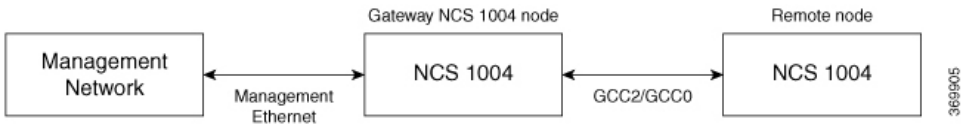
Remote Node Management

Remote node management using GCC

Remote node management using GCC is a network management method that

- leverages the General Communication Channel (GCC) embedded in optical transport networks,
- delivers reliable, out-of-band communication between centralized controllers and remote network nodes, and
- enables real-time monitoring, configuration, and maintenance activities without requiring direct physical access to each node.

Figure 1: Remote Node Management in Linear Topology



The remote nodes can be dynamically discovered over the GCC interface using OSPF. The connectivity to the management network can be achieved using OSPF and static routes.

Note The GCC2 and GCC0 interfaces are supported in NCS 1004. The GCC0 interface is supported on the Coherent DSP controller whereas the GCC2 interface is supported on the ODU controller.

Note The GCC0 and GCC2 interfaces are supported in Muxponder and Muxponder slice modes. Only the GCC0 interface is supported in the Regeneration (Regen) mode.

Note The GCC0 and GCC1 interfaces are supported on OTN-XP card and GCC2 interface is not supported.

Supported and unsupported features of the GCC interface

This table lists supported and unsupported features for remote node management using the GCC interface.

Table 1: Feature Support Overview:

Feature/Functionality	Supported on GCC Interface?	Notes
gRPC protocol	No	gRPC is not supported over the GCC interface.
Open Config	No	Not supported due to lack of gRPC support.
Streaming telemetry	No	Not supported due to lack of gRPC support.
Tx and Rx packet count statistics	Yes	Only Tx and Rx packet count information is available in GCC
Remote node management (after initial provisioning)	Yes	Devices can be managed over GCC only when connected through the management network using GCC.
Initial provisioning and bring-up via GCC	No	Must use console or management Ethernet interface for initial setup.
Remote management after headless/HA event	May be impacted	Events like reloads or driver restarts at intermediate nodes may affect management of subsequent nodes.
IP fragmentation for SCP protocol	No	Not supported; reduce packet size to less than 1454 bytes as a workaround.
TCP MSS configuration to avoid fragmentation	Yes	Use `tcp mss <maximum segment size>` in global config mode.
Coherent DSP controller (QXP card) GCC0 interface	Yes	Supported on QXP card.
GCC0 speed on QXP card	Yes	7.7 Mbps.

Note For operations not supported on the GCC interface, use the console or management Ethernet interface as alternatives. For SCP protocol, configure TCP MSS or IPv4 MTU settings to avoid IP fragmentation issues.

Supported protocols

These protocols are supported over the GCC interface:

- PING
- SSH
- TELNET
- SCP
- TFTP
- FTP
- SFTP
- HTTP
- HTTPS
- OSPF

Enable the GCC interface

Use this task to enable GCC0, GCC1, or GCC2 interfaces on various line cards (1.2T, OTN-XP) to support management and communication channels.

Procedure

Step 1 Enter configuration mode.

Step 2 Configure the controller and the GCC interface for a line card.

If you want to configure	Then use the command
GCC2 interface on the 1.2T card	**controller odu4** *R/S/I/P/L* **gcc2**
GCC0 interface on the 1.2T card	**controller CoherentDSP** *R/S/I/P/L* **gcc2**
GCC0 interface for the OTN-XP card	**controller {otu2 | otu2e | otu4}** *R/S/I/P/L* **gcc0**
GCC1 interface for the OTN-XP card	**controller {odu2 | odu2e | odu4 | oducn}** *R/S/I/P/L* **gcc1**

Example:

This sample configuration enables the GCC2 interface for the 1.2T line card.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#controller odu4 0/1/0/0/1
RP/0/RP0/CPU0:ios(config-otu)#gcc2
RP/0/RP0/CPU0:ios(config-otu)#commit
RP/0/RP0/CPU0:ios(config-otu)#exit
```

This sample configuration enables the GCC0 interface for the 1.2T line card.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#controller CoherentDSP0/0/1/1
RP/0/RP0/CPU0:ios(config-otu)#gcc0
RP/0/RP0/CPU0:ios(config-otu)#commit
RP/0/RP0/CPU0:ios(config-otu)#exit
```

This sample configuration enables the GCC0 interface for the OTN-XP line card.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#controller otu2 0/0/0/4/1
RP/0/RP0/CPU0:ios(config-otu)#gcc0
RP/0/RP0/CPU0:ios(config-otu)#commit
RP/0/RP0/CPU0:ios(config-otu)#exit
```

This sample configuration enables the GCC1 interface for the OTN-XP line card.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#controller odu2 0/0/0/4/1
RP/0/RP0/CPU0:ios(config-otu)#gcc1
RP/0/RP0/CPU0:ios(config-otu)#commit
RP/0/RP0/CPU0:ios(config-otu)#exit
```

Configure the GCC interface

Use this task to configure the GCC0, GCC1, and GCC2 interfaces on 1.2T and OTN-XP cards using static or loopback IP addresses.

Procedure

Step 1 Enter configuration mode.

Step 2 Specify the GCC2 interface for a line card.

If you want to configure	Then use the command
GCC2 interface on the 1.2T card	**interface gcc2** *R/S/I/P/L*
GCC0 interface on the 1.2T card	**interface gcc0** *R/S/I/P*

Step 3 Use the command **ipv4 address** *ipv4-address net-mask* to set the IPv4 address for the interface.

Example:

This sample configures the GCC2 interface using the static IP address on the 1.2T line card

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#interface gcc2 0/1/0/0/1
RP/0/RP0/CPU0:ios(config-if)#ipv4 address 198.51.100.244 255.255.255.0
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
RP/0/RP0/CPU0:ios(config)#exit
RP/0/RP0/CPU0:ios#show run interface gcc2 0/1/0/0/1
interface GCC20/1/0/0/1
ipv4 address 10.1.1.1 255.255.255.0
!
```

This sample configures the GCC2 interface using the loopback IP address on 1.2T card.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:R2(config)#interface gcc2 0/1/0/0/1
RP/0/RP0/CPU0:R2(config-if)#ipv4 unnumbered loopback 0
RP/0/RP0/CPU0:ios(config-if)#exit
```

This sample checks the status of GCC2 interface.

```
RP/0/RP0/CPU0:ios#show ipv4 interface brief
Wed Sep 22 17:10:04.190 IST
Interface IP-Address Status Protocol Vrf-Name
GCC20/0/0/0/1 198.51.100.234 Up Up default
GCC20/3/0/1/3 198.51.100.244 Up Up default
Loopback0 198.51.100.224 Up
```

This sample configures the GCC0 interface using the static IP address on 1.2T or OTN-XP card.enables the GCC1 interface for the OTN-XP line card.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#interface gcc0 0/1/0/0
P/0/RP0/CPU0:ios(config-if)#ipv4 address 198.51.100.244 255.255.255.0
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
RP/0/RP0/CPU0:ios(config)#exit
RP/0/RP0/CPU0:ios#show run interface gcc0 0/1/0/0
interface GCC00/1/0/0
ipv4 address 198.51.100.244 255.255.255.0
!
```

This sample configures the the GCC0 interface using the loopback IP address on 1.2T or OTN-XP card.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:R2(config)#interface gcc0 0/1/0/0
RP/0/RP0/CPU0:R2(config-if)#ipv4 unnumbered loopback 0
RP/0/RP0/CPU0:ios(config-if)#exit
RP/0/RP0/CPU0:ios(config)#exit
```

Configure static routes over the GCC interface

Use this task to configure the router to forward packets for specific networks or hosts via the GCC interface using manually defined routes.

Procedure

Step 1 Enter configuration mode.

Step 2 Enter the router static configuration mode.

Step 3 Use the command **address-family ipv4 unicast** *ip4 address default-gateway* to enter address family configuration mode. This step also configures a routing session using standard IPv4 address prefixes.

Example:

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#router static address-family ipv4 unicast 0.0.0.0/0 10.105.57.1
RP/0/RP0/CPU0:ios(config)#exit
```

Configure OSPF routes over the GCC interface

Enable OSPF dynamic routing between gateway and remote nodes using GCC interfaces.

This task enables OSPF over GCC interfaces, which facilitates efficient OSPF communication in specialized network environments.

Procedure

	Command or Action	Purpose
Step 1	Enter configuration mode and enable the OSPF routing process using the command **router ospf** *process-id* **router-id** *ip-address* **Example:**	
Step 2	Assign the OSPF area and specify the interfaces to include using the command **area** *area-id* **interface** *type R/S/I/P/L*	
Step 3	On the remote node, redistribute connected routes into OSPF, using the command **redistribute connected**.	

Step 4	Exit the configuration mode upon completion.
	Example:
	Gateway Node:
	`configure`
	`router ospf 1`
	`router-id 192.0.2.89 area 0`
	`interface Loopback0`
	`!`
	`interface MgmtEth0/RP0/CPU0/1`
	`!`
	`interface GCC20/0/0/0/1`
	`!`
	`interface GCC20/0/0/0/2`
	Remote Node:	

	Command or Action	Purpose
	configure router ospf 1 router-id 192.0.2.92 redistribute connected area 0 interface Loopback0 ! interface GCC20/0/0/0/1 ! interface GCC20/0/0/0/2	

OSPF is configured over GCC interfaces, enabling gateway and remote nodes to dynamically exchange routing information.

iBGP supports over GCC interfaces

iBGP support over GCC interfaces is a routing capability that

- allows external devices to exchange BGP routes through the management interfaces of NCS 1004 systems,
- enables NCS 1004 devices to advertise local networks and manage them using BGP-learned paths, and
- establishes iBGP sessions over GCC for exchanging BGP routes.

You can configure VRF on the GCC management interfaces (port 0 and port 1) of the NCS 1004 device to achieve traffic isolation between the two management ports.

NCS 1004 supports GCC0 and GCC2 interfaces for the 1.2T line card, allowing greater flexibility and connectivity options. The device advertises its local networks using BGP and manages them through paths learned from the iBGP sessions established over the GCC interfaces.

Restrictions for iBGP Support Using GCC

- IP fragmentation is not supported on the GCC interface.
- The BGP configuration over Open Config (OC) is not supported.

Note The limitations of Remote Node Management Using GCC are applicable for iBGP Support Using GCC. For more information, see Limitations of Remote Node Management.

Caution: Follow restrictions for iBGP support using GCC

Consider these restrictions when implementing iBGP support using GCC:

- Do not use IP fragmentation on the GCC interface; this function is not supported.
- Do not configure BGP over Open Config (OC) on GCC interfaces; this configuration is not supported.
- Observe all limitations found for Remote Node Management using GCC; these also apply to iBGP support using GCC

Enabling the GCC Interface

To enable the GCC2 interface, use the following commands:

```
configure
controller odu4 R/S/I/P/L
gcc2
commit
exit
```

To enable the GCC0 interface, use the following commands:

```
configure
controller CoherentDSP R/S/I/P
gcc0
commit
exit
```

Configuring the Management Interface

To configure the management Ethernet interface with VRF, use the following commands:

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios#interface MgmtEth0/RP0/CPU0/1
RP/0/RP0/CPU0:ios(config-if)#vrf transport-vrf
RP/0/RP0/CPU0:ios(config-if)#ipv4 address ipv4-address
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
```

The following example displays how to configure the management Ethernet interface with VRF.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios#interface MgmtEth0/RP0/CPU0/1
RP/0/RP0/CPU0:ios(config-if)#vrf transport-vrf
RP/0/RP0/CPU0:ios(config-if)#ipv4 address 192.0.2.1 255.255.255.255
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
```

Configuring the Loopback Interface

To configure the loopback interface 0 with VRF, use the following commands:

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios#interface Loopback0
```

```
RP/0/RP0/CPU0:ios(config-if)#vrf transport-vrf
RP/0/RP0/CPU0:ios(config-if)#ipv4 address ipv4-address
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
```

The following example displays how to configure the loopback interface 0 with VRF.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios#interface Loopback0
RP/0/RP0/CPU0:ios(config-if)#vrf transport-vrf
RP/0/RP0/CPU0:ios(config-if)#ipv4 address 192.0.2.1 255.255.255.255
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
```

Configuring the GCC interface

To configure the GCC2 interface with VRF and static IP address, use the following commands:

configure

interface gcc2 *R/S/I/P/L*

vrf transport-vrf

ipv4 address *ipv4-address*

commit

exit

To configure the GCC0 interface with VRF and static IP address, use the following commands:

configure

interface gcc0 *R/S/I/P*

vrf transport-vrf

ipv4 address *ipv4-address*

commit

exit

Examples

The following sample displays how to configure the GCC2 interface with VRF and static IP address.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#interface gcc2 0/1/0/0/1
RP/0/RP0/CPU0:ios(config-if)#vrf transport-vrf
P/0/RP0/CPU0:ios(config-if)#ipv4 address 198.51.100.5 255.255.255.0
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
RP/0/RP0/CPU0:ios(config)#exit
```

The following sample displays how to configure the GCC2 interface using loopback IP address.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:R2(config)#interface gcc2 0/1/0/0/1
RP/0/RP0/CPU0:R2(config-if)#ipv4 unnumbered loopback 0
```

```
RP/0/RP0/CPU0:ios(config-if)#exit
RP/0/RP0/CPU0:ios(config)#exit
```

The following sample displays how to configure the GCC0 interface with VRF and static IP address.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#interface gcc0 0/1/0/0
RP/0/RP0/CPU0:ios(config-if)#vrf transport-vrf
P/0/RP0/CPU0:ios(config-if)#ipv4 address 198.51.100.2 255.255.255.0
RP/0/RP0/CPU0:ios(config-if)#commit
RP/0/RP0/CPU0:ios(config-if)#exit
RP/0/RP0/CPU0:ios(config)#exit
```

The following sample displays how to configure the GCC0 interface using the loopback IP address.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:R2(config)#interface gcc0 0/1/0/0
RP/0/RP0/CPU0:R2(config-if)#ipv4 unnumbered loopback 0
RP/0/RP0/CPU0:ios(config-if)#exit
RP/0/RP0/CPU0:ios(config)#exit
```

Verifying iBGP Support Using GCC

To verify BGP support using GCC configuration, use the following **show** commands:

```
RP/0/RP0/CPU0:ios#show bgp vrf transport-vrf neighbors brief
Neighbor        Spk    AS Description                           Up/Down  NBRState
198.51.100.0     0     200                                     00:51:49 Established
198.51.100.1       0    100                                      00:50:32 Established
```

```
RP/0/RP0/CPU0:ios#show bgp vrf transport-vrf
BGP VRF transport-vrf, state: Active
BGP Route Distinguisher: 192.0.2.7:0
VRF ID: 0x60000002
BGP router identifier 192.0.2.7, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000002   RD version: 51
BGP main routing table version 51
BGP NSR Initial initsync version 11 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 192.0.2.7:0 (default for vrf transport-vrf)

*> 209.165.201.30/27        198.51.100.0             0         0 200 i
*> 209.165.201.28/27        0.0.0.0                  0         32768 i
*> 209.165.201.26/27        0    100      0 i
*> 209.165.201.24/27       198.51.100.2             0    100      0 300 i
```

```
RP/0/RP0/CPU0:ios#show bgp vrf transport-vrf
BGP VRF transport-vrf, state: Active
BGP Route Distinguisher: 203.0.113.10:0
VRF ID: 0x60000002
BGP router identifier 203.0.113.10, local AS number 100
Non-stop routing is enabled
```

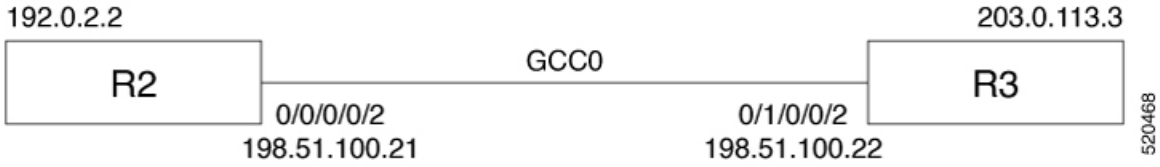
```
BGP table state: Active
Table ID: 0xe0000002   RD version: 51
BGP main routing table version 51
BGP NSR Initial initsync version 11 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 203.0.113.10:0 (default for vrf transport-vrf)

*> 209.165.201.30/27        198.51.100.0             0             0 200 i
*> 209.165.201.28/27        0.0.0.0                  0             32768 i
*>i209.165.201.26/27        198.51.100.12              0    100      0 i
*>i209.165.201.24/27       198.51.100.24               0    100      0 300 i
```

iBGP configuration parameters for GCC interfaces

This table summarizes the configuration required for enabling iBGP sessions between two NCS 1004 devices.

Use case:

Consider two NCS 1004 devices, R2 and R3, directly connected through GCC0 interfaces. In this use case, R2 (with IP address 198.51.100.21) and R3 (with IP address 198.51.100.22) are configured with the commands to establish an iBGP session using their respective transport-VRFs.



R2 is connected through GCC0 0/0/0/0 interface with IP address of 198.51.100.21 and R3 is connected through GCC0 0/1/0/0 with IP address of 198.51.100.22. The R2 and R3 devices are connected to external devices through management interfaces.

Table 2: iBGP configuration commands for NCS 1004

Configuration on R2	Configuration on R3

Global Configuration on R2
 	
`hw-module location 0/0`
` mxponder`
`  trunk-rate 600G`
`  client-rate 100GE`
 	
`vrf transport-vrf`
` address-family ipv4 unicast`
	Global Configuration on R3
	
	`hw-module location 0/0`
	` mxponder`
	`  trunk-rate 600G`
	`  client-rate 100GE`
	
	`vrf transport-vrf`
	` address-family ipv4 unicast`

Configuration on R2	Configuration on R3

Interface Configuration on R2
 	
`interface Loopback0`
` vrf transport-vrf`
` ipv4 address 192.0.2.2 255.255.255.255`
 	
`interface MgmtEth0/RP0/CPU0/1`
` vrf transport-vrf`
` ipv4 address 198.51.100.25 255.255.255.0`
 	
`controller ODU40/0/0/0/2`
` gcc2`
 	
`interface GCC20/0/0/0/2`
` vrf transport-vrf`
` ipv4 address 198.51.100.21 255.255.255.0`
	Interface Configuration on R3
	
	`interface Loopback0`
	` vrf transport-vrf`
	` ipv4 address 203.0.113.3 255.255.255.255`
	
	`interface MgmtEth0/RP0/CPU0/1`
	` vrf transport-vrf`
	` ipv4 address 198.51.100.32 255.255.255.0`
	
	`controller ODU40/1/0/0/2`
	` gcc2`
	
	`interface GCC20/1/0/0/2`
	` vrf transport-vrf`
	` ipv4 address 198.51.100.22 255.255.255.0`

Route-policy Configuration on R2
 	
`route-policy PASS-ALL`
`  pass`
`end-policy`
	Router Policy Configuration on R3
	
	`route-policy PASS-ALL`
	`  pass`
	`end-policy`

Static Route Configuration on R2
 	
`router static`
` address-family ipv4 unicast`
`  0.0.0.0/0 198.51.100.28`
` !`
` vrf transport-vrf`
`  address-family ipv4 unicast`
`   198.51.100.0/24 198.51.100.22`
	Static Route Configuration on R3
	
	`router static`
	` address-family ipv4 unicast`
	`  0.0.0.0/0 198.51.100.28`
	
	` vrf transport-vrf`
	`  address-family ipv4 unicast`
	`   198.51.100.0/24 198.51.100.21`

BGP Configurationon R2
 	
`router bgp 100`
` bgp router-id 192.0.2.123`
` address-family vpnv4 unicast`
` !`
` vrf transport-vrf`
`  rd auto`
`  address-family ipv4 unicast`
`   network 203.0.113.1/32`
`  !`
`  neighbor 198.51.100.22`
`   remote-as 100`
`   address-family ipv4 unicast`
`    route-policy PASS-ALL in`
`    route-policy PASS-ALL out`
`    next-hop-self`
`   !`
	BGP Configuration on R3
	
	`router bgp 100`
	` bgp router-id 192.0.2.124`
	` address-family vpnv4 unicast`
	` !`
	` vrf transport-vrf`
	`  rd auto`
	`  address-family ipv4 unicast`
	`   network 203.0.113.3/32`
	`  !`
	`  neighbor 198.51.100.21`
	`   remote-as 100`
	`   address-family ipv4 unicast`
	`    route-policy PASS-ALL in`
	`    route-policy PASS-ALL out`
	`    next-hop-self`
	`   !`

Configuration on R2	Configuration on R3

BGP Verification on R2
 	
`RP/0/RP0/CPU0:ios#show bgp sessions`
`Mon Jul 20 14:47:30.378 UTC`
 	
`Neighbor VRF Spk`
`AS InQ OutQ NBRState NSRState`
`198.51.100.22 transport-vrf 0`
`100 0 0 Established None`
	BGP Verification on R3
	
	`RP/0/RP0/CPU0:regen#show bgp sessions`
	`Tue Jul 21 02:50:14.134 UTC`
	
	`Neighbor VRF Spk`
	`AS InQ OutQ NBRState NSRState`
	`198.51.100.21 transport-vrf 0`
	`100 0 0 Established None`