Configure High Availability

High Availability refers to the ability to establish redundancy of networking functionality and configuration data between two peer routers. This chapter provides overview information on high availability, and how you can configure high availability on a Cisco CSR 1000v instance running on different cloud service providers.

- Overview of High Availability Version 3, on page 1
- Configure High Availability Version 3, on page 6
- Configure High Availability for CSR 1000v Running on Azure, on page 14
- Configure High Availability on CSR Running on Amazon Web Services, on page 25
- Configure High Availability in CSR 1000v Running On Google Cloud Platform, on page 29
- Example Configurations, on page 34
- Verify High Availability, on page 35
- Troubleshoot High Availability Issues, on page 35

Overview of High Availability Version 3

Feature History Table

Release	Description
Cisco IOS XE Gibraltar 16.11.1	High availability version 3 introduced.
Cisco IOS XE Amsterdam 17.3.1	The `pip3 install csr-[aws/azure/gcp]-ha --user` script introduced to install high availability version 3.

The High Availability feature is supported for Cisco CSR 1000V Routers running on Microsoft Azure, Google Cloud Platform (GCP), and Amazon Web Services (AWS). A typical use case for the Cisco CSR 1000V is to interconnect two subnets within a virtual network. You can deploy Cisco CSR 1000V routers between the front-end (public) and the back-end (private) subnets. The Cisco CSR 1000V router represents a single point of failure for access to back-end resources. To mitigate this single point of failure, you must deploy two Cisco CSR 1000V routers between the two subnets.

The back-end subnet contains a routing table with entries pointing to the next hop router, which is one of the two Cisco CSR 1000V instances. The peer Cisco CSR 1000V routers communicate with one another over a tunnel using the Bi-directional Forwarding Detection (BFD) protocol. If the connection is lost between a

router and a peer, BFD generates an event. This event causes the active router that is working to update the entries in the route table so that the routing table points to the default route.

The routing table controls the upstream traffic of the Cisco CSR 1000V router and the routing protocol configured on the router determines the path of the downstream traffic.

In cloud environments, it is common for virtual networks to implement a simplistic mechanism for routing, which is based on a centralized route table. However, you can also create multiple route tables, where each route table has a subnet assigned. This subnet acts as the source of route information, and the route table is populated automatically which includes one or more individual routes depending on the network topology. You can also configure the routes in the route table.

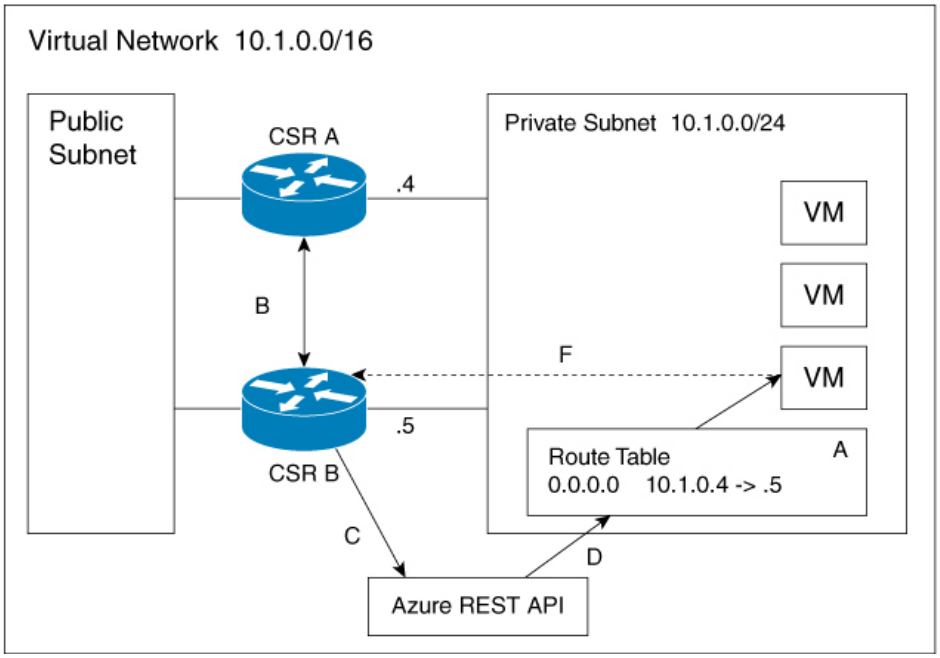
A subnet has a centralized route table, which allows two Cisco CSR 1000V routers to operate in a redundant mode. You can deploy two Cisco CSR 1000V routers in the same virtual network with their interfaces directly connected to subnets in the virtual network. You can add routes to the route table to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

The Cisco CSR 1000V router uses the Bi-directional Failure Detection (BFD) protocol to detect whether a peer router is operating properly. An IP tunnel is created between the two peer routers and each router periodically sends a BFD protocol message to the other router. If one router fails to receive a BFD message from the peer for a specific period, the active router concludes that 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 router uses the programmatic API to update the route table entries.

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.

Figure 1: High Availability Version 3



The subnet on the right has an address block of 10.1.0.0/24. The two Cisco CSR 1000V routers that are connected to this subnet provide a redundant path for traffic leaving this leaf subnet. The subnet is associated with a route table which provides the route information to the virtual machines attached to the subnet.

Consider this scenario: Initially the default route in the route table has the IP address of the next hop router - 10.1.0.4 (CSR A). All the traffic leaving the subnet goes through CSR A. CSR A is currently the active router for the default route. When CSR A fails, CSR B detects the failure as this router stops receiving BFD protocol messages from CSR A. CSR B writes to the route table via a RESTAPI 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 then becomes the active router for the route to the 10.1.0.0 network.

Step	Description
A	CSR A with address 10.1.0.4 is the active router for the 10.1.0.0 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 10.1.0.0 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.

Topologies Supported

1-for-1 redundancy topology: If both the Cisco CSR 1000v routers have a direct connection to the same subnet, the routers provide a 1-for-1 redundancy. An example of 1-for-1 redundancy is shown in the preceding figure. All the traffic that is intended for a Cisco CSR 1000v only goes to one of the routers - the Cisco CSR 1000v that is currently active. The active Cisco CSR 1000v router is the next-hop router for a subnet. The other Cisco CSR 1000v 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. Instead of router B being the next hop, the route entry is changed to router A as the next hop.

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. To configure a redundancy node, you require the following information:

- Route Table – The identity of the route table in the cloud. Route table includes a region or group in which the table was created, an identifier for the creator or the owner of the table, and a name or identifier for

the specific table. Optionally, you can specify an individual route within the table. If you do not specify an individual route, the redundancy node represents all the routes in the table.

- Credentials - Authentication of the identity of the Cisco CSR 1000v router. Each cloud provider handles the process of obtaining and specifying the credentials differently.
- Next Hop - The next hop address that is written to the route entry when a trigger event occurs. Next Hop is usually the interface of the CSR 1000v routers on the subnet that is protected.
- Peer Router - Identifies the redundant router that will forward 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 you do not specify this value, 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 under redundancy for cloud high availability.
- Revert to Primary Router: After a router recovers from a failure, the *Revert to Primary Router* event occurs. The purpose of this event is to ensure that the primary router for the route is re-established as the active router. This event is triggered by a timer and you need not configure this event. In the route table entry, the event handler changes the next hop address that is defined in the redundancy node only if it is different from the next hop address that is currently set for the route.

  This Revert to Primary Router event is generated periodically using a CRON job in the guestshell environment. The job is scheduled to run every 5 minutes and checks if each redundancy node that is configured in the primary mode has this router's next hop interface set in the route table. If the route table entry already points to this router's next hop interface, then an update is not required. If a redundancy node configuration of the mode parameter is secondary, then the *Revert to Primary Router* event is ignored.
- Redundancy Node Verification: 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. This event is not generated automatically or algorithmically. This event verifies the ability of the event handler to execute its functions. Execute a script, manually or programmatically, to trigger the Redundancy Node verification event. For further information about the verification event, see *User-Defined Triggers*, in the *Advanced Programming for High Availability on Microsoft Azure* section.

High Availability Versions and OS Compatibility

Choose one of the following deployment options for High Availability on Cisco IOS XE Fuji 16.11.x and later:

- **High Availability Version 1**: This version is supported until Cisco IOS XE 16.11.1 release. From Cisco IOS XE 16.11.x, if you attempt to configure redundancy nodes in Cisco IOS, you receive a warning that the configuration is deprecated.

- **High Availability Version 2 with Redundancy Node Configuration in Cisco IOS XE**: You can configure High Availability Version 2 for Cisco CSR1000V 16.11.1 or later running on Microsoft Azure using CLI commands. This version provides access to the new features in HA version 2 and allows you to continue using your existing redundancy node configurations. However, this deployment option is deprecated from Cisco IOS XE 16.12.1.
- **High Availability Versions 2 and 3 with Redundancy Node Configuration in the guestshell**: You can configure high availability version 2 using guestshell-based Python scripts from Cisco IOS XE 16.9.1 release. The high availability version 3 is available from Cisco IOS XE 16.11.1 release, and this HA version is the recommended version.

  If you currently use redundancy node configuration in Cisco IOS XE, we recommend that you use the Redundancy Node configuration in the guestshell using the Cisco IOS XE Fuji 16.11.1 release.

Differences in High Availability across Cisco IOS XE Releases

The following table specifies the functionalities that are supported across the high availability versions.

Feature/Functionality	Suport in High Availability Version 1	Suport in High Availability Version 2	Suport in High Availability Version 3
Redundancy Node Configuration in IOS XE	Yes	Yes	Yes
Redundancy Node Configuration in Guestshell	No	Yes	Yes
Revert to the primary router after recovery	No	No	Yes
CSR1000V authentication by an application in Azure Active Directory	Yes	Yes	Yes
CSR1000V authentication by Managed Identity (formerly known as Managed Service Identity)	No	Yes	Yes

By default, in Cisco IOS XE Gibraltar 16.11.x, the Cisco CSR 1000v on AWS runs HA Version 2. To run HA Version 3, you must manually install and enable guestshell, and install the `csr_aws_ha` Python package in guestshell.

What's New in High Availability Version 3

The first version of high availability in the AWS cloud was introduced in Cisco IOS XE 16.3.1. The second version of high availability or HA Version 2 was released in Cisco IOS XE Fuji 16.5.1.

HA Version 3 is released in Cisco IOS XE Gibraltar 16.11.1. This high availability version supports several new features, a new configuration, and a deployment mechanism. Here's an overview of what's new in high availability version 3:

- **Cloud Agnostic**: This version of high availability is functional on CSR 1000v routers running on any cloud service provider. While there are some differences in the cloud terminology and parameters, the set of functions and scripts used to configure, control, and show the high availability features are common across the different cloud service providers. High Availability Version 3 (HAv3) is supported in CSR 1000v routers running on AWS, Azure, and GCP. Support for the GCP provider has been added in 16.11.1. Check with Cisco for current support of high availability in the individual provider's clouds.
- **Active/active operation**: You can configure both Cisco CSR 1000v routers 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 as the secondary router. To enable load sharing, take all the routes and split them between the two Cisco CSR 1000v routers. Note that this functionality is new for AWS-based clouds.
- **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 this 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. This functionality is also new for the AWS-based clouds.
- **User-supplied Scripts**: The guestshell is a container in which you can deploy your own scripts. HAv3 exposes a programming interface to user-supplied scripts. This implies that you can now write scripts that can trigger both failover and reversion events. You can also develop your own algorithms and triggers to control which Cisco CSR 1000v provides the forwarding services for a given route. This functionality is new for AWS-based clouds.
- **New Configuration and Deployment Mechanism**: The implementation of HA has been moved out of the Cisco IOS XE code. High availability code now runs in the guestshell container. For further information on guestshell, see the "Guest Shell" section in the Programmability Configuration Guide. In HAv3, the configuration of redundancy nodes is performed in the guestshell using a set of Python scripts. This feature has now been introduced for AWS-based clouds.

Configure High Availability Version 3

The following sections specify the common configuration steps to configure High Availability Version 3 for a CSR 1000v running on any cloud service provider.

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 do not need to configure these prerequisites as they are included automatically in the startup-config file.

Step 1 Perform the following configuration:

Example:

```
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
```

Step 2 To configure High Availability, you must verify whether IOX is configured and running:

Example:

```
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
```

Step 3 Enter the following command to verify that the guest application is defined and running:

Example:

```
show app-hosting list
App id State
------------------------------------------------------
guestshell RUNNING
```

If the state of the guestshell displays DEPLOYED in the output of the preceding command, you must enable the guestshell by 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
```

Configure a Tunnel Between Cisco CSR 1000v Routers

You must configure a tunnel between the Cisco CSR 1000v routers and enable Bi-directional Forwarding Detection (BFD) and a routing protocol (EIGRP or BGP) on the tunnel for peer failure detection. To authenticate and encrypt IP traffic as it traverses a network, either use an IPsec tunnel or VxLAN GPE tunnel.

Step 1 To configure an IPsec tunnel, enter the configuration mode commands to give the following configuration. 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 100 min_rx 100 multiplier 3
```

Step 2 To create a VxLAN GPE tunnel, enter the following configuration

```
interface Tunnel100
ip address 192.168.101.1 255.255.255.0
bfd interval 100 min_rx 100 multiplier 3 tunnel source GigabitEthernet1
tunnel mode vxlan-gpe ipv4 tunnel destination 40.114.93.164
tunnel vxlan vni 10000
```

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 endpoints 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 the cloud's network security group. See the cloud provider's documentation for configuring network security filters.

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 that is 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.

Step 1 **router eigrp** *as-number*

Example:

```
Device(config)# router eigrp 1
```

Enables the EIGRP routing process and enters the router configuration mode.

Step 2 **network** *ip-address subnet-mask*

Share the network of the tunnel using EIGRP.

Example:

```
network 192.168.101.0 0.0.0.255
```

Step 3 **bfd all-interfaces**

Enables BFD globally on all the interfaces that are associated with the EIGRP routing process.

Example:

```
Device(config-router)# bfd all-interfaces
```

Step 4 **end**

Exits the router configuration mode and returns the router to the privileged EXEC mode.

Example:

```
Device(config-router)# end
```

Step 5 **show bfd neighbors**

Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered.

Example:

```
Device# show bfd neighbors

IPv4 Sessions
NeighAddr      LD/RD        RH/RS    State  Int
192.168.101.2  4097/4097    Up       Up     Tu100
```

Verify the Tunnel Surface

To verify that the tunnel interface is configured and enabled, run the `show ip interface brief` command.

Example:

```
# show ip interface brief
                       IP-Address       OK? Protocol   Method   Status
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 DHCP  up up
```

Configure the BFD Peer Router

Run the following command:

Example:

```
redundancy
cloud-ha bfd peer <peer_router_ip_address>
```

This configuration command identifies the peer router. The IP address is that of the peer Cisco CSR 1000v within the tunnel carrying the BFD protocol between the two CSR 1000v routers.

Install the High Availability Package

Step 1 Execute the `#Router>guestshell` command to enter the guestshell.

Step 2 Install the appropriate Python package based on the cloud provider on which the CSR 1000v instance is running:

Cloud Provider	Package Name
Microsoft Azure	csr_azure_ha
Amazon Web Services	csr_aws_ha
Google Cloud Platform	csr_gcp_ha

Note The package name for Microsoft Azure is the same for both HAv2 and HAv3. If you perform an install by executing the **pip install csr_azure_ha –-user** command, the latest HA V3 is downloaded.

Step 3 Install the package that is appropriate for your cloud service provider by using the **[guestshell@guestshell]$ pip install <package_name> --user** command.

Note If you are a user who has newly installed Cisco CSR1000V Release 17.3.1, you must use **pip3 install csr-[aws]-ha --user** to install high availability version 3. If you are upgrading from 17.2 to 17.3.1, and you have already enabled the Guestshell, you need not use the latest python script.

Step 4 From the home directory, navigate to the subdirectory named `cloud:[guestshell@guestshell]$ cd cloud`.

Verify High Availability in Guestshell

Execute the `[guestshell@guestshell cloud]$ systemctl status csr_ha` command.

Example:

```
[guestshell@guestshell cloud]$ systemctl status csr_ha
azure-ha.service - Azure High Availability service
Loaded: loaded (/etc/systemd/user/csr_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/csr_ha.service
├─ 29 python
/home/guestshell/.local/lib/python2.7/site-packages/csr__ha/server/ha_serve...
└─103 python
/home/guestshell/.local/lib/python2.7/site-packages/csr_ha/server/ha_serve...
```

Note If the service is not running, start the service by running the `[guestshell@guestshell azure]$ sudo systemctl start csr_ha` command.

Set the Path Environment Variable

Step 1 Update the path environment in the guest shell to specify the location of the configuration scripts.

Step 2 Execute the `source ~/.bashrc` command in the guestshell.

Configure the Redundancy Nodes

You can use Python scripts to create and modify redundancy nodes. Python scripts use the parameters that are shown in the following table. These parameters are described here in general for edification. To see the actual parameters, see the "Configure Redundancy Nodes" section specific for each cloud provider section.

Name of the parameter	Is this parameter required?	Description
Node Index	Yes	The index that is used to uniquely identify this node. Valid values: 1–1023
Cloud Provider	Yes	Specifies the specific cloud offer from the cloud provider.
User Identifier	Yes	A cloud-specific parameter which identifies a user or subscriber to the cloud service.
Scope	Yes	Some cloud providers offer a mechanism to bundle resources in a logical container. This parameter is used to identify this container.
Route table	Yes	The name or identifier of the route table to be updated.
Route	No	IP address of the route to be updated in CIDR format. Can be IPv4 or IPv6 address. If a route is unspecified, then the redundancy node is considered to apply to all valid routes in the routing table. Different clouds may have restrictions on what constitutes a valid route.

Name of the parameter	Is this parameter required?	Description
Next hop address	Yes	Interface or 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 interface with an IPv4 or IPv6 address.
Mode	No	Indicates whether this router is the primary or secondary router for servicing this route. Default value is secondary.

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
```

Here, -i specifies the index of the redundancy node (1-1023).

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

Delete a Node

Execute the `delete_node.py{ -i value }` command to delete an existing redundancy node.

Example:

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

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

The -i parameter is mandatory. 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 the system displays the parameters of the specified node. If the script is unsuccessful, a non-zero value is returned. An error message is written to the log file.

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

Recover deleted virtual disk

Step 1 Enable the guestshell.

Step 2 Install the appropriate HA Python package for your cloud provider.

Step 3 Run the Python scripts to reconfigure all the redundancy nodes. You can reapply the node configuration by using the `create_node` script.

Configuring High Availability in the Cloud Provider Network

To complete the configuration for the high availability service, you must also execute configuration changes in the cloud provider's network. This section discusses these configuration requirements in general. The details of these steps are specific to each cloud provider. For specific configuration steps, see the appropriate cloud provider sections configuration sections.

- **Authenticate the CSR 1000v**: For the CSR 1000 router to access any resource in the cloud, you must first provide the proof of its identity. That is, you must authenticate the router.
- **Grant Access to the Route Table**: You must authorize the router to read and write to the route table. This usually involves configuring some form of identity and access management on the route table itself.
- **Network Security**: Cloud providers usually support a mechanism for filtering traffic which can pass to and from various network resources such as subnets and network interfaces. To enable the exchange of the BFD protocol messages between the pair of peer routers, it is necessary for the network security group that is associated with the CSR interfaces hosting the tunnel to allow ports 4789 and 4790 to be passed.
- **Verify a Redundancy Node**: If all the above configuration is complete, you must verify that a route that is represented by a redundancy node is successfully updated by the CSR 1000v router. You can verify this by simulating a peer failure for this redundancy node. This is a diagnostic tool that verifies whether the CSR 1000v router is capable of reading and writing the route table that is specified by the redundancy node.

Trigger the failover Using EEM

You can configure your Cisco CSR1000V instance with an Embedded Event Manager (EEM) applet which can detect the transition of an interface state and trigger a failover of the Cisco CSR1000V instance.

To trigger the failover, enter configuration mode in the Cisco IOS XE CLI of your Cisco CSR 1000V instance. Run the following commands:

```
event manager applet Interface_GigabitEthernet2
event syslog pattern "Interface GigabitEthernet2, changed state to administratively down"
action 1 cli command "enable"
action 2 cli command "guestshell run node-event.py -i 10 -e peerFail" exit
exit
```

Note The above-mentioned EEM script is only a generic example. You must modify the script to suit your particular environment and to trigger on an event you specify. For example, you can modify the script to trigger on BFD messages or a line protocol state.

Set user-defined Triggers

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

As a sample scenario, to process the redundancy node and update an associated route table entry, run the `node_event` Python script.

For example, `node_event.py –i node_index -e peerFail` processes the redundancy node, and updates the associated route table entry.

Configure High Availability for CSR 1000v Running on Azure

The first version of high availability in the Azure cloud was introduced in Cisco IOS XE Everest 16.5.1. The second version of high availability, or HA Version 2, was released in Cisco IOS XE Fuji 16.9.1. High Availability Version 3 is supported on Cisco IOS XE Gibraltar 16.11.1 release and later. The configuration of the BFD peer router is simplified in HAv3.

Create Binding to BFD Peer

When you configure High Availability Version2 with IOS XE releases 16.9.x to 16.11.x, you can create a binding to a BFD peer by executing the following command:

Example:

```
redundancy
cloud provider azure index
bfd peer <peerIpAddress>
```

When you configure either High Availability Version 2 or Version 3 with IOS XE releases 16.12.1 and later, you can create a binding to a BFD peer by executing the following command:

```
redundancy
cloud-ha bfd peer <peerIpAddress>
```

Configure Cloud Specific Redundancy Parameters

The following table specifies the redundancy parameters that are specific to Microsoft Azure:

Parameter Switch	Switch	Description
Node Index	-i	The 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	The Azure subscription id.
Resource Group Name	-g	The name of the resource group that contains the route table.
Route Table Name	-t	The 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. 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	The 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.

Create a Redundancy Node

Run the following script to create a redundancy node and add it to the database: `create_node { switch value } […[{ switch value }].`

You must configure the following parameters for a valid redundancy node:

- Node Index
- Cloud Provider
- Subscription ID
- Resource Group Name
- Route Table Name

```
create_node -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 the configuration is successful, the script returns a value of zero.

Set Redundancy Node Parameters

To change the value of parameters in an existing redundancy node, run the following script: `set_params{ 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 mandatory. This command sets the values of the specified parameters. If the specified parameter is already defined for the redundancy node, the value of the parameter is updated.

```
set_params -i 10 -n 192.168.7.5 -m primary
```

In this example, the next hop address and mode will be updated for the redundancy node with index 10.

If this configuration is successful, the script returns a value of zero.

Clear Redundancy Node Parameters

If you want to clear the value of specified parameters for an existing redundancy node, run the following script: `clear_params -i value { switch } […[{ switch }].`

Example:

```
clear_params -i 10 -r -n
```

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

Specify only the switch parameter when you clear an associated value. Do not include the current value of the parameter.

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

If the clearing is successful, the script returns a value of zero.

Authenticate the CSR 1000v

To update a routing table in the Azure network, you must first authenticate the CSR 1000v router. This is accomplished by creating an application which represents the CSR 1000v router in the Azure Active Directory. You can use the application that is granted permissions, to access the Azure network resources.

You can create the application by using the following two mechanisms:

- System-assigned managed identity - Azure automatically creates an application and binds it to the router. This mechanism was previously called as Managed Service Identity by Azure.
- Manual application registration in Azure Active Directory - Here, the user creates an application in the Azure Active Directory, which represents the CSR 1000v router.

You can manually create a managed identity in Azure Active Directory by creating an application which represents the router. The application is assigned a set of identifiers; tenant ID, application ID, and application key. These application identifiers must be configured in the high availability feature either as the default AAD application or within an individual redundancy node.

Alternatively, when you create the CSR 1000v, you can configure Azure to create a system-assigned managed identity for the CSR 1000v instance. In this case, you need not configure any application identifiers in the high availability feature. That is, in the absence of the configuration of an application's tenant ID, application ID, and application key, the high availability feature assumes that the CSR 1000v router is using a system-assigned managed identity.

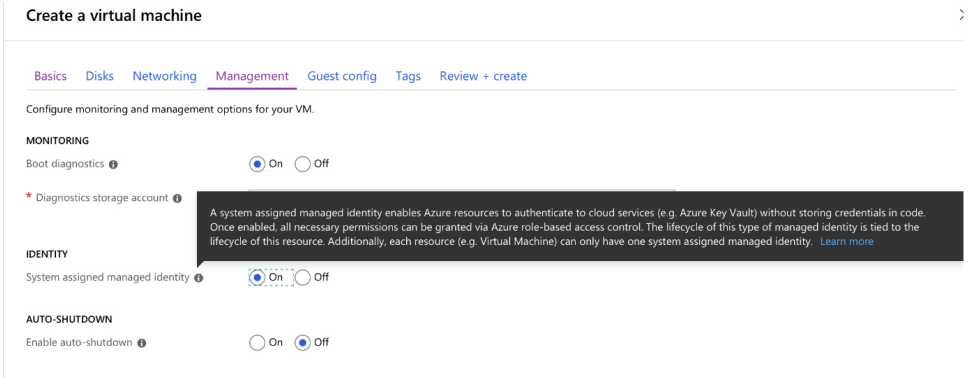
System Assigned Managed Identity

When you create the CSR 1000v router, you can enable for it to be assigned a system managed identity by Azure. There are two ways in which you can create a CSR 1000c router from the Azure marketplace:

- Solution template – A CSR 1000v router is created along with other Azure resources to create a networking solution in a single step.
- Standalone – A standalone CSR 1000v router is created, usually within an existing virtual network, with the base CSR image.

If you create a CSR 1000v router by using one of the solution template offerings in the Azure marketplace, a system-assigned managed identify for the CSR 1000v is enabled by default. If you create a standalone CSR 1000v by using a base CSR image, then a system-managed identity is enabled as shown in the following image:

Figure 2: Enable System Managed Identity



Authentication Using Azure Active Directory Service Principal

This section explains how to create an application in a Microsoft Azure Active Directory with permissions to access Microsoft Azure Resource Manager APIs.

Step 1 See the latest instructions on registering an application with Azure Active Directory in Microsoft Azure documentation. See also: https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-v1-add-azure-ad-app.

Step 2 Go to the portal for Microsoft Azure by visiting https://portal.azure.com.

Step 3 Choose your account name and sign in using your Microsoft Azure password.

Step 4 In the left navigation, click **Azure Active Directory** and select an Active Directory in the main pane. Click **Switch Directory** at the top of the pane to select the active directory.

Step 5 Verify whether you are authorized to create a new application. See 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 6 Navigate to the Active Directory that you want to use.

Step 7 To create a new application, select **Create** > **New Application Registration**.

Step 8 Specify the name of the application and ensure that **Web App / API** is selected as the Application type

Step 9 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. 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 is the sign-on URL: `http://mydir.onmicrosoft.com/myapp`.

Step 10 Click **Create**.

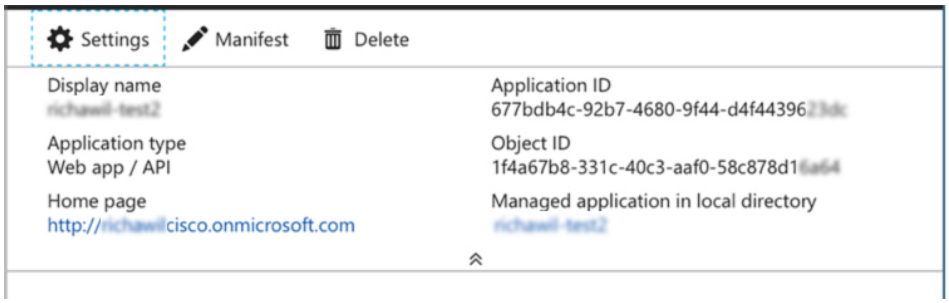
Step 11 Navigate to the Azure Active Directory page. Search for the application that you created. Make a note of the assigned Application ID.

Obtain the Application ID and Tenant ID

Before you begin

Create an application in the Microsoft Azure Active Directory.

Step 1 After you create the application, the registered app should appear on the screen as shown in the following image:



Step 2 Use the portal to create an Azure Active Directory application and service principal that can access resources. Make a note of the Application ID. See step 2 in the *Get application ID and authentication key* section in the Microsoft Documentation.

Step 3 Select **Azure Active Directory**.

Step 4 Select **Properties**. Make a note of the value in the **Directory ID** field. This is your tenant ID.

Create an Authentication key for the Application

Step 1 From the Microsoft Azure portal, select the **Azure Active Directory**.

Step 2 Select **App Registrations**.

Step 3 Select the application that you previously created in the *Obtain the Application ID and Tenant ID* section.

Step 4 Click **Settings**.

Step 5 To create a key for API access, select **Keys** and specify a value for **Duration**. Duration is the length of time after which the key becomes invalid.

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

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

Step 7 You must convert the API key to URL unencoded format. To find a suitable conversion tool, enter URL encoder into an Internet search engine. You might need the unencoded API key for procedures such as *Configure Failure Detection for the Cisco CSR 1000v on Microsoft Azure*.

Example:

```
URL encoded API Key:   5yOhH593dtD%2FO8gzAlWgulrkWz5dH02d2STk3LdbI4c%3D
         URL unencoded API Key:  5yOhH593dtD/O8gzAlWgulrkWz5dH02d2STk3LdbI4c=
```

Manage Azure Active Directory Applications in Guestshell

There are a set of utility scripts that can be run in the guestshell environment to manage applications in the Azure Active Directory, whether they were created manually as user-assigned identities or system-assigned identities. The following sections describe the use of these scripts and how to configure the binding between a redundancy node and the application used to authenticate the CSR 1000v router.

- Managing user-defined applications: If you have chosen to use a user-assigned identity for the CSR 1000v router, the application that was created in Azure Active Directory must be configured in the high availability feature. The application can be configured as the default application used for all the redundancy nodes, or for individual redundancy nodes.
- Set the default application: If you configure a user-assigned application as the default application using the `set_default_aad_app` script, all the redundancy nodes use the specified application for authentication, unless a redundancy node has an individual application configured.

Set the Default Application

Set the default application by running the `set_default_aad_app.py{ switch value } [...[{ switch value }]` script. See the following table for the AAD Redundancy Node Parameters:

Parameter Name	Switch	Description
Cloud Provider	-p	Specifies which Azure cloud is in use {azure | azusgov | azchina}
Tenant ID	-d	Identifies the AAD instance.
Application ID	-a	Identifies the application in AAD.
Application Key	-k	Access key that is created for the application. Key should be specified in unencoded URL format.

```
[guestshell@guestshell]$ set_default_aad_app.py -p azure -d
c4426c0b-036f-4bfb-b2d4-5c910c5389d6 -a 3d6e2ef4-8160-4092-911d-53c8f68ba808 -k
hZFvMGfzJuwFiukez27e/duyztom1bj7QL0Yix+KY9c=
```

```
[guestshell@guestshell]$ set_default_aad_app.py -h
usage: set_default_aad_app.py [-h] -p {azure,azusgov,azchina} -a A -d D -k K
AAD Application
required arguments:
  -p {azure,azusgov,azchina} <cloud_provider> {azure | azusgov | azchina}
  -a A                    to add the applicationId
  -d D                    to add the tenantId
  -k K                    to add the applicationKey
```

Clear the Default Application

You can clear the default user-assigned application configuration by using the `clear_default_aad_app` script.

```
[guestshell@guestshell]$ clear_default_aad_app.py
```

Clear the Application List

If you create a user-assigned application and associate the application with individual redundancy nodes, information about these applications is cached in memory. You can display the list of known applications by using the `show_auth_applications.py` script. Clear the cache using the clear_aad_application_list script.

```
[guestshell@guestshell]$ clear_aad_application_list.py
```

Managing all Applications

Use the following scripts to manage all the applications - user-assigned or system-assigned.

Showing Authentication Applications

The CSR 1000v router maintains a list of configured applications. You can view this list by using the show_auth_applications script.

```
[guestshell@guestshell]$ show_auth_applications.py
```

Clearing the Authentication Token

When an event is triggered on a redundancy node, the CSR 1000v router uses the configured application to obtain an authentication token from the Azure network. This token is cached up to five minutes in the router. You can clear the cached token by using the `clear_token` script.

This script clears either the default user-assigned application or the system-assigned application. The script does not clear the token on any user assigned application which is explicitly configured on an individual redundancy node.

```
[guestshell@guestshell]$ clear_token.py
```

Refreshing the Authentication Token

The CSR 1000v router can be forced to obtain a new token for the active application by using the `refresh_token` script.

This script refreshes either the default user-assigned application or the system-assigned application. This script does not refresh the token on any user-assigned application which is explicitly configured on an individual redundancy node.

```
[guestshell@guestshell]$ refresh_token.py
```

Select the Authentication Application

You can choose either system-assigned or user-assigned applications to identify a CSR 1000v router for the purpose of authentication. You can use the same mechanism for all the applications within a single CSR 1000v router. You can also have multiple user-assigned applications across multiple redundancy nodes.

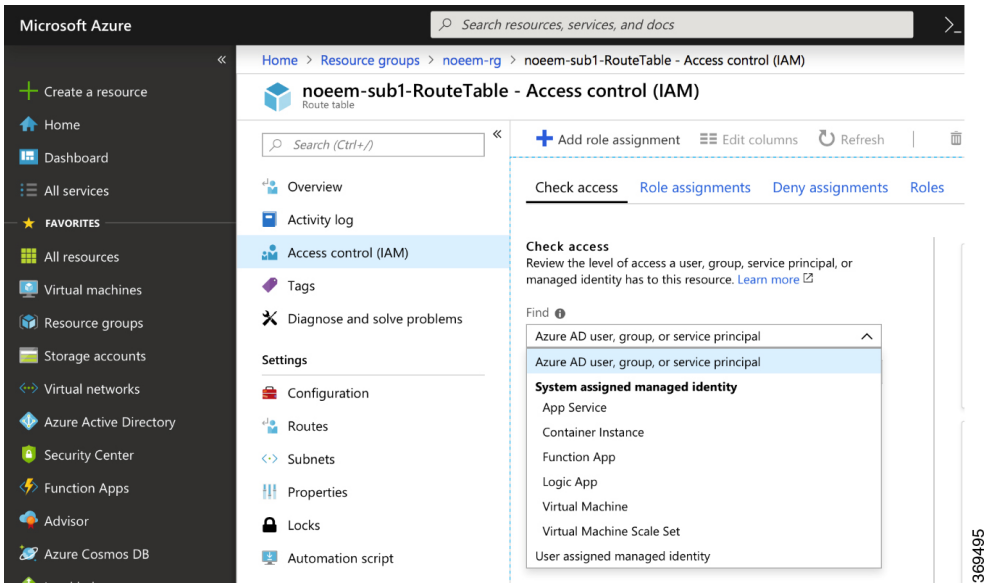
The following table summarizes which application is used by the CSR 1000v router when processing a redundancy node:

Table 1:

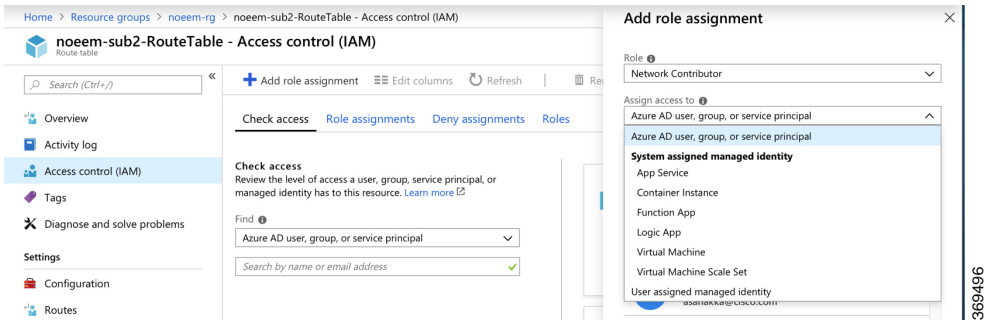
Is A Default Application Configured?	Does Node Have a User Assigned Application Configured?	Will CSR Use This Application?
No	No	System assigned application
No	Yes	User assigned application configured on this redundancy node
Yes	No	User assigned application configured as the default by set_default_aad_app.py
Yes	No	User assigned application configured on this redundancy node

Configuring IAM for the Route Table

Step 1 To add an application into an existing network, in the **All Resources** pane, choose a private side subnet from the left pane. For example, *noeem-sub1-RouteTable*.



Step 2 In the center pane, select **Access control (IAM)**. Select the plus icon to add a role assignment.



Step 3 In the Add Role Assignment screen, set the **Role to Network Contributor**.

Step 4 Select the **Assign Access to Pulldown** menu. If you are using system-assigned managed identity, select the **Virtual Machine** sub option and go to Step 6. If you are using user-assigned managed identity, select the option and go to step 5.

Step 5 In the **Select** field, enter the name of the user-assigned application that you created in **Azure Active Directory**. Click **Save**.

Step 6 In the **Select** field, enter the name given to the CSR 1000v instance. If you have configured the CSR 1000v instance properly for system-assigned identity, the CSR 1000v instance appears in the search results.

Step 7 Select the CSR 1000v instance by name, and click **Save**.

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 the entries in the route table, ensure that all the routes have an entry type of Virtual Appliance.

Configuring the Network Security Group

If you have a network security group attached to NIC0 of the router, you must allow the BFD protocol to pass the interface. Configure an inbound and outbound security rule that allows ports 4789 and 4790 to be passed.

Configuring the Console Timeout

When you start an SSH session to the Cisco CSR 1000v router, ensure that you do not configure the terminal VTY timeout as infinite. That is, 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 exec-timeout 0 0 command causes an issue as Azure enforces a timeout for the console idle period of 4 to 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, the maximum number of simultaneous terminal sessions allowed 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
```

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

Migrate from High Availability Version 1 to Version 3

In high availability version 3, configuration of redundancy nodes is no longer supported via the IOS configuration command line interface. Attempts to configure a redundancy node via IOS will result in a

warning message indicating such operations are deprecated. To migrate from high availability version 1 you must convert the IOS configuration to the format supported in guestshell.

If you are performing an in-place upgrade or a binary upgrade to the IOSXE release 16.11, then the configuration of redundancy nodes in IOS will be automatically copied to guestshell by performing the following steps:

If you are running the guestshell in the current IOS release, copy all important files out of the guestshell, as they will be erased as part of this upgrade.

Step 1 In IOS executive mode, run the guestshell destroy command.

Step 2 Perform the binary upgrade to IOS XE release 16.11. Wait for the CSR 1000v router to reboot and stabilize.

Step 3 Enable the guestshell and enter the shell.

Step 4 Install the CSR HA package for Azure, version 3. For more information, see Installing the CSR HA Package.

Step 5 In IOS, configure the binding to the BFD peer router:

```
conf t
redundancy
cloud-ha bfd peer <peer-ip-addr>
end
```

This launches a process in the IOS to translate the configuration of redundancy nodes from IOS to guestshell. This process can take several minutes to complete, as each node is verified before the node is transferred.

Step 6 Wait for the transfer to complete. As each node is transferred, its configuration is added to a file in guestshell `~/cloud/HA/node_file`. When the size of this file stabilizes, the transfer is done. Open this file in the guestshell and examine its contents. Execute the script `cat node_file`.

Step 7 Verify whether all the nodes and their parameters have been successfully transferred. Use the create_node and the set_params scripts to make any adjustments.

Step 8 Restore any files copied out of the guestshell in step 1.

Step 9 Delete configuration of all redundancy nodes in IOS.

Migrate from High Availability Version 2 to Version 3

Step 1 If you are currently running high availability version 2, a majority of the configuration of redundancy nodes should already be in guestshell. These nodes are stored in a file in the `~/azure/HA/node_file` location. Copy this file to bootflash by executing the `(guestshell#) cp ~/azure/HA/node_file /bootflash` command.

A redundancy node configuration in IOS still contains the binding between the node and the BFD peer. For example:

```
Router(config)#redundancy
Router(config-red)#cloud provider azure 4
Router(cloud-azure)#bfd peer 172.17.1.1
```

Step 2 To migrate this configuration to high availability version 3, add a global binding of all redundancy nodes to the BFD peer:

```
Router(config)#redundancy
Router(config-red)#cloud-ha bfd peer 172.17.1.1
And delete the individual redundancy nodes:
Router(config-red)#no cloud provider azure 4
```

Step 3 Delete all the existing nodes. This removes the nodes from the IOS configuration and deletes the same nodes configured in the guestshell. Once you delete all the nodes and you install the new HA version 3 python package, restore the configured nodes by copying the node file back by running the `(guestshell#) cp /bootflash/node_file ~/cloud/HA/node_file` script.

Step 4 Restart the High Availability setup after you copy the node_file. To manually restart the High Availability setup, run the `sudo systemctl start csr_ha` command.

Configure High Availability on CSR Running on Amazon Web Services

Table 2: Cloud Specific Configuration of Redundancy Parameters

Parameter	Switch	Description
Node Index	-i	Index that is used to uniquely identify this node. Valid values: 1–1023.
Region Name	-rg	Name of the region that contains the route table. For example, us-west-2.
Route Table Name	-t	Name of the route table to be updated. The name of the route table must begin with the substring rtb-. For example, rtb-001333c29ef2aec5f
Route	-r	If a route is unspecified, then the redundancy node is considered to apply to all routes in the routing table. The CSR cannot change routes which are of type local or gateway.
Next Hop Interface	-n	Name of the interface to which packets should be forwarded in order to reach the destination route. The name of the interface must begin with the substring eni-. For example, eni-07160c7e740ac8ef4.

Parameter	Switch	Description
Mode	-m	Indicates whether this router is the primary or secondary router for servicing this route. Valid values are primary or secondary. This is an optional parameter. The default value is secondary.

Create a Redundancy Node

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

Example:

```
create_node { switch  value } […[{ switch  value }]
```

A valid redundancy node must have the following parameters configured:

- Node Index
- Region Name
- Route Table Name
- Next Hop Interface Name

For example,

```
create_node.py -i 2 -t rtb-001333c29ef2aec5e -rg us-west-2 -n eni-07160c7e740ac8ef3 -r
2600:1f14:49b:9b03::/64
```

If successful, the script returns a value of zero.

Set Redundancy Node Parameters

To change the value of parameters in an existing redundancy node, run the following script: `set_params -i node_index { switch value } […[{ switch value }]`.

Example:

```
set_params.py -i 10 -r 15.0.0.0/16 -m primary
```

The index parameter (-i) is mandatory. This command sets the values of the specified parameters. If the specified parameter is already defined for the redundancy node, the value of the parameter is updated.

If this configuration is successful, the script returns a value of zero.

Clear Redundancy Node Parameters

If you want to clear the value of specified parameters for an existing redundancy node, run the following script:
`clear_params -i node_index {switch … switch].`

Example:

```
clear_params -i 10 -r -n
```

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

Specify only the switch parameter when you clear an associated value. Do not provide the existing values for the parameters to be cleared.

If the clearing is successful, the script returns a value of zero.

Authenticate the CSR1000v Router

If you want the CSR 1000v router to update a routing table in the AWS network, you must first authenticate the router. In AWS, you must create a policy that permits the CSR 1000v router to access the route table. For example:

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "cloudwatch:",
                "s3:",
                "ec2:AssociateRouteTable",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DescribeRouteTables",
                "ec2:DescribeVpcs",
                "ec2:ReplaceRoute",
                "ec2:DescribeRegions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DisassociateRouteTable",
                "ec2:ReplaceRouteTableAssociation",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

An IAM role is then created using this policy and applied to the EC2 resource.

After the CSR 1000v EC2 instances are created, the IAM role created above needs to be attached to each router.

Note See the AWS documentation for instructions on how to create policies, IAM roles, and how to associate a role to an EC2 instance.

Disable Source/Destination Address Checking

By default, network interfaces created in AWS have source and destination address checking enabled. The interface verifies all the traffic that passes through matches the source or destination address of the interface, otherwise it is dropped. For the CSR1000v to perform routing, this setting must be disabled on each CSR1000v interface.

Note See the AWS documentation for instructions on how to disable source/destination address checking on a network interface

Route Table Entry Types

The route tables in AWS cloud support different target types. These route targets include multiple types of gateways and connections. The CSR 1000v router is only capable of updating routes with a network interface target. Routes with other target types are ignored for the purposes of high availability.

If you configure a redundancy node without a specific route destination, the CSR 1000v attempts to update all the routes within a route table with a target type of network interface. All the other routes are ignored.

Configure Security Group

If you have a security group in use by the eth0 interface of the EC2 instance of the CSR 1000v, you must allow the BFD protocol to pass through the interface. Configure an inbound and outbound security rule that allows ports 4789 and 4790 to be passed.

Note See the AWS documentation for instructions on configuring security groups and attaching them to subnets and network interfaces.

Migrate from High Availability Version 2 to Version 3

In the high availability feature version 3, the configuration of redundancy nodes is temporarily supported via the IOS configuration command line interface. However, attempts to configure a redundancy node via IOS will result in a warning message indicating such operations are deprecated. To migrate from a CSR running high availability version 2 requires the configuration in IOS to be converted to the format supported in guestshell.

If you are performing an in-place upgrade or a binary upgrade to the IOS XE release 16.11.x, then the configuration of the redundancy nodes in IOS can be automatically copied to guestshell by performing the following steps:

If you are running the guestshell in the current IOS release, copy all important files out of the guestshell, as they will be erased as part of this upgrade. Files can be temporarily stored in /bootflash for example.

Step 1 Perform the binary upgrade to IOS XE release 16.11. Wait for the CSR to reboot and stabilize.

Step 2 Enable the guestshell and enter the shell:

```
guestshell enable
guestshell
```

Step 3 Install the CSR HA package for AWS, version 3 by using the script (guestshell) pip install csr_aws_ha --user.

Note If you are a user who has newly installed Cisco CSR1000V Release 17.3.1, you must use `pip3 install csr-[aws]-ha --user` to install high availability version 3. If you are upgrading from 17.2 to 17.3.1, and you have already enabled the Guestshell, you need not use the latest python script.

Step 4 In the IOS, configure the binding to the BFD peer router:

```
conf t
redundancy
cloud-ha bfd peer <peer-ip-addr>
end
```

This launches a process in the IOS to translate the configuration of redundancy nodes in IOS to guestshell. This process can take several minutes to complete, as each node is verified as it is transferred.

Step 5 Wait for the transfer to complete. As each node is transferred, its configuration is added to a file in guestshell `~/cloud/HA/node_file`. When the size of this file stabilizes, the transfer is done. Open this file in the guestshell and examine its contents by running the script `cat node_file`.

Step 6 Verify whether all the nodes and their parameters have been successfully transferred. Use the `create_node` and/or `set_params` scripts to make any adjustments.

Step 7 Restore any files copied out of the guestshell in step 1.

Step 8 Delete the configuration of all the redundancy nodes in IOS.

Configure High Availability in CSR 1000v Running On Google Cloud Platform

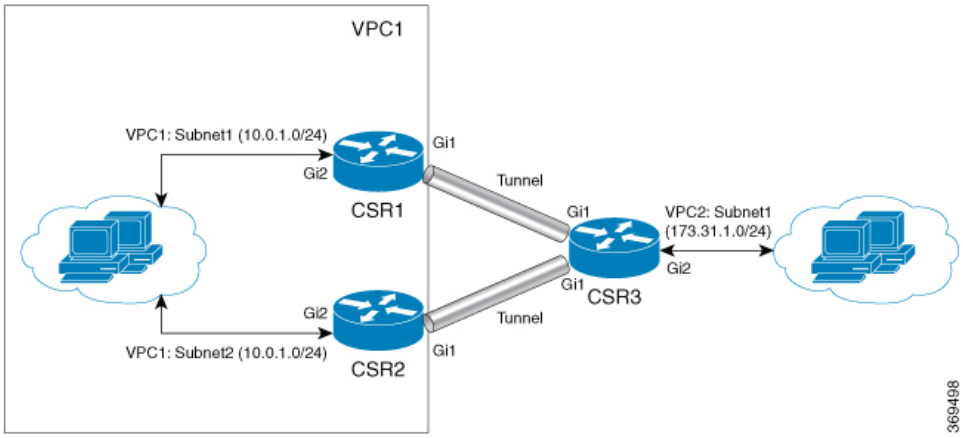
In the Google cloud, each static route belongs to the route table associated with a VPC and consists of following fields:

- Name and Description: These fields identify the route. A name is required, but a description is optional. Every route in your project must have a unique name.
- Network: Each route must be associated with exactly one VPC network.
- Destination range: The destination range is a single IPv4 CIDR block containing the IP addresses of systems that receive incoming packets. GCP does not support IPv6 destination ranges. Destinations must be expressed in CIDR notation, and the broadest destination possible is 0.0.0.0/0.

- Priority: Priority is used to determine which route should be used if multiple routes have identical destinations. Lower numbers indicate higher priorities; for example, a route with a priority value of 100 has a higher priority than one with a priority value of 200.
- Next hop: Static routes can have next hops that point to the default Internet gateway, a GCP instance, or a Cloud VPN tunnel. Refer to static route next hops for more information.
- Tags: You can specify a list of network tags so that the route will only apply to instances that have at least one of the listed tags. If you don't specify tags, GCP applies the route to all instances in the network.

For more information, see https://cloud.google.com/vpc/docs/routes. To configure High Availability in an active/active operation for two CSR 1000v routers in the Google network, you must create two routes in the route collection for each destination range, where each route points to one of the two routers as the next hop.

To understand this better, consider the following topology:



In the above topology, there are two routers configured in the HA mode. Both the routers have one interface in VPC1 and another in VPC. These two CSR 1000v routers have a Tunnel configured to another CSR that has an interface in VPC2. In this scenario, the following are the route entries in VPC1 for destination range of VPC 2 (172.31.0.0/16):

route-vcp2-csr1	172.31.0.0/16	100	None	IP:10:1:0:3	test-vpc
route-vcp2-csr2	172.31.0.0/16	200	None	IP:10.0.2.3	test-vpc

The active route is decided based on the route priority. Since route-vpc2-csr1 has a lower value, this route has a higher priority, thereby making CSR 1 as the active route.

Reversion to Primary CSR After Fault Recovery

If CSR 1 fails, CSR 2 detects a peer fail event through the BFD tunnel and deletes route-vpc2-csr1 from route collection making route-vpc2-csr2 as the active route for destination range 172.31.0.0/16.

When CSR 1 recovers, it adds route-vpc2-csr1 route back to the route collection which makes it the primary route again for all traffic to VPC 2. Please note it is possible to set equal route priority for both route entries in which case Google cloud uses both routes to send traffic to destination range.

On each CSR, you must create nodes corresponding to each route entry in route collection with next hop as the two CSRs.

When using mode (primary or secondary) option in HA to create a new node, ensure that the route with the higher priority (lower number) is marked as primary and the route with lower priority is marked as secondary.

User-Supplied Scripts

The guestshell is a container in which you can deploy your own scripts. HA Version 3 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.

Cloud Specific Configuration of Redundancy Parameters

Parameter	Is this parameter required?	Switch	Description
Node Index	Yes	-i	The index that is used to uniquely identify this node. Valid values: 1–255.
Cloud Provider	Yes	-p	Specify gcp for this parameter.
Project	Yes	-g	Specify the Google Project ID.
routeName	Yes	-a	The route name for which this CSR is next hop. For example from Fig. 2, if we are configuring node on CSR 1, this would be route-vpc2-csr1.
peerRouteName	Yes	-b	The route name for which the BFD peer CSR is next hop. For example from Fig. 2, if we are configuring node on CSR 1, this would be route-vpc2-csr2.

Parameter	Is this parameter required?	Switch	Description
Route	yes	-r	The IP address of the route to be updated in CIDR format. Can be IPv4 or IPv6 address. If a route is unspecified, then the redundancy node is considered to apply to all routes in the routing table of type virtual appliance. Note: Currently Google cloud does not have IPv6 support in VPC.
Next hop address	Yes	-n	The 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. The value can be an IPv4 or IPv6 address. Note: Currently Google cloud does not have IPv6 support in VPC.
hopPriority	Yes	-o	The route priority for the route for which the current CSR is the next hop.
VPC	Yes	-v	The VPC network name where the route with the current CSR as the next hop exists.

Create a Redundancy Node

Run the following script to create a redundancy node and add it to the database: `create_node { switch value } […[{ switch value }].`

You must configure the following parameters for a valid redundancy node:

- Node Index
- Cloud Provider

- Project ID
- Route Name
- Peer Route Name
- Route
- Next Hop Address
- Hop Priority
- VPC Name

```
create_node -i 1 -g <project-id> -r dest_network -o 200 -n nexthop_ip_addr -a route-name1 -b route-name2
 -p gcp -v vpc_name
```

If the configuration is successful, the script returns a value of zero.

Set Redundancy Node Parameters

To change the value of parameters in an existing redundancy node, run the following script: `set_params{ switch value } [...[{ switch value }].`

Example:

```
set_params -i 10 -r 15.0.0.0/16 -n 172.168.7.5
```

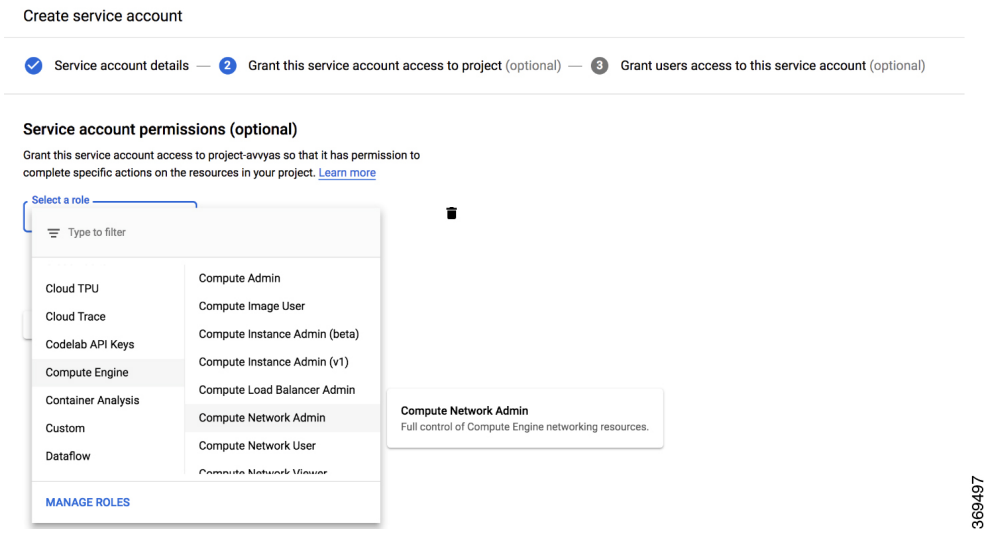
The index parameter (-i) is mandatory. This command sets the values of the specified parameters. If the specified parameter is already defined for the redundancy node, the value of the parameter is updated.

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 this configuration is successful, the script returns a value of zero.

Authenticate the CSR 1000V Router

Ensure that the service account associated with the CSR 1000v routers at least have a Compute Network Admin permission.



You can also provide the required permissions in a credentials file with name 'credentials.json' and place it under the /home/guestshell directory. The credentials file overrides the permissions supplied through the service account associated with the CSR 1000v instance.

Example Configurations

Example: Redunancy Nodes with Active/Active Configuration

Consider the HA configuration where route-name1 corresponds to route entry with next hop as CSR 1 and route-name2 corresponds to route entry with next hop as CSR 2 for destination network ‘dest_network’. To configure the routers in an active/active mode, set equal route priority for route-name1 and route-name2. In this case, Google cloud distributes the traffic between the routes using a five-tuple hash for affinity, thus implementing an ECMP routing design.

The node configuration on both routers corresponding to the route entries in Google route collection for the VPC would be:

```
create_node -i 1 -g <project-id> -r dest_network -o 200 -n nexthop_ip_addr_csr1 -a route-name1
 -b route-name2 -p gcp -v vpc_name

create_node -i 2 -g <project-id> -r dest_network -o 200 -n nexthop_ip_addr_csr2 -a route-name2
 -b route-name1 -p gcp -v vpc_name
```

Example: Redundancy Nodes with Active-Passive Configuration

Similarly, to configure CSRs in an active-passive mode, set the priority of one route higher than the other. In this case, Google cloud routes all the traffic from the VPC vpc_name to dest_network via the higher priority route (route-name1 for this example).

The node configuration on both routers corresponding to the route entries in Google route collection for the VPC would be:

```
create_node -i 1 -g <project-id> -r dest_network -o 200 -n nexthop_ip_addr_csr1 -a route-name1
 -b route-name2 -p gcp -v vpc_name
```

```
create_node -i 2 -g <project-id> -r dest_network -o 400 -n nexthop_ip_addr_csr2 -a route-name2
 -b route-name1 -p gcp -v vpc_name
```

Verify High Availability

Perform the following verification procedure by checking the log files. You can write a verbose log file to the directory ~/cloud/HA/events. Examine this log file to verify whether the operation is successful.

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

Troubleshoot High Availability Issues

Open the event file that is generated. This file is a debug log of the attempt to read and update the route described by the redundancy node. If the HA setup works as expected, the configuration output displays the status *Event handling completed*. If the system does not display this status, examine the log file in detail to determine which step of the verification failed.

Some of the common causes for failure include:

- Inability to obtain authentication credentials.
- The guestshell does not have network access.
- The authentication service is not running in Guestshell.
- The credentials for the CSR 1000v router are missing or incorrect.
- The router cannot access the route table entry.
- The route table was not correctly identified in the redundancy node
- The router was not granted permission to access the route table
- The specific route specified in the redundancy node does not exist

Note Cisco recommends that you use the `node_event` script with the verify event to test the configuration and the operation of the redundancy node.

Example: Troubleshooting Issues for High Availability Version 3

Execute the following command: `router#show iox`. See the following examples that provide the possible issues and how you can check and resolve these issues:

```
CSR#show iox

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

```
Libvirtd             : Running

CSR#guestshell enable

CSR#show app-hosting list
App id                           State
---------------------------------------------------------
guestshell                       RUNNING

CSR#guestshell
[guestshell@guestshell ~]$

[guestshell@guestshell ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=38 time=25.7 ms

Possible Cause:
The configuration of IOX and the creation of the VirtualPortGroup interface to provide the
 guestshell network access is part of the "day zero" configuration of the CSR.  If any of
the above steps did not work, check that the startup configuration of the CSR has been
altered.

How to Fix:
A reload of the CSR will re-apply the day zero configuration.

---------

Problem:
HA package installation failure

How to Check:
CSR#guestshell
gsday0-csr#guestshell
[guestshell@guestshell ~]$ ls
cloud
[guestshell@guestshell ~]$ cd cloud
[guestshell@guestshell cloud]$ ls
HA

You should see the directory ~/cloud/HA.
On an Azure provided cloud, you should also see a ~/cloud/authMgr directory.

Possible Cause:
The HA package was not installed, or was not installed using the --user option.

How to Fix:
Install the package and set up the environment:
 pip install csr_<provider>_ha --user
 source ~/.bashrc

---------

Problem:
HA server not running.

How to Check:
[guestshell@guestshell ~]$ systemctl status csr_ha
● csr_ha.service - CSR High Availability service
   Loaded: loaded (/etc/systemd/user/csr_ha.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-04-08 15:01:51 UTC; 2h 1min ago
 Main PID: 286 (python)
   CGroup: /system.slice/libvirtd.service/system.slice/csr_ha.service
           ├─286 python /home/guestshell/.local/lib/python2.7/site-packages/c...
           └─295 python /home/guestshell/.local/lib/python2.7/site-packages/c...
```

```
On an Azure provided network, the auth-token service should also be running.
[guestshell@guestshell ~]$ systemctl status csr_ha
● csr_ha.service - CSR High Availability service
   Loaded: loaded (/etc/systemd/user/csr_ha.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-04-08 15:01:51 UTC; 2h 1min ago
 Main PID: 286 (python)
   CGroup: /system.slice/libvirtd.service/system.slice/csr_ha.service
           ├─286 python /home/guestshell/.local/lib/python2.7/site-packages/c...
           └─295 python /home/guestshell/.local/lib/python2.7/site-packages/c...
[guestshell@guestshell ~]$ 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 Mon 2019-04-08 16:08:15 UTC; 57min ago
 Main PID: 542 (python)
   CGroup: /system.slice/libvirtd.service/system.slice/auth-token.service
           └─542 /usr/bin/python /home/guestshell/.local/lib/python2.7/site-p...

Possible Cause:
If the HA server has an error and crashes, it is automatically restarted.

How to Fix:
A service can be restarted manually
[guestshell@guestshell ~]$ sudo systemctl start csr_ha


---------

Problem:
CSR authentication not working on Azure.
This is an Azure specific error.

How to check:
If you perform a node_event on a redundancy node, and it fails while trying to read the
route table, it will generate a file ~/cloud/HA/events/routeTableGetRsp.
[guestshell@guestshell ~]$ cat routeTableGetRsp
{"error":{"code":"AuthenticationFailedMissingToken","message":"Authentication failed. The
'Authorization' header is missing the access token."}}

Possible Cause:
There are multiple possible causes.  And it depends upon the authentication mechanism you
are using:
 - System assigned managed identity
 - Registered application in Azure Active Directory (AAD)

Likely cause of a failure using system assigned managed identity is that it is not enabled
 on the CSR.

How to Fix:
Verify the CSR is enabled for system assigned managed identity.
In the Azure portal, navigate to the virtual machine running the CSR.
Under the Settings menu, select the Identity item.
Under the system assigned tab, verify the status is set to On.

When using AAD for authentication, the likely cause of the error is a mis-configuration of
 the application or a mis-match in the identifiers for the application configured in the
guestshell.

How to Fix:
The application in AAD must be given the proper permissions to read and write a route table.
In the Azure portal, navigate to the registered application you have created.
Under the API Access menu, select the Required permissions item.
Select the Windows Azure Active Directory API.  In the Enable Access pane, verify the
following permissions are set:
```

```
 - Application permission to read and write directory data
 - Delegated permission to sign in and read user profile
Select the Windows Azure Service Management API.  In the Enable Access pane, verify the
following permissions are set:
 - Delegated permission to access Azure service management as organization users

How to Fix:
In the Azure portal, navigate to the registered application you have created.
Select the Setting button for the application.
Verify the application_id, tenant_id, and application key in the portal match the values
configured in guestshell.  Verify the application key configured in guestshell is in URL
unencoded format.

-------------

Problem:
Route table entry not updated by a peer failure event.

How to Check:
For every node event a log file is generated in the directory ~/cloud/HA/events.
This file will indicate the event that was processed and its result.  Examine this file for
 possible errors.  It is likely in the case of an error that a file
~/cloud/HA/events/routeTableGetRsp is also written.  Also examine this file for additional
 insights.

Possible Causes:
A route was not correctly identified in a redundancy node.  Depending upon what parameter
in the redundancy node is in error, you may see different results.

Some examples:
[guestshell@guestshell events]$ cat routeTableGetRsp
{"error":{"code":"SubscriptionNotFound","message":"The subscription
'b0b1a9e2-444c-4ca5-acd9-bebd1e6874ef' could not be found."}}
This implies the Azure subscription ID was not entered correctly.

[guestshell@guestshell events]$ cat node*
Route GET request failed with code 403
Route table get response:
 {"error":{"code":"AuthorizationFailed","message":"The client
'b3ce41c0-bcef-41d7-9741-26bea31221c1' with object id 'b3ce41c0-bcef-41d7-9741-26bea31221c1'
 does not have authorization to perform action 'Microsoft.Network/routeTables/read' over
scope
'/subscriptions/b0b1a9e2-444c-4ca5-acd9-bebd1e6873eb/resourceGroups/gsday0-rg/providers/Microsoft.Network/routeTables/gsday0-sub4-RouteTable'."}}
Route table not found.
This implies the name of the route table was incorrect or does not exist.

[guestshell@guestshell events]$ cat node*
Did not find route 17.0.0.0/8 event type peerFail
This implies that the route does not exist.

How to Fix:
Make sure the identifiers in the redundancy node match the values in the cloud provider's
portal.

------------

Problem:
Route table entry not updated by a peer failure event.

How to Check:
For every node event a log file is generated in the directory ~/cloud/HA/events.
This file will indicate the event that was processed and its result.  Examine this file for
 possible errors.  It is likely in the case of an error that a file
~/cloud/HA/events/routeTableGetRsp is also written.  Also examine this file for additional
```

```
 insights.

Possible Causes:
The CSR has not been given permission to access the route table.
Fetching the route table
Route table get response:
 {"error":{"code":"AuthorizationFailed","message":"The client
'b3ce41c0-bcef-41d7-9741-26bea31221c1' with object id 'b3ce41c0-bcef-41d7-9741-26bea31221c1'
 does not have authorization to perform action 'Microsoft.Network/routeTables/read' over
scope
'/subscriptions/b0b1a9e2-444c-4ca5-acd9-bebd1e6873eb/resourceGroups/gsday0-rg/providers/Microsoft.Network/routeTables/gsday0-sub2-RouteTable'."}}
Route GET request failed with code 403
Route table get response:
 {"error":{"code":"AuthorizationFailed","message":"The client
'b3ce41c0-bcef-41d7-9741-26bea31221c1' with object id 'b3ce41c0-bcef-41d7-9741-26bea31221c1'
 does not have authorization to perform action 'Microsoft.Network/routeTables/read' over
scope
'/subscriptions/b0b1a9e2-444c-4ca5-acd9-bebd1e6873eb/resourceGroups/gsday0-rg/providers/Microsoft.Network/routeTables/gsday0-sub2-RouteTable'."}}
Route table not found.
CSR HA: Set route table for verify
Route Table not found


If none of these troubleshooting tips have resolved your problem, run this command:
[guestshell@guestshell ~]$ cd ~/cloud/HA
[guestshell@guestshell ~]$ bash debug_ha.sh
[guestshell@guestshell ~]$ ls /bootflash
You should see a file name ha_debug.tar.  Copy this file off the CSR and provide it to Cisco
 Technical Support for analysis.
```