102
resourceGroup azha-rg
subscriptionId b0b1a9e2-4444-4ca5-acd9-bebd1e6873eb cloud azure
```

Simulate a peer failure for this redundancy node. This is a diagnostic tool to verify that the Cisco CSR 1000v can read and write the route table specified by the redundancy node. It does not change the entry in the route table. It writes a verbose log file to the directory `~/azure/HA/events`. Examine this log file to verify the operation was successful.

```
[guestshell@guestshell events]$ node_event.py -i 1 -e verify
[guestshell@guestshell events]$ pwd
/home/guestshell/azure/HA/events
[guestshell@guestshell events]$ ls
event.2018-06-13 20:10:21.093942
```

Open the event file. It is a debug log of the attempt to read and update the route described by the redundancy node. If everything worked, the last line of the file will be Event handling completed.

Use the show_node script to display a node that you previously configured:

```
[guestshell@guestshell HA]$ show_node.py -i 1
Redundancy node configuration:
index  1
routeTableName  azha1-sub2-RouteTable
route  15.0.0.0/8
nextHop  192.168.35.102
resourceGroup  azha-rg
subscriptionId  b0b1a9e2-4444-4ca5-acd9-bebd1e6873eb
cloud   azure
```

If the service is not running, start it with the following command:

```
[guestshell@guestshell azure]$ sudo systemctl start auth-token
```

Use the following command to verify that the high availability service is running:

```
guestshell@guestshell azure]$ systemctl status azure-ha
● azure-ha.service - Azure High Availability service
   Loaded: loaded (/etc/systemd/user/azure-ha.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2018-06-13 19:56:00 UTC; 7min ago
 Main PID: 29 (python)
   CGroup: /system.slice/libvirtd.service/system.slice/azure-ha.service
           ├─ 29 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_ha/server/ha_serve...
           └─103 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_ha/server/ha_serve...
```

If the service is not running, start it with the following command:

```
[guestshell@guestshell azure]$ sudo systemctl start azure-ha
```

Check if the Python path has been set using the following command:

```
[guestshell@guestshell azure]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py
```

If this path is not found, run these commands in the guestshell:

```
[guestshell@guestshell azure]$ source ~/.bashrc

[guestshell@guestshell azure]$ sudo systemctl start waagent
```

Verify that the authentication token service is running:

```
[guestshell@guestshell azure]$ systemctl status auth-token
● auth-token.service - Authentication Token service
   Loaded: loaded (/etc/systemd/user/auth-token.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2018-06-13 19:56:00 UTC; 5min ago
 Main PID: 36 (python)
   CGroup: /system.slice/libvirtd.service/system.slice/auth-token.service
           └─36 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_guestshell/TokenMgr...
```

If the service is not running, start it with the command

```
[guestshell@guestshell azure]$ sudo systemctl start auth-token
```

Verify the high availability service is running

```
[guestshell@guestshell azure]$ systemctl status azure-ha
 azure-ha.service - Azure High Availability service
  Loaded: loaded (/etc/systemd/user/azure-ha.service; enabled; vendor preset: disabled)
  Active: active (running) since Wed 2018-06-13 19:56:00 UTC; 7min ago
Main PID: 29 (python)
  CGroup: /system.slice/libvirtd.service/system.slice/azure-ha.service
          ├─ 29 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_ha/server/ha_serve...
          └─103 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_azure_ha/server/ha_serve...
```

If the service is not running, start it with the command:

```
[guestshell@guestshell azure]$ sudo systemctl start azure-ha
```

Check if the Python path has been set

```
[guestshell@guestshell azure]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py If this path is
not found, run this command in guestshell [guestshell@guestshell azure]$ source ~/.bashrc
```

Use the show_node script to display a node you have configured:

```
[guestshell@guestshell HA]$ show_node.py -i 1
Redundancy node configuration: index 1
routeTableName azha1-sub2-RouteTable

route  15.0.0.0/8 nextHop 192.168.35.102
resourceGroup azha-rg
subscriptionId b0b1a9e2-4444-4ca5-acd9-bebd1e6873eb cloud azure
```

Simulate a peer failure for this redundancy node. This is a diagnostic tool to verify the CSR is capable of reading and writing the route table specified by the redundancy node. It does not change the entry in the route table. It will write a verbose log file to the directory ~/azure/HA/events. Examine this log file to verify the operation was successful.

```
[guestshell@guestshell events]$ node_event.py -i 1 -e verify
[guestshell@guestshell events]$ pwd
/home/guestshell/azure/HA/events
```

```
[guestshell@guestshell events]$ ls
 event.2018-06-13 20:10:21.093942
```

Change into the azure directory and find subdirectories named HA, tools, and waagent.

```
cd azure
ls
```

Open the event file using **vi**, or view the event file using **cat**. The file is a debug log of the attempts made to read and update the route described in the redundancy node. If everything works, the last line of the file is Event handling completed.

Verify the Linux Azure agent is running:

```
[guestshell@guestshell azure]$ systemctl status waagent
waagent.service - Azure Linux Agent
   Loaded: loaded (/usr/lib/systemd/system/waagent.service; enabled; vendor preset: disabled)

   Active: active (running) since Wed 2018-06-13 19:56:00 UTC; 5min ago
 Main PID: 28 (python)
   CGroup: /system.slice/libvirtd.service/system.slice/waagent.service
           ├─28 /usr/bin/python -u /usr/sbin/waagent -daemon
           └─92 python -u /usr/sbin/waagent -run-exthandlers
```

If the service is not running, start it with the following command:

```
[guestshell@guestshell azure]$ sudo systemctl start atd
```

Check if the Python path has been set:

```
[guestshell@guestshell azure]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py
```

If this path is not found, run the following command in the guestshell

```
[guestshell@guestshell azure]$ source ~/.bashrc
```

Use the show_node script to display a node you have previously configured:

```
[guestshell@guestshell HA]$ show_node.py -i 1
Redundancy node configuration:
index  1
routeTableName  azha1-sub2-RouteTable
route  15.0.0.0/8
nextHop  192.168.35.102
resourceGroup  azha-rg
subscriptionId  b0b1a9e2-4444-4ca5-acd9-bebd1e6873eb
cloud   azure
```

Simulate a peer failure for this redundancy node. This is a diagnostic tool that verifies that the Cisco CSR 1000v is capable of reading and writing the route table specified by the redundancy node. It does not change the entry in the route table. It writes a verbose log file to the directory `~/azure/HA/events`. Examine this log file to verify the operation was successful.

```
[guestshell@guestshell events]$ node_event.py -i 1 -e verify
[guestshell@guestshell events]$ pwd
/home/guestshell/azure/HA/events
[guestshell@guestshell events]$ ls
event.2018-06-13 20:10:21.093942
```

Open the event file. It is a debug log of the attempt to read and update the route described by the redundancy node. If everything worked, the last line of the file contain the message: Event handling completed.

# Advanced Programming for High Availability on Microsoft Azure

Cisco CSR 1000v routers use the BFD protocol to detect failure of their peer router and trigger a peerFail event. However, the BFD protocol does not detect some failures. For example, if the Gigabit Ethernet interface on CSR A connected to the back-end network goes down, or stops forwarding traffic.

This section describes a few advanced programming techniques that you can use to customize your control of failover and reversion events.

## Trigger Cisco CSR 1000v Failover Using EEM

Define an Event Manager applet to detect the transition of an interface state. The applet triggers a failover of the Cisco CSR 1000v.

**Before You Begin**

Enter configuration mode in the Cisco IOS XE CLI of the Cisco CSR 1000v.

**Procedure**

```
event manager applet Interface GigabitEthernet2
event syslog pattern Interface GigabitEthernet2, changed state to down
action 1 cli command enable
action 2 cli command guestshell run node_event.py -i 10 -e peerFail
exit
exit
```

## Trigger Cisco CSR 1000v Reversion After Recovery

Define an Event Manager applet to detect when the router recovers and re-establishes a BFD session with its peer router. The applet reverts a route entry that previously changed, by pointing the route entry back to this router.

**Before You Begin**

Enter configuration mode in the Cisco IOS XE CLI of the Cisco CSR 1000v.

**Procedure**

```
event manager applet "bfd_session_up
event syslog pattern .*BFD_SESS_UP.*
action 1 cli command enable
action 2 cli command "guestshell run node_event.py -i 10 -e revert"
exit
```

## User-Defined Triggers

You can write your own Python script to recognize an event or condition and call the node_event script. You can enter the command manually at the guestshell prompt.

**Example**

To process the redundancy node and update an associated route table entry, run the `node_event` Python script. `node_event.py -i` *`node_index`* `-e peerFail` processes the redundancy node and updates the associated route table entry.

# Configuring High Availability Version 1 for the CSR 1000v on Microsoft Azure

## Create an Application in a Microsoft Azure Active Directory

This section explains how to create an application in a Microsoft Azure Active Directory with permissions to access Microsoft Azure Resource Manager APIs. These configuration steps use the classic portal.

**Note** When entering the API Access Key using High Availability version 1, the key must be URL encoded.

When entering the API Access Key using High Availability version 2, the key must be URL unencoded.

Refer to Create an Authentication Key for the Application, on page 34.

**Step 1** Go to the portal for Microsoft Azure: https://portal.azure.com.

**Step 2** Choose your account name and sign in using your Microsoft Azure password.

**Step 3** Click **Azure Active Directory** in the left navigation pane and select an active directory in the main pane. Click **Switch directory** at the top of the pane to select the active directory.

**Step 4** Verify that you are authorized to create a new application. Refer to the following Microsoft Azure documentation for creating an application in the Azure Active Directory: Use portal to create an Azure Active Directory application and service principal that can access resources.

**Step 5** To view applications, select **App registrations**.

**Step 6** To create a new application, select **New application registration**.

**Step 7** Specify the name of the application and ensure that "Web App / API" is selected as the Application type.

**Step 8** Specify the Sign-on URL. Use a name for the sign-on URL which is in the URI format, but it does not have to be reachable. (Note that the APP-ID URI is not the App ID.) You can use a string in the following format: "http://<your_directory_domain_name>/<app_name>". For example, if your application name is "myapp" and the domain name of your directory is "\mydir.onmicrosoft.com", use the following example as the sign-on URL:

**Example:**

```
http://mydir.onmicrosoft.com/myapp
```

**Step 9** Click **Create**.

**Step 10** Click the checkmark symbol at the bottom right of the dialog box.

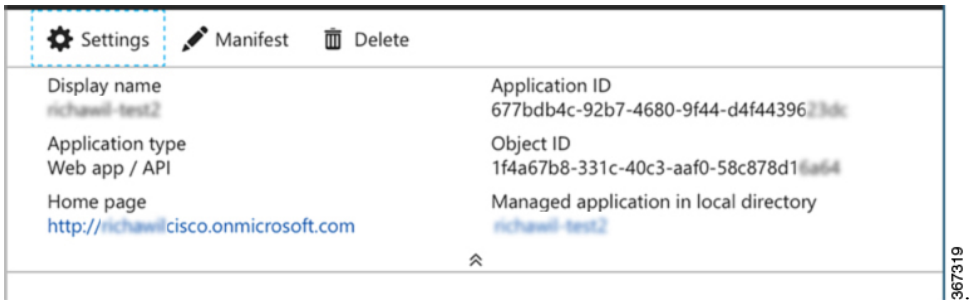
**Step 11** Under the name of the application that you have added, click "CONFIGURE".

### What to do next

Go to Obtain the Application ID and Tenant ID, on page 34.

# Obtain the Application ID and Tenant ID

**Step 1** After you create the application, the registered app should appear on the screen as shown below.




Also refer to step 2 in section "Get application ID and authentication key" in the Microsoft Documentation: Use portal to create an Azure Active Directory application and service principal that can access resources

**Step 2** Take a note of the "Application ID".

**Step 3** Select **Azure Active Directory**.

**Step 4** Select **Properties**.

**Step 5** Take a note of the value of the **`Directory ID`** field. This is your tenant ID.

### What to do next

Go to Create an Authentication Key for the Application, on page 34.

# Create an Authentication Key for the Application

Create an authentication key for the application by performing the following steps:

**Step 1** Select **Azure Active Directory**.

**Step 2** Select **App registrations**.

**Step 3** Select the application that you previously created in Obtain the Application ID and Tenant ID, on page 34.

**Step 4** Select **Settings**.

**Step 5** To create a key for API access, select **Keys** and choose a value for **Duration**—the length of time until the key becomes invalid.

**Step 6** Make a note of the API key from the Value field.

**Step 7** For HA Version 1, convert the API key to URL encoded format. (To find a suitable conversion tool, enter URL encoder into an internet search engine.) Having a URL encoded API key prevents issues later;for example, when the API key is used in step 10 of Configure Failure Detection for the Cisco CSR 1000v on Microsoft Azure, on page 41.

**Note** When entering the API Access Key using High Availability version 1, the key must be URL encoded.

When entering the API Access Key using High Availability version 2, the key must be URL unencoded.

**Note** Store the API key carefully as it cannot be retrieved later.

**Example:**

API Key (unencoded)

```
5yOhH593dtD/O8gzAlWgulrkWz5dH02d2STk3LDbI4c=
```

API Key (URL encoded)

```
5yOhH593dtD%2FO8gzAlWgulrkWz5dH02d2STk3LDbI4c%3D
```

**Example:**

(Python 3.x example)

```
import urllib.parse
s = HOC6OhG5aJ4phxMSFVRtz59qWgIuD4C2zXd7Q7v5EFk=  # API Key (unencoded)
urllib.parse.quote_plus(s)
HOC6OhG5aJ4phxMSFVRtz59qWgIuD4C2zXd7Q7v5EFk%3D    # API Key (URL encoded)
```

### What to do next

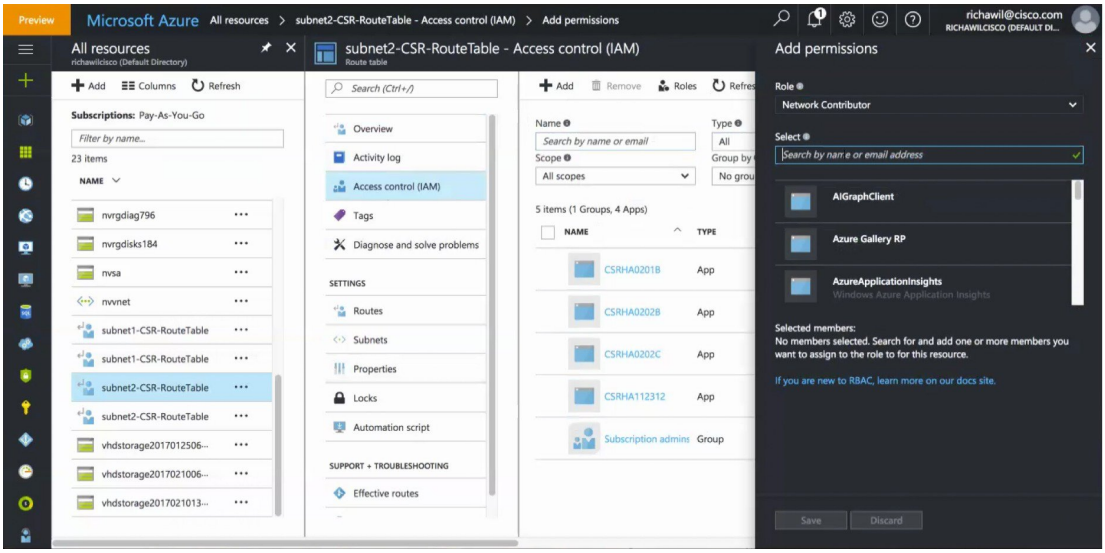
Go to Add an Application under Access Control to a Route Table , on page 35.

# Add an Application under Access Control to a Route Table

This section explains how to configure the route table of a subnet to allow the application (for example, "CSRHA2") to modify the CSR 1000v route table.

**Step 1** To add an application into an existing network, in the **All resources** pane, choose a private side subnet in the left pane;for example, "subnet2-CSR-RouteTable".

**Example:**




**Step 2** In the middle blade, select "Access control (IAM)" and in the right blade, click **+Add**.

**Step 3** In the "Role" textbox, choose **Network contributor**and in the "Assign access to" checkbox, select "Azure AD user, group, or application".

**Step 4** In the "Select" textbox, enter the name of the application.

**Step 5** Click **Save**.

**Step 6** After completing the procedures in this document up to this point, ensure that you have saved the values of the following IDs and keys:

- Tenant ID (For example: `227b0f8f-684d-48fa-9803-c08138b77ae9` )
- App ID (For example: `80848f32-8120-43fb-ba65-3d5aa596cd0c` ).
- API key (For example: `5yOhH593dtD%2FO8gzAlWgulrkWz5dH02d2STk3LDbI4c%3D` ).

The application is now authorized to update the route table.

### What to do next

Next, go to either Configure a Trustpool, on page 38 or Configure a Trustpoint, on page 36 to establish a secure connection between the Microsoft Azure Management API and the Cisco CSR 1000v.

# Configuring High Availability SSL

To establish a secure connection between the Microsoft Azure Management API and the Cisco CSR 1000v perform one of the following two procedures:

## Configure a Trustpoint

To establish a secure connection between the Microsoft Azure Management API and Cisco CSR 1000v, follow this procedure on how to configure an individual trustpoint.

Perform the steps in this procedure or perform the steps in Configure a Trustpool, on page 38 to establish a secure connection between the Microsoft Azure Management API and the Cisco CSR 1000v.

**Note** Information in the steps below, for configuring a trustpoint, has changed compared to what was shown in previous versions of this document. The Microsoft certificates mentioned below need to be used from 7/21/2017 onwards to avoid interruption of the service. At the time of writing further information on certificate changes is found at: https://blogs.technet.microsoft.com/kv/2017/04/20/azure-tls-certificates-changes/?WT.mc_id=azurebg_email_Trans_33716_1407_SSL_Intermediate_Cert_Change. See also: https://www.microsoft.com/pki/mscorp/cps/default.htm.

### Before you begin

Each Cisco CSR 1000v VM is assumed to be configured and running in Microsoft Azure.

Step 3 in the procedure below assumes that you have `openssl` available in the OS that you are using (e.g Linux). For a Windows operating system, which does not include OpenSSL tools, to generate a certificate you can choose from one of the following methods.

- Using `makecert.exe`

- Using IIS Manager - `inetmgr.exe`
- Installing and running OpenSSL tools. For example, see https://wiki.openssl.org/index.php/Binaries

**Step 1** Go to the PKI Repository for Microsoft, which shows the active certificates used by Microsoft authentication servers.

**Step 2** Choose and download the .crt file for a certificate; for example, Microsoft IT TLS CA 2.crt and save this file with a CA certificate name such as msit_tls_ca.crt.

**Step 3** Convert the .crt file into a .pem file as follows:

**openssl** x509 **-in** *crtfile* **-inform der -outform pem -out** *pemfile*

**Example:**

```
openssl x509 -in msit_tls_ca.crt -inform der -outform pem -out msit_tls_ca.pem
```

Open the msit_tls_ca.pem file using a text editor and find the very long sequence of characters between the lines: -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- . Save this sequence of characters, which is the certificate, in a file; for example, cert.txt. (This is used later in step 11.)

**Step 4** **configure terminal**

Enters configuration mode.

**Step 5** **crypto pki trustpoint** *trustpoint-name*.

Enters ca-trustpoint configuration mode. This declares the certificate authority for the Cisco CSR 1000v. *trustpoint-name*- the name of the certificate authority (CA).

**Example:**

```
(config)# crypto pki trustpoint MicrosoftSSL
```

**Step 6** Specify manual cut-and-paste certificate enrollment using the following command: **enrollment terminal** .

**Example:**

```
(ca-trustpoint)# enrollment terminal
```

**Step 7** Specify the subject name in the certificate request using the command: **subject-name cn=** *crtfile*.

*crtfile*- the CA Certificate Name from Step 3.

**Example:**

```
(ca-trustpoint)# subject-name cn=msit_tls_ca.crt
```

**Step 8** **exit**

Exit ca-trustpoint configuration mode.

**Step 9** **crypto pki authenticate** *trustpoint-name*

*trustpoint-name*- the name of the certificate authority (CA) in step 6.

**Step 10** Paste in the certificate text that you previously extracted from the PEM file in step 4. Enter a blank line (Return key). Enter the word `quit` and press the Return key.

You are prompted to paste in the certificate. This certificate text is the text that you had previously saved;for example, in text file "cert.txt".

**Step 11** Enter `yes` to accept the certificate.

**Step 12** **exit**

Exit configuration mode.

**Example:**

(config)# exit

## Configure a Trustpool

The following procedure provides instructions on configuring a trustpool to establish a secure connection between the Microsoft Azure Management API and the Cisco CSR 1000v. A trustpool is a list of certificate authorities (CAs) that has been approved by Cisco as being trustworthy. Perform the steps in this procedure or alternatively, go to Configure a Trustpoint, on page 36 to establish a secure connection between the Microsoft Azure Management API and the Cisco CSR 1000v.

**Before you begin**

Each Cisco CSR 1000v VM is assumed to be configured and running in Microsoft Azure.

**Step 1** Use the `ssh` command to gain access to the Cisco IOS XE CLI on the Cisco CSR 1000v and enter commands in the following steps.

**Step 2** **crypto pki trustpool import** http://www.cisco.com/security/pki/trs/ios.p7b

This command imports certificate authorities from the specified URL.

**Example:**

```
crypto pki trustpool import url http://www.cisco.com/security/pki/trs/ios.p7b

Reading file from http://www.cisco.com/security/pki/trs/ios.p7b Loading
http://www.cisco.com/security/pki/trs/ios.p7b !!!!
% PEM files import succeeded.
```

**Step 3** **show crypto pki trustpool**

Shows the trustpool certificates in a verbose format.

# Configure a Name Server

Configure a name server for the Cisco CSR 1000v. See the IP Addressing: DNS Configuration Guide.

# Configure a Tunnel Between Cisco CSR 1000v Routers

This section describes how to configure a tunnel between Cisco CSR 1000v routers and enable Bi-directional Forwarding Detection (BFD) and a routing protocol (EIGRP or BGP) on the tunnel between the routers for peer failure detection. To authenticate and encrypt IP traffic as it traverses a network, choose between using either an IPSEC tunnel (step 1) or VxLAN GPE tunnel (step 2).

**Step 1** To configure an IPSEC tunnel, enter the configuration mode commands to give the following configuration. (Use either an IPSEC tunnel (step1) or VxLAN tunnel (step 2)). The command **crypto isakmp policy** 1 defines an IKE policy, with a high priority (1), and enters config-isakmp configuration mode.

**Example:**

```
crypto isakmp policy 1
 encr aes 256
 authentication pre-share
crypto isakmp key cisco address 0.0.0.0
!
!
crypto ipsec transform-set uni-perf esp-aes 256 esp-sha-hmac
 mode tunnel
!
!
crypto ipsec profile vti-1
 set security-association lifetime kilobytes disable
 set security-association lifetime seconds 86400
 set transform-set uni-perf
 set pfs group2
!
!
interface Tunnel1
 ip address 192.168.101.1 255.255.255.252
 load-interval 30
 tunnel source GigabitEthernet1
 tunnel mode ipsec ipv4
 tunnel destination 23.96.91.169
 tunnel protection ipsec profile vti-1
 bfd interval 500 min_rx 500 multiplier 3
```

**Note** We recommend that you specify a BFD interval of 500 ms or more. You can increase the BFD timers to account for the varying latency in different regions.

**Step 2** To create a VxLAN GPE tunnel, enter configuration mode commands to give the following configuration. (Use either an IPSEC tunnel (step1) or VxLAN (step 2)).

For further information on configuring a VxLAN GPE tunnel, see: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cether/configuration/xe-16/ce-xe-16-book/vxlan-gpe-tunnel.html

The tunnel destination address must be the public IP address of the corresponding Cisco CSR 1000v. For the tunnel IP address, use any unique IP address. However, the tunnel end points of each redundant Cisco CSR 1000v must be in the same subnet.

**Note** To allow VxLAN to pass traffic through the tunnel, you must ensure that UDP ports 4789 and 4790 are allowed in a Microsoft Azure Network Security Group(NSG). For further information on NSGs, see: https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-nsg.

**Note** We recommend that you specify a BFD interval of 500 ms or more. You can increase the BFD timers to account for the varying latency in different regions.

**Example:**

```
interface Tunnel100
 ip address 192.168.101.1 255.255.255.0
 shutdown
 bfd interval 500 min_rx 500 multiplier 3
 tunnel source GigabitEthernet1
 tunnel mode vxlan-gpe ipv4
```

```
tunnel destination 40.114.93.164
tunnel vxlan vni 10000
```

### What to do next

After configuring either a VxLAN or IPSEC tunnel, you can configure either EIGRP, BGP or OSPF over the tunnel interface. The following section explains how to configure EIGRP: Configuring EIGRP over Virtual Tunnel Interfaces, on page 9.

# Configuring EIGRP over Virtual Tunnel Interfaces

Configure EIGRP over the virtual tunnel interfaces using the following steps.

**Note** Other than using EIGRP, which is the protocol used in the following steps, you also have the option of using either BGP, or OSPF.

### Before you begin

Configure either a VxLAN or IPsec tunnel between the Cisco CSR 1000v routers.

### SUMMARY STEPS

1. **router eigrp** *as-number*
2. **network** *ip-address subnet-mask*
3. **bfd all-interfaces**
4. **end**
5. **show bfd neighbors**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **router eigrp** *as-number*<br>**Example:**<br>`Device(config)# router eigrp 1` | Enables the EIGRP routing process and enters router configuration mode. |
| **Step 2** | **network** *ip-address subnet-mask*<br>**Example:**<br>`network 192.168.101.0 0.0.0.255` | Share the network of the tunnel using EIGRP. |
| **Step 3** | **bfd all-interfaces**<br>**Example:**<br>`Device(config-router)# bfd all-interfaces` | Enables BFD globally on all interfaces that are associated with the EIGRP routing process. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **end**<br>**Example:**<br>`Device(config-router)# end` | Exits router configuration mode and returns the router to privileged EXEC mode. |
| **Step 5** | **show bfd neighbors**<br>**Example:**<br>`Device# show bfd neighbors`<br><br>`IPv4 Sessions`<br>`NeighAddr LD/RD RH/RS State Int`<br>`192.168.101.2 4097/4097 Up Up Tu100` | Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered. |

# Configure Failure Detection for the Cisco CSR 1000v on Microsoft Azure

Follow the steps in this procedure to configure failure detection for the Cisco CSR 1000v and to specify resource identifiers, such as "azure_subscription_id" to Microsoft Azure. Configure a Cisco CSR 1000v to monitor Bidirectional Forwarding Detection (BFD) events using the following steps:

**Step 1** # **redundancy**

Enters redundancy mode. Enter commands in configuration mode, to give a configuration similar to the one shown in the above example.

**Step 2** # **cloud provider azure** *node-id*

*node-id* is a numeric value (in the range 1-255) that identifies an instance of a routing table to be updated in case of a detected failure. A single Cisco CSR 1000v can be used to update multiple routing tables by creating multiple nodes.

**Example:**

```
# cloud provider azure 100
```

**Step 3** # **bfd peer** *peer-ip-address*

*peer-ip-address* is the tunnel IP address of the neighboring Cisco CSR 1000v.

**Example:**

```
# bfd peer 192.168.101.2
```

**Step 4** # **default-gateway ip addr** *ip-addr*

*ip-addr* is the IP address of the Cisco CSR 1000v on the private subnet.

**Example:**

```
# default-gateway ip addr 10.60.1.6
```

**Step 5** # **route-table** *route-table-name*

*route-table-name* is the route table used in Add an Application under Access Control to a Route Table , on page 35.

**Example:**

```
# route-table HaEastRouteTable
```

**Step 6** # **resource-group** *resource_group_name*

*resource_group_name* is the name of the resource group containing the the subnet route table to be updated in the case of a CSR failure. For more information, see https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview.

**Note** This resource group may not be the same resource group that contains other Microsoft Azure resources.

**Example:**

```
# resource group comapnynameusawest
```

**Step 7** # **subscription-id** *azure_subscription_id*

*azure_subscription_id* is the Microsoft Azure subscription ID, which identifies the customer who is responsible for paying the cost of using these Microsoft Azure cloud services.

**Example:**

```
# subscription-id ab2fe6b2-c2bd-44
```

**Step 8** # **tenant-id** *active_directory_tenant_id*

*active_directory_tenant_id* is the Tenant ID saved in Add an Application under Access Control to a Route Table , on page 35).

**Example:**

```
# tenant-id 227b0f8f-684d-48fa-9803-c08138b77ae9
```

**Step 9** # **app-id** *application_id*

*application_id* is the App ID .

**Example:**

```
80848f32-8120-43fb-ba65-3d5aa596cd0c
```

**Step 10** # **app-key** *api-key*

*api-key* is the (URL encoded) API key that you saved in Add an Application under Access Control to a Route Table , on page 35.

**Example:**

```
app-key 5yOhH593dtD%2FO8gzAlWgulrkWz5dH02d2STk3LDbI4c%3D
```

**Step 11** **cidr ip** *ip_network_addr/mask*

*ip_network_addr/mask* identifies an individual route within the route table by its address prefix in CIDR format. This is an optional parameter available in Cisco IOS XE Fuji 16.7 or later. If this parameter is not specified, all the routes in the route table are updated. See the restriction on the use of an “all routes” configuration in Route Table Entry Types, on page 26.

**Example:**

```
cidr ip 15.0.0.0/8
```

---

### Example

This is a summary showing the example configuration commands used in the steps above:

```
redundancy
 cloud provider azure 100
  bfd peer 192.168.101.2
  default-gateway ip 10.60.1.6
  route-table HaEastRouteTable
  cidr ip 15.0.0.0/8
```

```
resource-group companynameusawest
subscription-id ab2fe6b2-c2bd-44
tenant-id 227b0f8f-684d-48fa-9803-c08138b77ae9
app-id 80848f32-8120-43fb-ba65-3d5aa596cd0c
app-key 5yOhH593dtD%2FO8gzAlWgulrkWz5dH02d2STk3LDbI4c%3D
```

# Route Table Entry Types

The route tables in Microsoft Azure support different entry types. The entry type for a route can be one of the following: Virtual network gateway, Internet, or Virtual Appliance. The next hop address identifies a resource in the Azure network.

Routes with an entry type of Virtual network gateway or Internet do not have an explicit IP address for the next hop and are not supported by the High Availability feature.

(Cisco IOS XE Everest 16.6) When you configure High Availability on the Cisco CSR 1000v, all the routes within a route table must have an entry type of Virtual Appliance. These routes require an explicit IP address for the next hop.

(Cisco IOS XE Everest 16.7 or later) When you configure High Availability on the Cisco CSR 1000v, you can specify individual routes to be updated in the case of failure. Ensure that you configure each individual route as having an entry type of Virtual Appliance. If you configure a redundancy node that represents all of the entries in the route table, ensure that all of the routes have an entry type of Virtual Appliance.

# Verify the Configuration of CSR 1000v High Availability

Use the following EXEC mode commands to verify that the Cisco CSR 1000v has been successfully configured for High Availability.

**Step 1** **show crypto pki trustpool**

You can use this command for verification if you imported the trustpool using the configuration command: **crypto pki trustpool import url** *URL* where *URL* is; for example, http://www.cisco.com/security/pki/trs/ios.p7b.

**Step 2** **show crypto pki trustpoint**

You can use this command for verification if you installed an individual trustpoint using this configuration command: **crypto pki trustpoint** *name*.

**Step 3** **show redundancy cloud provider azure** *node_id*

Use this command to check the redundancy configuration.

**Step 4** **show bfd neighbors**

Use this command to verify that neighboring Cisco CSR 1000v routers have established a BFD session.

**Step 5** **show running-configuration**

Use this command to verify that the high availability configuration commands entered in the preceding sections appear in the running configuration.

**Example:**

```
# show running-configuration
```

```
crypto isakmp policy 1
 encr aes 256
 authentication pre-share
crypto isakmp key cisco address 0.0.0.0
!
!
crypto ipsec transform-set uni-perf esp-aes 256 esp-sha-hmac
 mode tunnel
!
!
crypto ipsec profile vti-1
 set security-association lifetime kilobytes disable
 set security-association lifetime seconds 86400
 set transform-set uni-perf
 set pfs group2
!
!
interface Tunnel1
 ip address 192.168.101.1 255.255.255.252
 load-interval 30
 tunnel source GigabitEthernet1
 tunnel mode ipsec ipv4
 tunnel destination 23.96.39.216
tunnel protection ipsec profile vti-1
 bfd interval 500 min_rx 500 multiplier 3
interface GigabitEthernet2
 ip address 10.60.2.6 255.255.255.0
 negotiation auto
 no sh
 no mop enabled
 no mop sysid
```

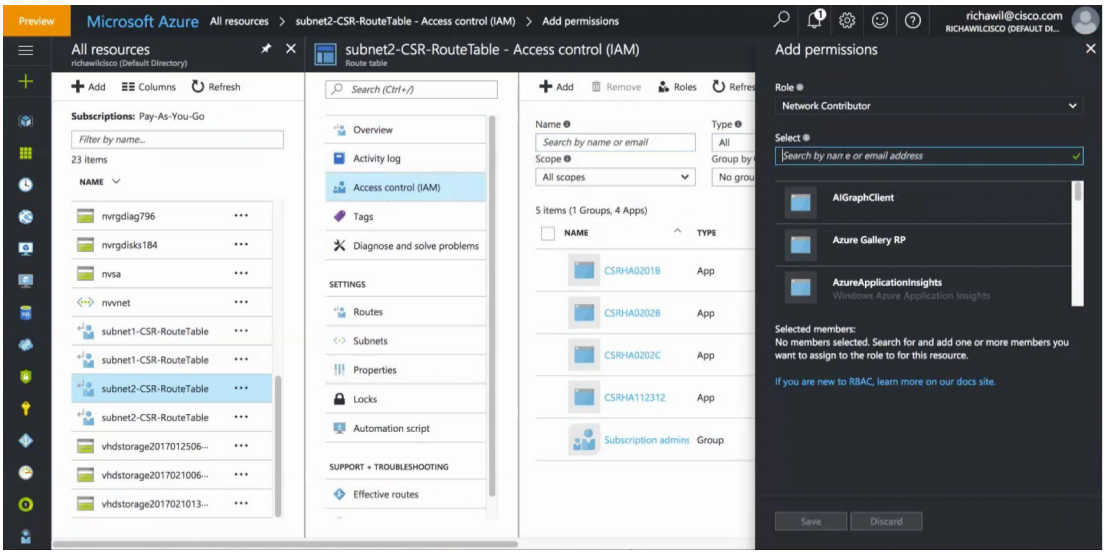
Save the configuration for future use, with the command: **copy running-configuration startup-configuration**.

# Configuring IAM for the Route Table

This section explains how you can configure the route table of a subnet to allow the application (for example, "CSRHA2") to modify the CSR 1000v route table.

**Step 1** To add an application into an existing network, in the **All resources** pane, choose a private side subnet in the left pane;for example, "subnet2-CSR-RouteTable".

**Example:**




**Step 2** Select "Access control (IAM)" and click **+Add**.

**Step 3** In the "Role" textbox, choose **Network contributor**and in the "Assign access to" checkbox, select "Azure AD user, group, or application".

**Step 4** In the "Select" textbox, enter the name of the application.

**Step 5** Click **Save**.

**Step 6** Save the values of the following IDs and keys:

- Tenant ID (For example: `227b0f8f-684d-48fa-9803-c08138b77ae9` )
- App ID (For example: `80848f32-8120-43fb-ba65-3d5aa596cd0c`).
- API key (For example: `5yOhH593dtD%2FO8gzAlWgulrkWz5dH02d2STk3LDbI4c%3D`).

# Troubleshooting High Availability Issues

If you face any issues when you configure High Availability for your CSR1000v instances, execute the debug_ha.sh script in the `~/azure/HA/` directory. This script gathers all the important logs regarding High Availability and compresses them into a single tar file named ha_debug.tar.gz on the bootflash. Send this .gz file to the Cisco TAC team to get help in resolving your issue.