

# Configure the OTN Circuits

This chapter describes the OTN circuits and procedures to configure the OTN circuits.


- Create a GMPLS UNI Circuit, on page 1
- Create a GMPLS NNI Circuit, on page 13
- OCH Mutual Circuit Diversity, on page 18
- Configure 1+1+R, on page 24
- Logical Patch Cord, on page 25


# Create a GMPLS UNI Circuit

### Before you begin

***Table 1: Feature History***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| GMPLS Support for NCS4K-4H-QDD-P Line Card | Cisco IOS XR Release 6.5.35 | GMPLS UNI circuits can now be created for the NCS4K-4H-QDD-P line card. This enhancement optimizes network resources and improves network utilization across packet and optical networks. |

Configure refresh optical interval. See Configure the Refresh Optical Interval, on page 8.

Configure loopback interface. See Provision Loopback Interface.

Configure the OSPF on an interface . See Configure the OSPF on an Interface, on page 5.

Configure the MPLS-TE on an OTN Controller. See Configure the MPLS-TE on an OTN Controller, on page 9.

### Procedure

**Step 1** **configure**

**Step 2** **lmp {gmpls | port | trace} optical-uni {controller | neighbor | router-id}** *controller-name R/S/I/P*

**Example:**

```
RP/0/ (config)# lmp gmpls optical-uni controller optics 0/0/0/4
```

Enters the LMP GMPLS UNI controller configuration mode. The value of lmp port ranges from 1 to 65535.

**Step 3** **neighbor** *name*

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-cntl)# neighbor xr4
```

Configures the LMP neighbor name of a controller.

**Step 4** **neighbor interface-id unnumbered** *value*

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-cntl)# neighbor interface-id unnumbered 4
```

Configures the interface identifier for the LMP. The value of interface-ID ranges from 1 to 4294967295.

**Step 5** **neighbor link-id ipv4 unicast** *address*

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-cntl)# neighbor link-id ipv4 unicast 1.2.2.4
```

Configures the LMP neighbor link identifier address.

**Step 6** **neighbor flexi-grid-capable**

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-cntl)# neighbor flexi-grid-capable
```

Enables GMPLS UNI flexible grid channel spacing.

**Step 7** **link-id ipv4 unicast** *value*

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-cntl)# link-id ipv4 unicast 1.2.3.4
```

Configures the LMP GMPLS UNI link identifier address.

**Step 8** **exit**

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-cntl)# exit
```

Exits the LMP GMPLS UNI controller configuration mode.

**Step 9** **lmp {gmpls | port | trace} optical-uni neighbor** *name*

**Example:**

```
RP/0/ (config)# lmp gmpls optical-uni neighbor xr4
```

Enters the LMP GMPLS UNI neighbor mode.

**Step 10** **ipcc routed**

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-nbr-xr4)# ipcc routed
```

Configures a GMPLS UNI LMP neighbor and create a routed IPCC.

**Step 11** **router-id ipv4 unicast** *value*

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-nbr-xr4)# router-id ipv4 unicast 1.1.1.1
```

Configures a router id for UNI LMP.

**Step 12** **exit**

**Example:**

```
RP/0/ (config-lmp-gmpls-uni-nbr-xr4)# exit
```

Exits the LMP GMPLS UNI neighbor mode.

**Step 13** **router-id ipv4 unicast** *value*

**Example:**

```
RP/0/ (config)# router-id ipv4 unicast 1.2.1.2
```

Configures a router id on the currently logged in router.

**Step 14** **mpls traffic-eng**

**Example:**

```
RP/0/ (config)# mpls traffic-eng
```

Enters the MPLS traffic-eng configuration mode.

**Step 15** **attribute-set xro** *attribute set name* **exclude strict lsp source** *head node IP address* **destination** *tail node IP address* **tunnel-id** *tunnel id* **extended-tunnel-id** *head node IP address*

**Note**
This step is applicable only when a diverse circuit is created.

**Example:**

```
RP/0/ (config)# attribute-set xro Xro_uni1_tun1_div_tun0
exclude strict lsp source 10.77.142.75 destination 10.77.142.71 tunnel-id 0 extended-tunnel-id
 10.77.142.75
```

Defines an attribute set for creating diverse circuit of a circuit with head node IP : 10.77.142.75, tail node IP:10.77.142.71 and tunnel id :0.

**Note**
The source, destination, tunnel-id and extended-tunnel-id is the information of the circuit whose diverse circuit you want to create.

**Step 16** **gmpls optical-uni controller** *controller-name R/S/I/P*

**Example:**

```
RP/0/ (config-mpls-te)# gmpls optical-uni controller optics 0/0/0/2
```

Enters the GMPLS UNI controller configuration mode.

**Step 17** **tunnel-properties tunnel-id** *value*

**Example:**

```
RP/0/ (config-te-gmpls-cntl)# tunnel-properties tunnel-id 6
```

Configures the GMPLS-UNI tunnel ID. The value of tunnel-ID ranges from 0 to 64535.

**Step 18** **tunnel-properties destination ipv4 unicast** *value*

**Example:**

```
RP/0/ (config-te-gmpls-cntl)# tunnel-properties destination ipv4 unicast 1.2.3.4
```

Specifies the GMPLS-UNI tunnel destination.

**Step 19** **tunnel-properties path-option 1 no-ero [xro-attribute-set] lockdown**

**Example:**

```
RP/0/ (config-te-gmpls-cntl)# tunnel-properties path-option 1 no-ero lockdown

RP/0/ (config-te-gmpls-cntl)# tunnel-properties path-option 1 no-ero xro-attribute-set
Xro_uni1_tun1_div_tun0 lockdown

RP/0/ (config-te-gmpls-cntl)# tunnel-properties path-option 1 explicit name
Explicit_path_tun100 lockdown verbatim
```

Configures the GMPLS-UNI path-option.

**Step 20** **exit**

**Example:**

```
RP/0/ (config-te-gmpls-cntl)# exit
```

Exits the GMPLS UNI controller configuration mode.

**Step 21** **commit**

---

### Example: Create a GMPLS-UNI Circuit

This example shows how to create a GMPLS-UNI circuit using Cisco IOS XR commands:

```
RP/0/(config)# lmp gmpls optical-uni controller optics 0/0/0/4
RP/0/(config-lmp-gmpls-uni-cntl)# neighbor xr4
RP/0/(config-lmp-gmpls-uni-cntl)# neighbor link-id ipv4 unicast 1.2.3.4
RP/0/(config-lmp-gmpls-uni-cntl)# neighbor flexi-grid-capable
RP/0/(config-lmp-gmpls-uni-cntl)# neighbor interface-id unnumbered 4
RP/0/(config-lmp-gmpls-uni-cntl)# link-id ipv4 unicast 1.2.3.4
RP/0/(config-lmp-gmpls-uni-cntl)# exit
RP/0/(config-lmp-gmpls-uni)# exit
RP/0/(config-lmp)# exit
RP/0/(config)# lmp gmpls optical-uni neighbor xr4
RP/0/(config-lmp-gmpls-uni-nbr-xr4)# ipcc routed
RP/0/(config-lmp-gmpls-uni-nbr-xr4)# router-id ipv4 unicast 1.1.1.1
RP/0/(config-lmp-gmpls-uni-nbr-xr4)# exit
RP/0/(config)# router-id ipv4 unicast 1.2.1.2
RP/0/(config)# mpls traffic-eng
RP/0/(config-mpls-te)# gmpls optical-uni controller optics 0/0/0/2
RP/0/(config-te-gmpls-cntl)# tunnel-properties tunnel-id 6
RP/0/(config-te-gmpls-cntl)# tunnel-properties destination ipv4 unicast 1.2.3.4
RP/0/(config-te-gmpls-cntl)# tunnel-properties path-option 10 no-ero lockdown
RP/0/(config-te-gmpls-cntl)# exit
RP/0/(config-te-gmpls-uni)# exit
RP/0/(config-mpls-te)# exit
```

### What to do next

Create an OTN Controller. Configure an OTN Controller

# Provision Loopback Interface

| | |
|---|---|
| **Purpose** | This procedure provisions the loopback interf |
| **Tools/Equipment** | None |
| **Prerequisite Procedures** | "Login to CTC" in System Setup and Softwa |
| **Required/As Needed** | As needed |
| **Onsite/Remote** | Onsite |
| **Security Level** | Provisioning or higher |

### Procedure

**Step 1** In the node view, click the **Provisioning** > **Network** > **Loopback IF** tabs.

**Step 2** If you want to create a loopback interface, complete the following:

- Click **Create**. The Create Loopback Interface dialog box appears.
- Enter the Interface ID, IP address, and network mask in the respective fields and click **OK**.

**Step 3** If you want to edit a loopback interface, complete the following:

- Click **Edit**. The Edit Loopback Interface dialog box appears.
- Modify the values of the IP Address and network mask as required and click **OK**.

**Step 4** Return to your originating procedure.

# Configure the OSPF on an Interface

### Before you begin

Optics controller should be created before configuring OSPF on an interface.

### Procedure

**Step 1** **configure**

**Step 2** **router ospf** *name-of-the-process*

**Example:**

```
RP/0/ (config)# router ospf abc
```

Enables OSPF routing and enters OSPF configuration mode.

**Step 3** **router-id** *id-of-the-router*

**Example:**

```
RP/0/ (config-ospf)# router-id 2.2.2.2
```

Specifies the OSPF router ID. The identifier is in the IPv4 address format.

**Step 4** **area** *id-of-the-area*

**Example:**

```
RP/0/ (config)# area 4
```

Specifies the OSPF area ID and enters the area configuration mode. The identifier can be either a decimal value or an IPv4 address. The OSPF area ID value ranges from 0 to 4294967295.

**Step 5** **interface loopback** *id*

**Example:**

```
RP/0/ (config-ospf-ar)# interface loopback  0
```

Configures OSPF on the specified interface.

**Step 6** **interface gcc0** *R/S/I/P*

**Example:**

```
RP/0/ (config-ospf-ar)# interface interface gcC0 0/1/0/1
```

Configures OSPF on the specified interface.

**Step 7** **commit**

---

#### Example: Configure OSPF on an Interface

The following example shows how to configure OSPF on an interface using Cisco IOS XR commands:

```
RP/0/# configure terminal
RP/0/(config)# router ospf abc
RP/0/(config-ospf)# router-id 2.2.2.2
RP/0/(config)# area 4
RP/0/(config-ospf-ar)# interface gcc0 0/0/0/4
RP/0/(config-ospf-ar)# exit
```

# Configure the OSPF-TE on an Interface

#### Before you begin

Optics controller should be created before configuring the OSPF-TE on an interface.

### Procedure

**Step 1** **configure**

**Step 2** **router ospf** *name-of-the-process*

**Example:**

```
RP/0/ (config)# router ospf abc
```

Enables OSPF routing and enters OSPF configuration mode.

**Step 3** **router-id** *id-of-the-router*

**Example:**

```
RP/0/ (config-ospf)# router-id 1.1.1.1
```

Specifies the OSPF router ID. The identifier is in the IPv4 address format.

**Step 4** **area** *id-of-the-area*

**Example:**

```
RP/0/ (config-ospf)# area 6
```

Specifies the OSPF area ID and enters the area configuration mode. The identifier can be either a decimal value or an IPv4 address. The OSPF area ID value ranges from 0 to 4294967295.

**Step 5** **mpls traffic-eng**

**Example:**

```
RP/0/ (config-ospf-ar)# mpls traffic-eng
```

Enables GMPLS for the specified OSPF-TE area.

**Step 6** **interface loopback** *range-of-the-interface loopback*

**Example:**

```
RP/0/(config-ospf-ar)# interface loopback 5
```

Creates a loopback interface for the specified OSPF-TE area and enters the loopback interface configuration mode. The interface loopback value ranges from 0 to 65535.

**Step 7** **passive [disable | enable]**

**Example:**

```
RP/0/ (config-ospf-ar-if)# passive enable
```

Specifies that the OSPF-TE configuration is passive.

**Step 8** **exit**

**Example:**

```
RP/0/ (config-ospf-ar-if)# exit
```

Exits the loopback interface configuration mode.

**Step 9** **interface GCC0** *R/S/I/P*

**Example:**

```
RP/0/(config-ospf-ar)# interface GCC0 0/0/0/20
```

Enables GCC on the interface and enters the OSPF-TE interface configuration mode.

**Step 10** **exit**

**Example:**

```
RP/0/ (config-ospf-ar)# exit
```

Exits the loopback interface configuration mode.

**Step 11** **mpls traffic-eng router-id loopback** *value*

**Example:**

```
RP/0/(config-ospf)# mpls traffic-eng router-id loopback 4
```

Enables GMPLS traffic on the loopback interface. The loopback value ranges from 0 to 65535.

**Step 12** **commit**

---

#### Example: Configure OSPF-TE on an Interface

The following example shows how to configure OSPF-TE on an interface using Cisco IOS XR commands:

```
RP/0/# configure terminal
RP/0/(config)# router ospf abc
RP/0/(config-ospf)# router-id 1.1.1.1
RP/0/(config-ospf)# area 6
RP/0/(config-ospf-ar)# mpls traffic-eng
RP/0/(config-ospf-ar)# interface loopback 5
RP/0/(config-ospf-ar-if)# passive enable
RP/0/(config-ospf-ar-if)# exit
RP/0/(config-ospf-ar)# interface GCC0 0/0/0/20
RP/0/(config-ospf-ar)# exit
RP/0/(config-ospf)# mpls traffic-eng router-id loopback 4
RP/0/(config-ospf)# exit
```

# Configure the Refresh Optical Interval

#### Before you begin

Optics controller should be created before configuring the refresh optical interval.

#### Procedure

---

**Step 1** **configure**

**Step 2** **rsvp**

**Example:**

```
RP/0/(config)# rsvp
```

Enters the RSVP mode.

**Step 3** **controller** *Type-of-the-controller R/S/I/P*

**Example:**

```
RP/0/(config-rsvp)# controller otu4 0/0/0/20
```

Enters the otu4 controller mode.

**Step 4** **signalling refresh out-of-band [missed | interval]** *value*

**Example:**

```
RP/0/(config-rsvp-cntl)# signalling refresh out-of-band missed 24
```

Specifies the interval between successive refreshes. The value of missed messages ranges from 1 to 110000 and refresh interval value ranges from 180 to 86400 seconds.

**Step 5** **commit**

### Example: Configure Refresh Optical Interval

The following example shows how to configure refresh optical interval using Cisco IOS XR commands:

```
RP/0/# configure terminal
RP/0/(config)# rsvp
RP/0/(config-rsvp)# controller otu4 0/0/0/20
RP/0/(config-rsvp-cntl)# signalling refresh out-of-band missed 24
RP/0/(config-rsvp-cntl)# exit
```

# Configure the MPLS-TE on an OTN Controller

### Before you begin

Optics controller should be created before configuring mpls-te on an otn controller.

### Procedure

**Step 1** **configure**

**Step 2** **mpls traffic-eng**

**Example:**

```
RP/0/ (config)# mpls traffic-eng
```

Enters the MPLS-TE configuration mode.

**Step 3** **gmpls [nni | optical-uni]**

**Example:**

```
RP/0/ (config-mpls-te)# gmpls nni
```

Enters the GMPLS Interface configuration mode. You can specify two types of interface: UNI and NNI.

**Step 4** **topology instance ospf** *name-of-the-topology instance* **area***value*

**Example:**

```
RP/0/ (config-te-gmpls-nni)# topology instance ospf abc area 5
```

Configures the topology instance of the OSPF. The value of OSPF area ID ranges from 0 to 4294967295.

**Step 5** **controller** *name-of-the-controller R/S/I/P*

**Example:**

```
RP/0/ (config-te-gmpls-nni-ti)# controller otu4 0/0/0/1
```

Configures the GMPLS-NNI on the specified OTN controller.

**Step 6** **admin-weight** *value-of-the-admin-weight*

**Example:**

```
RP/0/ (config-te-gmpls-nni-ti-cntl)# admin-weight 7
```

Configures admin weight on the specified controller. The valid range is from 0 to 65535.

**Step 7** **commit**

#### Example: Configure MPLS-TE on an OTN Controller

The following example shows how to configure MPLS-TE on an OTN controller using Cisco IOS XR commands:

```
RP/0/# configure terminal
RP/0/(config)#  mpls traffic-eng
RP/0/(config-mpls-te)#  gmpls nni
RP/0/(config-te-gmpls-nni-ti)#  controller otu4 0/0/0/1
RP/0/(config-te-gmpls-nni-ti-cntl)#  admin-weight 7
RP/0/(config-line)# exit
```

# Create an OTN Circuit through Control Plane

#### Before you begin

Optics controller should be created before creating an otn circuit.

#### Procedure

**Step 1** **configure**

**Step 2** **mpls traffic-eng**

**Example:**

```
RP/0/ (config)# mpls traffic-eng
```

Enters the MPLS traffic-eng configuration mode.

**Step 3** **gmpls nni**

**Example:**

```
RP/0/ (config-mpls-te)# gmpls optical-nni
```

Enters the GMPLS NNI configuration mode.

**Step 4** **controller odu-group-te** *tunnel-ID*

**Example:**

```
RP/0/ (config-te-gmpls-nni)# controller Odu-Group-Te 7
```

Enters the Odu-Group-Te configuration mode. The tunnel ID value ranges from 0 to 63535.

**Step 5** **destination** *type-of-the-destination* **unicast** *address-of-the-destination*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# destination ipv4 unicast 2.2.2.2
```

Specifies the destination IPv4 unicast address.

**Step 6** **static-uni ingress-port controller** *name-of-the-controller R/S/I/P* **egress-port unnumbered** *value*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# static-uni ingress-port controller GigabitEthernet 0/0/0/3
 egress-port unnumbered 6
```

Sets the static UNI endpoints of the NNI tunnel. The port IF index value ranges from 0 to 4294967295.

**Step 7** **signalled-bandwidth** *type-of-the-controller*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# signalled-bandwidth odu1
```

Sets the signal bandwidth of the controller.

**Step 8** **signalled-name** *name*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# signalled-name abcd
```

Specifies the signalled name for signalling. The maximum length is 64 characters.

**Step 9** **path-protection attribute-set** *name-of-the-attribute-set*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# path-protection attribute-set ss
```

Specifies the attribute set name for path protection. The maximum length is 32 characters.

**Step 10** **path-option** *value* **[dynamic | explicit] [lockdown | protected-by | restored-from]** *preference level-of-the-path-option* **[lockdown | restored-from]** *preference level-of-the-path-option* **lockdown**

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# path-option 5 dynamic protected-by 10 restored-from 30
lockdown
```

Configures the setup type and preference level of path option. The range of preference value is from 1 to 1000.

**Note**
You can modify a path option once you have created it.

**Step 11** **logging events lsp-status state**

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# logging events lsp-status state
```

Enables the interface lsp state alarms.

**Step 12** **commit**

---

**Example: Create an OTN Circuit**

The following example shows how to create an explicit path using Cisco IOS XR commands:

```
RP/0/ # configure terminal
RP/0/ (config)# mpls traffic-eng
RP/0/ (config-mpls-te)# gmpls optical-nni
RP/0/ (config-te-gmpls-nni)# controller Odu-Group-Te 7
RP/0/ (config-te-gmpls-tun-0x7)# destination ipv4 unicast 2.2.2.2
RP/0/ (config-te-gmpls-tun-0x7)# static-uni ingress-port controller GigabitEthernet 0/0/0/3
 egress-port unnumbered 6
RP/0/ (config-te-gmpls-tun-0x7)# signalled-bandwidth odu1
RP/0/ (config-te-gmpls-tun-0x7)# signalled-name abcd
RP/0/ (config-te-gmpls-tun-0x7)# path-protection attribute-set ss
RP/0/ (config-te-gmpls-tun-0x7)# path-option 5 dynamic protected-by 10 restored-from 30
lockdown
RP/0/ (config-te-gmpls-tun-0x7)# logging events lsp-status state
RP/0/ # commit
```

# Configure a Permanent Connection (xconnect)

**Before you begin**

Optics controller should be created before configuring a permanent connection.

**Procedure**

---

**Step 1** **configure**

**Step 2** **xconnect** *ID-of-the-xconnect* **endpoint-1** *Type-of-the-controller R/S/I/P* **endpoint-2** *Type-of-the-controller R/S/I/P*

**Example:**

```
RP/0/(config)# xconnect 5 endpoint-1 ODU1 0/0/0/2 endpoint-2 ODU1 0/0/0/2
```

Configures a permanent connection between two ODUk controllers. The cross connection ID value ranges from 1 to 32655

**Note**
A cross connection can only be made between same type of controllers such as ODU1-ODU1 and ODU2-ODU2.

**Step 3** **commit**

# View a Permanent Connections

**Before you begin**

Create a permanent connection. See Configure a Permanent Connection (xconnect), on page 12.

**Procedure**

**Step 1** **configure**

**Step 2** **show xconnect [all | id | trace]**

**Example:**

```
RP/0/# show xconnect all
```

Displays details of all the permanent connections.

**Step 3** **show xconnect [all | id | trace]** *ID-value*

**Example:**

```
RP/0/# show xconnect id 5
```

Displays details of all the permanent connections for the given connection ID. The cross connection ID value ranges from 1 to 32655.

**Step 4** **commit**

# Create a GMPLS NNI Circuit

**Before you begin**

Configure loopback interface. See Provision Loopback Interface.

Configure the OSPF on an interface . See Configure the OSPF on an Interface, on page 5.

Configure the MPLS-TE on an OTN Controller. See Configure the MPLS-TE on an OTN Controller, on page 9.

**Procedure**

**Step 1** **configure**

**Step 2** **mpls traffic-eng**

**Example:**

```
RP/0/ (config)# mpls traffic-eng
```

Enters the MPLS traffic-eng configuration mode.

**Step 3** **attribute-set xro** *attribute set name* **exclude strict lsp source** *head node IP address* **destination** *tail node IP address* **tunnel-id** *tunnel id* **extended-tunnel-id** *head node IP address*

**Note**
This step is applicable only when a diverse circuit is created.

**Example:**

```
RP/0/ (config)# attribute-set xro Xro_nni1_tun1_div_tun0
exclude strict lsp source 10.77.142.75 destination 10.77.142.71 tunnel-id 0 extended-tunnel-id
 10.77.142.75
```

Defines an attribute set for creating diverse circuit of a circuit with head node IP : 10.77.142.75, tail node IP:10.77.142.71 and tunnel id :0.

**Note**
The source, destination, tunnel-id and extended-tunnel-id is the information of the circuit whose diverse circuit you want to create.

**Step 4** **gmpls optical-nni controller** *controller-name R/S/I/P*

**Example:**

```
RP/0/ (config-mpls-te)# gmpls optical-nni controller Odu-Group-te 17
```

Enters the GMPLS-NNI controller configuration mode.

**Step 5** **destination ipv4 unicast** *value*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x11)# destination ipv4 unicast 1.2.3.4
```

Specifies the GMPLS-NNI tunnel destination.

**Step 6** **signalled-bandwidth ODU1**

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x11# signalled-bandwidth ODU1
```

Specifies the signalled bandwidth.

**Step 7** **path-option 1 dynamic protected-by** *value* **[xro-attribute-set]** *xro attribute set name* **lockdown**

**Note**
Use xro-attribute-set option only for creating a diverse circuit.

protected-by value is always set to none as only protection type 1+0 is supported with circuit diversity.

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x11)# path-option 1 dynamic protected-by 2 lockdown

RP/0/ (config-te-gmpls-tun-0x11)# path-option 1 dynamic protected-by none xro-attribute-set
 Xro_uni1_tun1_div_tun0 lockdown
```

Configures the GMPLS-NNI path-option.

**Step 8** **path-option 2 dynamic lockdown**

**Note**
This step is not applicable for creating a diverse circuit.

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x11)# path-option 2 dynamic lockdown
```

Configures the GMPLS-NNI path-option.

**Step 9** **path-protection attribute-set** *value*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x11)# path-protection attribute-set attSet1
```

Configures the GMPLS-NNI path-protection.

**Step 10** **static-uni ingress-portcontroller otu1** *R/S/I/P* **egress-port unnumbered** *value*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x11)# static-uni ingress-port controller otu1 0/1/0/20 egress-port
 unnumbered 56
```

Configures the interface identifier for the LMP. The value of interface-ID ranges from 1 to 4294967295.

**Step 11** **exit**

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x11)# exit
```

Exits the GMPLS UNI controller configuration mode.

**Step 12** **commit**

---

### Example: Create a GMPLS NNI Circuit

This example shows how to create a GMPLS NNI circuit using Cisco IOS XR commands:

```
RP/0/(config)# mpls traffic-eng
RP/0/(config-mpls-te)# gmpls optical-nni controller Odu-Group-te 17
RP/0/(config-te-gmpls-tun-0x11)# destination ipv4 unicast 1.2.3.4
RP/0/(config-te-gmpls-tun-0x11# signalled-bandwidth ODU1
RP/0/(config-te-gmpls-tun-0x11)# path-option 1 dynamic protected-by 2 lockdown
RP/0/(config-te-gmpls-tun-0x11)# path-option 2 dynamic lockdown
RP/0/(config-te-gmpls-tun-0x11)# path-protection attribute-set soumya
RP/0/(config-te-gmpls-tun-0x11)# static-uni ingress-port controller otu1 0/1/0/20 egress-port
 unnumbered 56
RP/0/(config-te-gmpls-tun-0x11)# exit
```

### What to do next

Create an OTN Controller. See Configure an OTN Controller.

# Configure the MPLS-TE on an OTN Controller using Local Termination

### Before you begin

Optics controller should be created before configuring mpls-te on an otn controller.

### Procedure

**Step 1** **configure**

**Step 2** **mpls traffic-eng**

**Example:**

```
RP/0/ (config)# mpls traffic-eng
```

Enters the MPLS-TE configuration mode.

**Step 3** **gmpls optical-nni**

**Example:**

```
RP/0/ (config-mpls-te)# gmpls optical-nni
```

Enters the GMPLS Interface configuration mode.

**Step 4** **topology instance ospf** *name-of-the-ospf instance* **area***value*

**Example:**

```
RP/0/ (config-te-gmpls-nni)# topology instance OTN abc area 0
```

Configures the topology instance of the OSPF. The value of OSPF area ID ranges from 0 to 4294967295.

**Step 5** **controller** *name-of-the-controller R/S/I/P*

**Example:**

```
RP/0/ (config-te-gmpls-nni-ti)# controller otu4 0/1/0/1
```

Configures the GMPLS-NNI on the specified OTN controller.

**Step 6** **tti-mode** *mode*

**Example:**

```
RP/0/ (config-te-gmpls-nni-ti-cntl)# tti-mode otu-sm
```

**Step 7** **admin-weight** *value-of-the-admin-weight*

**Example:**

```
RP/0/ (config-te-gmpls-nni-ti-cntl)# admin-weight 1
```

Configures admin weight on the specified controller. The valid range is from 0 to 65535.

**Step 8** **exit**

**Example:**

```
RP/0/ (config-te-gmpls-nni-ti-cntl)# exit
```

Exits the current sub mode.

**Step 9** **exit**

**Example:**

```
RP/0/ (config-te-gmpls-nni-ti)# exit
```

Exits the current sub mode.

**Step 10** **exit**

**Example:**

```
RP/0/ (config-te-gmpls-nni)# exit
```

Exits the current sub mode.

**Step 11** **gmpls optical-nni controller** *controller-name R/S/I/P*

**Example:**

```
RP/0/ (config-mpls-te)# gmpls optical-nni controller Odu-Group-te 17
```

Enters the GMPLS-NNI controller configuration mode.

**Step 12** **signalled-bandwidth***type-of-the-controller*

**Example:**

```
RP/0/(config-te-gmpls-tun-0x11)# signalled-bandwidth odu2
```

Sets the signal bandwidth of the controller.

**Step 13** **static-uni local-termination interface-name** *name-of-the-interface R/S/I/P* **remote-termination unnumbered** *value*

**Example:**

```
RP/0/(config-te-gmpls-tun-0x11)# static-uni local-termination interface-name TenGigE0/1/0/1/1
 remote-termination unnumbered 52
```

Configures the local termination interface identifier of the controller.

**Step 14** **destination** *type-of-the-destination* **unnumbered***value* **interface-ifindex** *index value*

**Example:**

```
RP/0/(config-te-gmpls-tun-0x11)#destination ipv4 unnumbered 13.13.13.13 interface-ifindex
55
```

Configures the destination.

**Step 15** **path-option** *value* **dynamic protected-by** *value* **lockdown**

**Example:**

```
RP/0/(config-te-gmpls-tun-0x11)# path-option 1 dynamic protected-by none lockdown
```

**Step 16** **commit**

---

### Example: Configure MPLS-TE on an OTN Controller Using Local Termination

The following example shows how to configure MPLS-TE on an OTN controller using local termination method:

```
RP/0/# configure
RP/0/(config)#  mpls traffic-eng
RP/0/(config-mpls-te)#  gmpls optical-nni
RP/0/(config-te-gmpls-nni)# topology instance ospf OTN area 0
RP/0/(config-te-gmpls-nni-ti)#  controller otu4 0/0/0/1
RP/0/(config-te-gmpls-nni-ti-cntl)# tti -mode otu-sm
RP/0/(config-te-gmpls-nni-ti-cntl)#  admin-weight 1
RP/0/(config-te-gmpls-nni-ti-cntl)#  exit
RP/0/(config-te-gmpls-nni-ti)#  exit
RP/0/(config-te-gmpls-nni)#  exit
RP/0/ (config-mpls-te)#  gmpls optical-nni controller Odu-Group-te 17
RP/0/(config-te-gmpls-tun-0x11)#  signalled -bandwidth odu2
RP/0/(config-te-gmpls-tun-0x11)#  static -uni local-termination interface-name
TenGigE0/1/0/1/1 remote-termination unnumbered 52
RP/0/(config-te-gmpls-tun-0x11)#  destination ipv4 unnumbered 13.13.13.13 interface- ifindex
 55
RP/0/(config-te-gmpls-tun-0x11)#path-option 1 dynamic protected-by none lockdown
```

# OCH Mutual Circuit Diversity

The OCH Mutual Circuit Diversity feature is an interoperability feature between a NCS 4000 series router and a NCS 2000 series router.

This feature enables the user to create two separate circuits whose paths use a different set of nodes.

Consider a DWDM circuit carrying a service. In order to provide protection and reduce the probability of simultaneous connection failures, the user can create a new circuit by defining a different set of nodes. In case of failure, the service is seamlessly carried forward by the other circuit, which has a different path. Typically, nodes dynamically choose the shortest path, where a circuit is created to reach the destination using minimum number of hops. This might result in network congestion if the same nodes are used by many circuits. Mutual circuit diversity enables the user to allocate different network paths for two circuits. Both the circuits are defined in such a way that there are no overlapping nodes (except the source node), and the paths are independent of each other.

This feature is supported on DWDM-enabled optical ports for the following cards:

- NCS4K-2H10T-OP-KS – port 2 to 11 when equipped with SFP+ with PID ONS-SC+-10G-C
- NCS4K-2H-W – trunk ports 2 and 3
- NCS4K-4H-OPW-QC2 – trunks ports 10 and 11

# Configuring Mutual Circuit Diversity - Overview of tasks

The following are the pre-requisites required to configure mutual circuit diversity (the user can use CTC to configure the following):

- Configure Link Management Protocol between the NCS 4000 and NCS 2000 nodes, refer Create an Local UNI LMP Using CTC
- Enable Refresh Optical Interval (RSVP), refer Configure a RSVP-TE Instance Using CTC

For configuring mutual diversity, the attributes are set for two circuits. Diverse paths are explicitly defined for both the circuits.

- Configure GMPLS tail node configuration
- Configure explicit path
- Create OCH trail circuits with mutual diversity

## Configure GMPLS tail node

This task enables the user to set up an optical unnumbered interface for the end point controllers.

**Procedure**

**Step 1** **configure**

**Step 2** **mpls traffic-eng**

**Example:**

```
RP/0/ (config) # mpls traffic-eng
```

Enters MPLS-TE configuration mode.

**Step 3** **gmpls optical-uni**

**Example:**

```
RP/0/ (config-mpls-te) # gmpls optical-uni
```

Enters the GMPLS UNI configuration submode.

**Step 4** **controller optics** *interface*

**Example:**

```
RP/0/ (config-te-gmpls) # controller optics 0/1/0/2
```

Enters the GMPLS UNI controller submode for the specified interface.

**Step 5** **commit**

### What to do next

Define paths for circuits

## Configure Explicit Path

This task enables the user to set-up the path for a circuit using strict or loose hops. Explicit path configuration is applicable to the GMPLS head node.

When a strict hop is configured, it identifies an exact path through which the circuit must be routed. When a loose hop is configured, the path can be changed.

**Procedure**

**Step 1** **configure**

**Step 2** **explicit-path name** *name*

**Example:**

```
RP/0/(config) # explicit-path name ExplicitPath0_2_0_2to1_1_1_85_sh0_sl1_p2
```

Provides the path name.

**Step 3** **index** *index-id* **next-address [strict | loose] ipv4 unicast unnumbered** *ip-address id*

**Example:**

```
RP/0/ (config) # index 10 next-address strict ipv4 unicast unnumbered 10.10.1.119 2130706962
```

Configures the ingress interface.

**Step 4** **index** *index-id* **next-address [strict | loose] ipv4 unicast unnumbered** *ip-address id*

**Example:**

```
RP/0/ (config) # index 80 next-address loose ipv4 unicast unnumbered 1.1.1.85 35
```

Configures the destination interface.

**Step 5** **commit**

**What to do next**

Configure diversity by defining the attributes for both the circuits

## Create OCH Trail Circuits with Mutual Diversity

This task enables the user to set the path attributes for a circuit. As earlier discussed, the attributes need to be defined for both the circuits and this configuration needs to be carried out twice. It is recommended to commit the configuration after setting the attributes for the second circuit, as signaling is initiated, only after the second circuit attributes are committed.

**Procedure**

**Step 1** **configure**

**Step 2** **mpls traffic-eng**

**Example:**

```
RP/0/ (config) # mpls traffic-eng
```

Enters MPLS-TE configuration mode.

**Step 3** **attribute-set xro exclude** *circuit-name*

**Example:**

```
RP/0/ (config-te) # attribute-set xro exclude CircuitB
```

Enters the attribute set submode and specifies the attribute set name. The path definition contains the circuit to be excluded.

**Step 4** **exclude srict lsp source** *source ip-address* **destination** *destination ip-address* **tunnel-id** *number* **extended tunnel-id** *source ip-address*

**Example:**

```
RP/0/ (config-te-attribute-set) # exclude strict lsp source 1.1.1.83 destination 1.1.1.63
tunnel-id 1 extended-tunnel-id 1.1.1.83
```

Sets the path diversity and defines the attributes.

**Step 5** **exit**

**Step 6** **gmpls optical-uni**

**Example:**

```
RP/0/ (config-mpls-te) # gmpls optical-uni
```

Enters the GMPLS UNI configuration submode.

**Step 7** **controller optics** *interface*

**Example:**

```
RP/0/ (config-te-gmpls) # controller optics 0/1/0/2
```

Enters the GMPLS UNI controller submode for the specified interface.

**Step 8** **announce srlgs**

**Example:**

```
RP/0/(config-te-gmpls-cntl)# announce srlgs
```

Announces discovered SRLGs to the system.

**Step 9** **tunnel-properties**

**Example:**

```
RP/0/(config-te-gmpls-cntl)# tunnel-properties
```

Enters the submode to configure tunnel-specific information for a GMPLS UNI controller.

**Step 10** **signalled-name** *circuit-name*

**Example:**

```
RP/0/(config-te-gmpls-cntl)# signalled-name Circuit A
```

Sets the name for the circuit which needs to follow a path different from the attributes defined earlier.

**Step 11** **tunnel-id** *number*

**Example:**

```
RP/0/(config-te-gmpls-tun)# tunnel-id 0
```

Specifies a tunnel-ID for a headend router of a GMPLS tunnel. The tunnel-ID is a 16-bit number ranging from 0 to 65535.

**Step 12** **record srlg**

**Example:**

```
RP/0/(config-te-gmpls-tun)# record srlg
```

Enables SRLG recording.

**Step 13** **destination ipv4 unicast** *address*

**Example:**

```
RP/0/(config-te-gmpls-tun)# destination ipv4 unicast 1.1.1.85
```

Specifies a tunnel destination for a headend router of a GMPLS tunnel. The destination argument is an IPv4 address.

**Step 14** **path-option** *number* **explicit-path name** *name* **xro-attribute-set exclude** *attribute* **lockdown verbatim**

**Example:**

```
RP/0/(config-te-gmpls-tun)# path-option 10 explicit-path name
ExplicitPath0_2_0_2to1_1_1_85_sh0_sl1_p2 xro-attribute-set exclude CircuitB lockdown verbatim
```

The XRO attribute set is attached to the GMPLS UNI tunnel through the path option. The path-option range is 1 to 1000.

**Step 15** **record-route**

**Example:**

```
RP/0/(config-te-gmpls-cntl)# record-route
```

Records the path taken by the circuit.

**Step 16** **commit**

# Example for Configuring Mutual Circuit Diversity

Let us consider two circuits, Circuit A and Circuit B, with the following parameters:

- Circuit A: Source address - 1.1.1.83; Destination address - 1.1.1.85
- Circuit B: Source address - 1.1.1.83; Destination address - 1.1.1.63

GMPLS tail node configuration

```
Circuit A
```

```
-----------
mpls traffic-eng
    gmpls optical-uni
        controller optics0/1/0/2
    !
!

Circuit B
----------
mpls traffic-eng
    gmpls optical-uni
        controller optics0/7/0/10
!
```

Explicit path configuration

```
Circuit A
----------
explicit-path name ExplicitPath0_2_0_2to1_1_1_85_sh0_sl1_p2
    index 10 next-address strict ipv4 unicast unnumbered 10.10.1.119 2130706962
    index 80 next-address loose ipv4 unicast unnumbered 1.1.1.85 35
!

Circuit B
----------
explicit-path name ExplicitPath0_15_0_10to1_1_1_63_sh0_sl7_p10
    index 10 next-address strict ipv4 unicast unnumbered 10.10.1.119 2130706964
    index 20 next-address loose ipv4 unicast unnumbered 1.1.1.63 169
!
```

Configuring mutual diversity by defining attributes for both the circuits

```
Circuit A
----------
mpls traffic-eng
    attribute-set xro exclude-CircuitB
       exclude strict lsp source 1.1.1.83 destination 1.1.1.63 tunnel-id 1 extended-tunnel-id
 1.1.1.83
    !

    gmpls optical-uni
        controller Optics0/2/0/2
            logging discovered-srlgs
            announce srlgs
            tunnel-properties
                signalled-name CircuitA
                tunnel-id 0
                record srlg
                destination ipv4 unicast 1.1.1.85
                path-option 10 explicit name ExplicitPath0_2_0_2to1_1_1_85_sh0_sl1_p2
xro-attribute-set exclude-CircuitB lockdown verbatim
                record-route
            !
        !
    !
!

Circuit B
----------
mpls traffic-eng
    attribute-set xro exclude-CircuitA
       exclude strict lsp source 1.1.1.83 destination 1.1.1.85 tunnel-id 0 extended-tunnel-id
```

```
 1.1.1.83
    !

    gmpls optical-uni
        controller Optics0/15/0/10
            logging discovered-srlgs
            announce srlgs
                tunnel-properties
                signalled-name VZO2toHUB1
                tunnel-id 1
                record srlg
                destination ipv4 unicast 1.1.1.63
                path-option 10 explicit name ExplicitPath0_15_0_10to1_1_1_63_sh0_sl7_p10
xro-attribute-set exclude-CircuitA lockdown verbatim
                record-route
            !
        !
    !
```

# Configure 1+1+R

This task enables the user to define a protect path and a restore path for a working path.

**Procedure**

**Step 1** **configure**

**Step 2** **mpls traffic-eng gmpls optical-nni**

**Example:**

```
RP/0/(config) # mpls traffic-eng gmpls optical-nni
```

Enters the MPLS traffic engineering and GMPLS NNI configuration mode.

**Step 3** **controller odu-group-te** *tunnel-ID*

**Example:**

```
RP/0/ (config-te-gmpls-nni)# controller Odu-Group-Te 7
```

Enters the Odu-Group-Te configuration mode. The tunnel ID value ranges from 0 to 63535.

**Step 4** **signalled-name** *name*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# signalled-name abcd
```

Specifies the signalling name. The maximum length is 64 characters.

**Step 5** **signalled-bandwidth** *controller*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# signalled-bandwidth odu1
```

Sets the signal bandwidth of the controller.

**Step 6** **static-uni ingress port controller** *controller R/S/I/P* **egress-port unnumbered** *value*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# static-uni ingress-port controller GigabitEthernet 0/0/0/3
 egress-port unnumbered 6
```

Sets the static UNI endpoints of the tunnel. The port index value ranges from 0 to 4294967295.

**Step 7** **destination ipv4 unicast** *destination-address*

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# destination ipv4 unicast 2.2.2.2
```

Specifies the destination IPv4 unicast address.

**Step 8** **path-option** *value* [ **dynamic** | **explicit** ] [ **protected-by** | **restored-from**] *preference-level* [ **protected-by** | **restored-from** *preference-level* **lockdown**

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# path-option 1 dynamic protected-by 2 restored-from 3
lockdown
```

Configures the path option 1; paths that will serve as the protect and restore paths are defined.

**Step 9** **path-option** *value* [ **dynamic** | **explicit** ] [ **protected-by** | **restored-from**] *preference-level* [ **protected-by** | **restored-from** *preference-level* **lockdown**

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# path-option 2 dynamic restored-from 3 lockdown
```

Configures the path option 2; restore path is defined.

**Step 10** **path-option** *value* [ **dynamic** | **explicit** ] [ **protected-by** | **restored-from**] *preference-level* [ **protected-by** | **restored-from** *preference-level* **lockdown**

**Example:**

```
RP/0/ (config-te-gmpls-tun-0x7)# path-option 3 dynamic lockdown
```

**Step 11** **commit**

# Logical Patch Cord

A logical patch cord creates a connection between two optical ports. This is an external connection, enables the network administrator to connect the front plates of the cards.

## Enabling a Logical Patch Cord

This task enables the user to create a connection between two optical ports.

**Procedure**

**Step 1** **configure**

**Step 2** **hw-module patchcord port optics** *interface* **port optics** *interface*

**Example:**

```
RP/0/ (config) # hw-module patchcord port optics 0/0/0/0 port optics 0/0/0/1
```

Enables connectivity between the two ports.

**Step 3** **commit**

### What to do next

Verify a configured patchcord:

```
show hw-module patchcord all
Hw-module Patchcord Configuration
-------------------------------------------------------
Source Port             Destination Port
-------------------------------------------------------
Optics0_0_0_0           Optics0_1_0_0
```