

Configuring Link Bundling

The Link Bundling feature allows you to group multiple point-to-point links together into one logical link and provide higher bidirectional bandwidth, redundancy, and load balancing between two routers. A virtual interface is assigned to the bundled link. The component links can be dynamically added and deleted from the virtual interface.

The virtual interface is treated as a single interface on which one can configure an IP address and other software features used by the link bundle. Packets sent to the link bundle are forwarded to one of the links in the bundle.

A link bundle is simply a group of ports that are bundled together and act as a single link. The advantages of link bundles are as follows:

- Multiple links can span several line cards to form a single interface. Thus, the failure of a single link does not cause a loss of connectivity.
- Bundled interfaces increase bandwidth availability, because traffic is forwarded over all available members of the bundle. Therefore, traffic can flow on the available links if one of the links within a bundle fails. Bandwidth can be added without interrupting packet flow.

Cisco IOS XR software supports the following method of forming bundles of Ethernet interfaces:

- IEEE 802.3ad—Standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. Links that are incompatible or have failed are automatically removed from a bundle.

- Limitations and Compatible Characteristics of Ethernet Link Bundles, on page 2
- Configuring Ethernet Link Bundles, on page 3
- Replace Ethernet bundles, on page 8
- Configuring LACP Fallback, on page 10
- VLANs on an Ethernet Link Bundle, on page 11
- Configuring VLAN over Bundles, on page 12
- LACP Short Period Time Intervals, on page 17
- Configuring the Default LACP Short Period Time Interval, on page 17
- Configuring Custom LACP Short Period Time Intervals, on page 19
- Bundle Consistency Checker, on page 26
- Information About Configuring Link Bundling, on page 30

Limitations and Compatible Characteristics of Ethernet Link Bundles

This list describes the properties and limitations of ethernet link bundles:

- The router supports mixed speed bundles. Mixed speed bundles allow member links of different bandwidth to be configured as active members in a single bundle. The ratio of the bandwidth for bundle members must not exceed 10. Also, the total weight of the bundle must not exceed 64. For example, 100Gbps link and 10Gbps links can be active members in a bundle and load-balancing on member links is based on bandwidth weightage.
- The weight of each bundle member is the ratio of its bandwidth to the lowest bandwidth member. Total weight of the bundle is the sum of weights or relative bandwidth of each bundle member. Since the weight for a bundle member is greater than or equal to 1 and less than or equal to 10, the total member of links in a bundle is less than 64 in mixed bundle case.
- Any type of Ethernet interfaces can be bundled, with or without the use of LACP (Link Aggregation Control Protocol).
- A single router can support a maximum of 256 bundle interfaces. Link bundles of only physical interfaces are supported.
- When enabling HQoS profile, the maximum available trunks by default (bundle main + sub-interfaces) are 256. If you need more trunks, configure the **hw-module profile bundle-scale <256/512/1024>** command. With HQoS enabled on bundle interfaces, the maximum priority level supported is 4.
- The following limitations apply to the number of supported bundle members with HQoS profile on Layer2 and Layer3 intefaces:
  - Maximum of 1024 trunks (128 physical interfaces + 896 sub-interfaces) and 16 bundle members.
  - Maximum of 256 trunks (128 physical interfaces + 128 sub-interfaces) and 64 bundle members.
  - Maximum of 512 trunks (128 physical interfaces + 384 sub-interfaces) and 32 bundle members.
- The following limitations apply to bundle sub-interfaces and the number of members per bundle :
  - Maximum of 1024 bundle sub-interfaces, each containing up to 16 member-links.
  - Maximum of 256 bundle sub-interfaces, each containing up to 64 member-links
  - Maximum of 512 bundle sub-interfaces, each containing up to 32 member-links
- When the H-QoS profile is enabled using the `hw-module profile qos hqos-enable` command, small density XR NCS 540 routers support a maximum of 32 link bundles, with up to 36 members per bundle.
- Physical layer and link layer configuration are performed on individual member links of a bundle.
- Configuration of network layer protocols and higher layer applications is performed on the bundle itself.
- IPv4 and IPv6 addressing is supported on ethernet link bundles.
- A bundle can be administratively enabled or disabled.
- Each individual link within a bundle can be administratively enabled or disabled.

- Ethernet link bundles are created in the same way as Ethernet channels, where the user enters the same configuration on both end systems.
- QoS is supported and is applied proportionally on each bundle member.
- In case static MAC address is configured on a bundle-ether interface, the following limitations are applied:
  - Locally generated packets, such as ICMP, BGP, and so on, going out from the interface have the source MAC address as the statically configured MAC address.
  - Transit (forwarded) packets going out of the interface do not have the configured static MAC as source MAC address. In such a scenario, the upper 36-bits come from the system MAC address (or the original/dynamic MAC address) and the lower 12-bits come from the MAC address configured on the bundle. To check the dynamic pool of MAC addresses included, use the `show ethernet mac-allocation detail` command.

    For example, if the dynamic MAC address was 008A.9624.48D8 and the configured static MAC address is 0011.2222.ABCD. Then, the source MAC for transit (forwarded) traffic will be 008A.9624.4BCD.



Note This limitation can cause traffic blackholing for the transit traffic, in case there is L2 ACL applied for security purpose. In such case, it is necessary to add permit statement for both MAC addresses in the L2 ACL.

- Load balancing (the distribution of data between member links) is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.
- All links within a single bundle must terminate on the same two systems.
- Bundled interfaces are point-to-point.
- A link must be in the up state before it can be in distributing state in a bundle.
- Only physical links can be bundle members.
- Multicast traffic is load balanced over the members of a bundle. For a given flow, the internal processes selects the member link, and the traffic for the flow is sent over that member.
- Hardware capacity for static MAC entries is limited by Ternary Content-Addressable Memory (TCAM) availability. To ensure optimal system performance and prevent resource exhaustion, limit the configuration of static MAC addresses to a maximum of eight per device.
- For Cisco NCS 5500 Series Routers running Cisco IOS XR Release 26.2.1, interface-level `logging events link-status` is not supported on Bundle-Ether interfaces that are configured with v`lan-switched trunk dot1q`.

Configuring Ethernet Link Bundles

This section describes how to configure an Ethernet link bundle.

Note In order for an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

Note When you use either physical Ethernet sub-interfaces or bundle Ethernet sub-interfaces, the device adds broadcast packets exiting a sub-interface to the output counters of the main interface. The output counters of the sub-interface do not increase. For example, when you send 100 broadcast packets through an egress physical or bundle sub-interface, the main interface output counters increase by 100, and the output broadcast packet counter of the sub-interface remains at zero.

Tip You can programmatically perform the configuration using `openconfig-if-aggregate.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco NCS 5500 Series Routers*.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps*
5. **bundle minimum-active links** *links*
6. **bundle maximum-active links** *links* [**hot-standby**]
7. **exit**
8. **interface HundredGigE** interface-path-id
9. **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]
10. **bundle port-priority** *priority*
11. **no shutdown**
12. **exit**
13. **bundle id** *bundle-id* `[` **mode** `{` **active** `|` **passive** `|` **on** `}]` **no shutdown exit**
14. **end** or **commit**
15. **exit**
16. **exit**
17. Perform Step 1 through Step 15 on the remote end of the connection.
18. **show bundle Bundle-Ether** *bundle-id*
19. **show lacp Bundle-Ether** *bundle-id*

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates a new Ethernet link bundle with the specified bundle-id. The range is 1 to 65535.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface specific configuration commands are entered. Use the **exit** command to exit from the interface configuration submode back to the normal global configuration mode.

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.



Note

- Only a Layer 3 bundle interface requires an IP address.

Step 4 **bundle minimum-active bandwidth** *kbps*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* [**hot-standby**]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1 hot-standby
```

(Optional) Implements 1:1 link protection for the bundle, which causes the highest-priority link in the bundle to become active and the second-highest-priority link to become the standby. Also, specifies that a switchover between active and standby LACP-enabled links is implemented per a proprietary optimization.

Note
- The priority of the active and standby links is based on the value of the **bundle port-priority** command.

Step 7 **exit**

Example:

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

Exits interface configuration submode for the Ethernet link bundle.

Step 8 **interface HundredGigE** interface-path-id

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/1/0/1
```

Enters interface configuration mode for the specified interface.

Enter the **HundredGigE** keyword to specify the interface type. Replace the *interface-path-id* argument with the node-id in the *rack*/*slot*/*module* format.

Step 9 **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle id 3 mode on
```

Adds the link to the specified bundle.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the link to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Note
- If you do not specify the **mode** keyword, the default mode is **on (**LACP is not run over the port).

Step 10 **bundle port-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle port-priority 1
```

(Optional) If you set the **bundle maximum-active links** command to 1, you must also set the priority of the active link to the highest priority (lowest value) and the standby link to the second-highest priority (next lowest value). For example, you can set the priority of the active link to 1 and the standby link to 2.

Step 11 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

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

Exits interface configuration submode for the Ethernet interface.

Step 13 **bundle id** *bundle-id* [**mode** {**active** | **passive** | **on**}] **no shutdown exit**

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/1/0/1

RP/0/RP0/CPU0:router(config-if)# bundle id 3

RP/0/RP0/CPU0:router(config-if)# bundle port-priority 2

RP/0/RP0/CPU0:router(config-if)# no shutdown

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

RP/0/RP0/CPU0:router(config)# interface TenGigE 0/1/0/1

RP/0/RP0/CPU0:router(config-if)# bundle id 3

RP/0/RP0/CPU0:router(config-if)# no shutdown

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

(Optional) Repeat Step 8 through Step 11 to add more links to the bundle.

Step 14 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 15 **exit**

Example:

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

Exits interface configuration mode.

Step 16 **exit**

Example:

```
RP/0/RP0/CPU0:router(config)# exit
```

Exits global configuration mode.

Step 17 Perform Step 1 through Step 15 on the remote end of the connection.

Brings up the other end of the link bundle.

Step 18 **show bundle Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router# show bundle Bundle-Ether 3
```

(Optional) Shows information about the specified Ethernet link bundle.

Step 19 **show lacp Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router# show lacp Bundle-Ether 3
```

(Optional) Shows detailed information about LACP ports and their peers.

Replace Ethernet bundles

Use this task when you move physical interfaces from existing Bundle-Ether interfaces to new bundles.

Separate the removal and addition operations into two commits. This sequence gives the router time to complete bundle processing before it receives the new member assignments.

Note Allow the router to complete processing between the commits. The required delay depends on the device scale and configured features. No fixed interval applies to every configuration.

Before you begin

Before you begin, identify every member interface, bundle subinterface, and existing bundle ID that you want to replace.

Procedure

Step 1 Remove the existing bundle ID from every member interface.

Example:

```
Router# configure
Router(config)# interface <interface-path-id>
Router(config-if)# no bundle id <bundle-id>
Router(config-if)# exit
```

The member interfaces no longer reference the existing bundles.

Step 2 Delete every subinterface that belongs to the existing bundles.

Example:

```
Router(config)# no interface Bundle-Ether <bundle-id>.<subinterface-id>
```

Repeat the command for each subinterface that belongs to an existing bundle.

The existing bundle subinterfaces are deleted.

Step 3 Delete the existing main bundles.

Example:

```
Router(config)# no interface Bundle-Ether <bundle-id>
```

The existing main bundles are removed from the configuration.

Step 4 Commit the member removal and bundle deletion in a single commit operation.

Example:

```
Router(config)# commit
```

Step 5 Wait for some time to allow the router to complete bundle processing.

The wait time can vary depending on the device scale and configured features.

The router completes processing of the member removal and bundle deletion.

Step 6 Create the new bundles with new IDs.

Example:

```
Router# configure
Router(config)# interface Bundle-Ether <new-bundle-id>
Router(config-if)# exit
```

The new Bundle-Ether interfaces are created.

Step 7 Assign the member interfaces to the new bundles.

Example:

```
Router(config)# interface <interface-path-id>
Router(config-if)# bundle id <new-bundle-id> [mode {active | on | passive}]
Router(config-if)# no shutdown
Router(config-if)# exit
```

Repeat the commands for every member interface.

The member interfaces reference the new bundles.

Step 8 Commit the new bundle creation and member assignments in a single commit operation.

Example:

```
Router(config)# commit
```

Step 9 Verify the new bundle state.

Example:

```
Router# show bundle Bundle-Ether <new-bundle-id>
```

The command displays the operational state and member status of the new bundle.

The old bundles are removed and the member interfaces operate in the new bundles.

Configuring LACP Fallback

This section describes how to configure the LACP Fallback feature.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **bundle lacp-fallback timeout** *timeout value*
4. **end** or **commit**
5. **show bundle infrastructure database ma bdl-info Bundle-e1010 | inc** *text*
6. **show bundle infrastructure database ma bdl-info Bundle-e1015 | inc** *text*

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

The **interface Bundle-Ether** command enters into the interface configuration submode, where you can enter interface-specific configuration commands. Use the **exit** command to exit from the interface configuration submode back to the normal return to global configuration mode.

Step 3 **bundle lacp-fallback timeout** *timeout value*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle lacp-fallback timeout 4
```

Enables the LACP Fallback feature.

Step 4 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# commit
```

Saves configuration changes.

Step 5 **show bundle infrastructure database ma bdl-info Bundle-e1010 | inc** *text*

Example:

```
RP/0/RP0/CPU0:router# show bundle infrastructure database ma bdl-info Bundle-e1010 | inc "fallback"
```

(Optional) Shows the MA information of the bundle manager.

Step 6 **show bundle infrastructure database ma bdl-info Bundle-e1015 | inc** *text*

Example:

```
RP/0/RP0/CPU0:router# show bundle infrastructure database ma bdl-info Bundle-e1015 | inc "fallback"
```

(Optional) Shows the MA information of the bundle manager.

VLANs on an Ethernet Link Bundle

802.1Q VLAN subinterfaces can be configured on 802.3ad Ethernet link bundles. Keep the following information in mind when adding VLANs on an Ethernet link bundle:

- There is no separate limit defined for Layer 3 sub-interfaces on a bundle. However, an overall system limit of 4000 is applicable for NCS5001 and NCS5002, while a limit of 2000 is applicable for NCS5011.

Note The memory requirement for bundle VLANs is slightly higher than standard physical interfaces.

To create a VLAN subinterface on a bundle, include the VLAN subinterface instance with the **interface Bundle-Ether** command, as follows:

interface Bundle-Ether *interface-bundle-id.subinterface*

After you create a VLAN on an Ethernet link bundle, all VLAN subinterface configuration is supported on that link bundle.

VLAN subinterfaces can support multiple Layer 2 frame types and services, such as Ethernet Flow Points - EFPs) and Layer 3 services.

Layer 2 EFPs are configured as follows:

```
interface bundle-ether instance.subinterface l2transport. encapsulation dot1q xxxxx
```

Layer 3 VLAN subinterfaces are configured as follows:

```
interface bundle-ether instance.subinterface, encapsulation dot1q xxxxx
```

Note The difference between the Layer 2 and Layer 3 interfaces is the **l2transport** keyword. Both types of interfaces use **dot1q encapsulation**.

Configuring VLAN over Bundles

This section describes how to configure a VLAN bundle. The creation of a VLAN bundle involves three main tasks:

SUMMARY STEPS

1. Create an Ethernet bundle.
2. Create VLAN subinterfaces and assign them to the Ethernet bundle.
3. Assign Ethernet links to the Ethernet bundle.

DETAILED STEPS

Procedure

Step 1 Create an Ethernet bundle.

Step 2 Create VLAN subinterfaces and assign them to the Ethernet bundle.

Step 3 Assign Ethernet links to the Ethernet bundle.

These tasks are describe in detail in the procedure that follows.

Note In order for a VLAN bundle to be active, you must perform the same configuration on both ends of the bundle connection.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps*
5. **bundle minimum-active links** *links*
6. **bundle maximum-active links** *links* [**hot-standby**]
7. **exit**
8. **interface Bundle-Ether** *bundle-id.vlan-id*
9. **encapsulation dot1q***vlan-id*
10. **ipv4 address** *ipv4-address mask*
11. **no shutdown**
12. **exit**
13. Repeat Step 9 through Step 12 to add more VLANS to the bundle you created in Step 2.
14. **end** or **commit**
15. **exit**
16. **exit**
17. **configure**
18. **interface** {**TenGigE** | **FortyGigE** | **HundredGigE**}*interface-path-id*

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface-specific configuration commands. Use the **exit** command to exit from the interface configuration submode back to the normal global configuration mode.

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* [**hot-standby**]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1 hot-standby
```

(Optional) Implements 1:1 link protection for the bundle, which causes the highest-priority link in the bundle to become active and the second-highest-priority link to become the standby. Also, specifies that a switchover between active and standby LACP-enabled links is implemented per a proprietary optimization.

Note The priority of the active and standby links is based on the value of the **bundle port-priority** command.

Step 7 **exit**

Example:

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

Exits the interface configuration submode.

Step 8 **interface Bundle-Ether** *bundle-id.vlan-id*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Bundle-Ether 3.1
```

Creates a new VLAN, and assigns the VLAN to the Ethernet bundle you created in Step 2.

Replace the *bundle-id* argument with the *bundle-id* you created in Step 2.

Replace the *vlan-id* with a subinterface identifier.

Range is from 1 to 4093 inclusive (0, 4094, and 4095 are reserved).



Note When you include the *.vlan-id* argument with the **interface Bundle-Ether** *bundle-id* command, you enter subinterface configuration mode.

Step 9 **encapsulation dot1q***vlan-id*

Example:

```
RP/0/RP0/CPU0:router(config-subif)# encapsulation dot1q 100
```

Sets the Layer 2 encapsulation of an interface.

Step 10 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router#(config-subif)# ipv4 address 10.1.2.3/24
```

Assigns an IP address and subnet mask to the subinterface.

Step 11 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router#(config-subif)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# exit
```

Exits subinterface configuration mode for the VLAN subinterface.

Step 13 Repeat Step 9 through Step 12 to add more VLANS to the bundle you created in Step 2.

(Optional) Adds more subinterfaces to the bundle.

Step 14 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# end
```

or

```
RP/0/RP0/CPU0:router(config-subif)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 15 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# end
```

Exits interface configuration mode.

Step 16 **exit**

Example:

```
RP/0/RP0/CPU0:router(config)# exit
```

Exits global configuration mode.

Step 17 **configure**

Example:

```
RP/0/RP0/CPU0:router # configure
```

Enters global configuration mode.

Step 18 **interface** {**TenGigE** | **FortyGigE** | **HundredGigE**}*interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 1/0/0/0
```

Enters interface configuration mode for the Ethernet interface you want to add to the Bundle.

Enter the **GigabitEthernet** or **TenGigE** keyword to specify the interface type. Replace the *interface-path-id* argument with the node-id in the rack/slot/module format.



Note A VLAN bundle is not active until you add an Ethernet interface on both ends of the link bundle.

LACP Short Period Time Intervals

As packets are exchanged across member links of a bundled interface, some member links may slow down or time-out and fail. LACP packets are exchanged periodically across these links to verify the stability and reliability of the links over which they pass. The configuration of short period time intervals, in which LACP packets are sent, enables faster detection and recovery from link failures.

Short period time intervals are configured as follows:

- In milliseconds
- In increments of 100 milliseconds
- In the range 100 to 1000 milliseconds
- The default is 1000 milliseconds (1 second)
- Up to 64 member links
- Up to 1280 packets per second (pps)

After 6missed packets, the link is detached from the bundle.

When the short period time interval is *not* configured, LACP packets are transmitted over a member link every 30 seconds by default.

When the short period time interval is configured, LACP packets are transmitted over a member link once every 1000 milliseconds (1 second) by default. Optionally, both the transmit and receive intervals can be configured to less than 1000 milliseconds, independently or together, in increments of 100 milliseconds (100, 200, 300, and so on).

When you configure a custom LACP short period *transmit* interval at one end of a link, you must configure the same time period for the *receive* interval at the other end of the link.



Note You must always configure the *transmit* interval at both ends of the connection before you configure the *receive* interval at either end of the connection. Failure to configure the *transmit* interval at both ends first results in route flapping (a route going up and down continuously). When you remove a custom LACP short period, you must do it in reverse order. You must remove the *receive* intervals first and then the *transmit* intervals.

Configuring the Default LACP Short Period Time Interval

This section describes how to configure the default short period time interval for sending and receiving LACP packets on a Gigabit Ethernet interface. This procedure also enables the LACP short period.

SUMMARY STEPS

1. **configure**
2. **interface HundredGigE***interface-path*
3. **bundle id** *number* **mode active**

4. **lacp period short**
5. **end** or **commit**

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface HundredGigE***interface-path*

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/1/0/1
```

Creates a Gigabit Ethernet interface and enters interface configuration mode.

Step 3 **bundle id** *number* **mode active**

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle id 1 mode active
```

Specifies the bundle interface and puts the member interface in active mode.

Step 4 **lacp period short**

Example:

```
RP/0/RP0/CPU0:router(config-if)# lacp period short
```

Configures a short period time interval for the sending and receiving of LACP packets, using the default time period of 1000 milliseconds or 1 second.

Step 5 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Example

This example shows how to configure the LACP short period time interval to the default time of 1000 milliseconds (1 second):

```
config
interface HundredGigE 0/1/0/1
   bundle id 1 mode active
   lacp period short
   commit
```

The following example shows how to configure custom LACP short period transmit and receive intervals to *less than* the default of 1000 milliseconds (1 second):

```
config
interface HundredGigE 0/1/0/1
   bundle id 1 mode active
   lacp period short
   commit
```

```
config
interface HundredGigE 0/1/0/1
    lacp period short transmit 100
   commit
```

```
config
interface HundredGigE 0/1/0/1
   lacp period short receive 100
   commit
```

Configuring Custom LACP Short Period Time Intervals

This section describes how to configure custom short period time intervals (less than 1000 milliseconds) for sending and receiving LACP packets on a Gigabit Ethernet interface.

Note You must always configure the *transmit* interval at both ends of the connection before you configure the *receive* interval at either end of the connection. Failure to configure the *transmit* interval at both ends first results in route flapping (a route going up and down continuously). When you remove a custom LACP short period, you must do it in reverse order. You must remove the *receive* intervals first and then the *transmit* intervals.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps*
5. **bundle minimum-active links** *links*
6. **bundle maximum-active links** *links*
7. **exit**
8. **interface Bundle-Ether** *bundle-id.vlan-id*
9. **dot1q vlan** *vlan-id*
10. **ipv4 address** *ipv4-address mask*
11. **no shutdown**
12. **exit**
13. Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.
14. **end** or **commit**
15. **exit**
16. **exit**
17. **show ethernet trunk bundle-ether** *instance*
18. **configure**
19. **interface** {**HundredGigE** } *interface-path-id*
20. **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]
21. **no shutdown**
22. Repeat Step 19 through Step 21 to add more Ethernet interfaces to the VLAN bundle.
23. **end** or **commit**
24. Perform Step 1 through Step 23 on the remote end of the VLAN bundle connection.
25. **show bundle Bundle-Ether** *bundle-id* [**reasons**]
26. **show ethernet trunk bundle-ether** *instance*

DETAILED STEPS

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface-specific configuration commands. Use the **exit** command to exit from the interface configuration submode back to the normal global configuration mode.

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links*

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle maximum-active links 1
```

(Optional) Designates one active link and one link in standby mode that can take over immediately for a bundle if the active link fails (1:1 protection).

Note

- The default number of active links allowed in a single bundle is 8.
- If the **bundle maximum-active** command is issued, then only the highest-priority link within the bundle is active. The priority is based on the value from the **bundle port-priority** command, where a lower value is a higher priority. Therefore, we recommend that you configure a higher priority on the link that you want to be the active link.

Step 7 **exit**

Example:

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

Exits the interface configuration submode.

Step 8 **interface Bundle-Ether** *bundle-id.vlan-id*

Example:

```
RP/0/RP0/CPU0:router#(config)# interface Bundle-Ether 3.1
```

Creates a new VLAN, and assigns the VLAN to the Ethernet bundle you created in Step 2.

Replace the *bundle-id* argument with the *bundle-id* you created in Step 2.

Replace the *vlan-id* with a subinterface identifier. Range is from 1 to 4093 inclusive (0, 4094, and 4095 are reserved).

Note
- When you include the *vlan-id* argument with the **interface Bundle-Ether** *bundle-id* command, you enter subinterface configuration mode.

Step 9 **dot1q vlan** *vlan-id*

Example:

```
RP/0/RP0/CPU0:router(config-subif)# dot1q vlan 10
```

Assigns a VLAN to the subinterface.

Replace the *vlan-id* argument with a subinterface identifier. Range is from 1 to 4093 inclusive (0, 4094, and 4095 are reserved).

Step 10 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-subif)# ipv4 address 10.1.2.3/24
```

Assigns an IP address and subnet mask to the subinterface.

Step 11 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# exit
```

Exits subinterface configuration mode for the VLAN subinterface.

Step 13 Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.

(Optional) Adds more subinterfaces to the bundle.

Step 14 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# end
```

or

```
RP/0/RP0/CPU0:router(config-subif)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting (yes/no/cancel)?`
- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 15 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# exit
```

Exits interface configuration mode.

Step 16 **exit**

Example:

```
RP/0/RP0/CPU0:router(config)# exit
```

Exits global configuration mode.

Step 17 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Step 18 **configure**

Example:

```
RP/0/RP0/CPU0:router # configure
```

Enters global configuration mode.

Step 19 **interface** {**HundredGigE** } *interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/1/0/1
```

Enters the interface configuration mode for the Ethernet interface you want to add to the Bundle.

Enter the **HundredGigE** keyword to specify the interface type. Replace the *interface-path-id* argument with the node-id in the rack/slot/module format.

Note
- A VLAN bundle is not active until you add an Ethernet interface on both ends of the link bundle.

Step 20 **bundle id** *bundle-id* [**mode** {**active** | **on** | **passive**}]

Example:

```
RP/0/RP0/CPU0:router(config-if)# bundle-id 3
```

Adds an Ethernet interface to the bundle you configured in Step 2 through Step 13.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the interface to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Note
- If you do not specify the **mode** keyword, the default mode is **on (**LACP is not run over the port).

Step 21 **no shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-if)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 22 Repeat Step 19 through Step 21 to add more Ethernet interfaces to the VLAN bundle.

—

Step 23 **end** or **commit**

Example:

```
RP/0/RP0/CPU0:router(config-subif)# end
```

or

```
RP/0/RP0/CPU0:router(config-subif)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting (yes/no/cancel)?`
- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 24 Perform Step 1 through Step 23 on the remote end of the VLAN bundle connection.

Brings up the other end of the link bundle.

Step 25 **show bundle Bundle-Ether** *bundle-id* [**reasons**]

Example:

```
RP/0/RP0/CPU0:router# show bundle Bundle-Ether 3 reasons
```

(Optional) Shows information about the specified Ethernet link bundle.

The **show bundle Bundle-Ether** command displays information about the specified bundle. If your bundle has been configured properly and is carrying traffic, the State field in the **show bundle Bundle-Ether** command output will show the number “4,” which means the specified VLAN bundle port is “distributing.”

Step 26 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Bundle Consistency Checker

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Bundle Consistency Checker (BCC)	Release 7.3.1	From the running configuration, Bundle Consistency Checker (BCC) fetches information about the ingress/egress traffic from the bundle, sub-bundle, and active member nodes and saves it in the database. BCC also collects data from all the running nodes and then compares it with the information saved in the database. Any inconsistencies, programming errors, stale entries are reported.

In a scaled setup, a bundle programming check is difficult to perform and time consuming. Moreover, an issue is reported only when the user detects it, and not automatically. During multiple test executions, it isn't possible to detect the initial failure, which causes other subsequent failures. Bundle Consistency Checker (BCC) implements bundle programming and consistency check by using the following steps:

1. BCC uses the running configuration to detect discrepancies.
2. BCC forms a Bundle Consistency Checker Data Base (BCCDB) with the bundle, sub-bundle, or member information fetched from the running configuration.
3. BCC dumps the required data from all available nodes. It then uses BCCDB as a source to verify bundle programming and consistency in all other layer dumps.
4. BCC reports inconsistencies, programming errors, stale entries, and deletes any pending objects.

Supporting Interfaces

The following interfaces support BCC:

- Bundle
- Bundle sub-interface

The following table lists BCC behaviour during inconsistencies in bundle configuration or programming errors.

Case	BCC Behaviour
When no bundle is configured	Router# **show bundle consistency** Building configuration... Dumping Data..... Done Parsing Data..... Not Done BCC Stopped: Found 3 info/exceptions/errors Logs Preview: 2020-07-13 10:34:22,774: INFO: Bundlemgr PD dont have any bundle data 2020-07-13 10:34:22,832: INFO: BMPI dont have any bundle data 2020-07-13 10:34:23,728: INFO: No Bundle is configured/No member is added to Bundle Logs: /var/log/bcc_exception.log /var/log/bcc_debug.log
When a bundle is configured but no member is added	Router# **show bundle consistency** Building configuration... Dumping Data..... Done Parsing Data..... Not Done BCC Stopped: Found 4 info/exceptions/errors Logs Preview: 2020-07-13 10:36:32,513: INFO: Bundlemgr PD dont have any bundle data 2020-07-13 10:36:32,566: INFO: No member is added to bundle BE1(0x3c00400c) 2020-07-13 10:36:32,566: INFO: BMPI dont have any bundle data 2020-07-13 10:36:33,453: INFO: No Bundle is configured/No member is added to Bundle Logs: /var/log/bcc_exception.log /var/log/bcc_debug.log

Case	BCC Behaviour

When a bundle is configured and members are added	`Router# show bundle consistency`
	`Building configuration...`
	`Dumping Data..... Done`
	`Parsing Data..... Done`
	`Bundle Consistency Check..... Done`
	`Bundle Programming Check..... Done`
	`Stale Entry Check..... Done`
	`Bundle Health Check..... Done`
	
	`Overall Results:`
	
	`Inconsistencies : 0 Stale Entries : 0`
	`BCM Programming Error : 0 Delete Pending DPA Objects : 0`
	`Info/Error/Python Exception : 0 Overall Bundle Health Status : WARNING`
	
	`Execute 'show bundle status' to see detailed reason for 'WARNING' in bundle health check`

Case	BCC Behaviour
When there is no encapsulation configuration for L2 or L3 sub-bundle or no member for L2 bundle	Router# **show bundle consistency** Building configuration... Dumping Data..... Done Parsing Data..... Done Bundle Consistency Check..... Done Bundle Programming Check..... Done Stale Entry Check..... Done Bundle Health Check..... Done Overall Results: Inconsistencies : 0 Stale Entries : 0 BCM Programming Error : 0 Delete Pending DPA Objects : 0 Info/Error/Python Exception : 3 Overall Bundle Health Status : WARNING Execute 'show bundle status' to see detailed reason for 'WARNING' in bundle health check Logs Preview: 2020-07-12 17:38:26,568: INFO: No member is added to bundle BE2(0x80042bc) ==> l2 bundle main 2020-07-12 17:38:32,573: interface Bundle-Ether1.1: Dont have any encapsulation config. ==> l3 sub 2020-07-12 17:38:32,574: interface Bundle-Ether1.130: Dont have any encapsulation config. ==> l2sub Logs: /var/log/bcc_inconsistencies.log /var/log/bcc_programming_error.log /var/log/bcc_stale_entries.log /var/log/bcc_delay_delete.log /var/log/bcc_bundle_health.log /var/log/bcc_exception.log /var/log/bcc_debug.log

Case	BCC Behaviour
During programming errors	Router# **show bundle consistency** Building configuration... Dumping Data..... Done Parsing Data..... Done Bundle Consistency Check..... Done Bundle Programming Check..... Done Stale Entry Check..... Done Bundle Health Check..... Done Overall Results: Inconsistencies : 0 Stale Entries : 0 BCM Programming Error : 1 Delete Pending DPA Objects : 0 Info/Error/Python Exception : 0 Overall Bundle Health Status : WARNING Execute 'show bundle status' to see detailed reason for 'WARNING' in bundle health check Logs Preview: 2020-07-12 18:48:22,658: Programming Error 1: BE1(0x80041ec) NPU 0,0/RP0/CPU0 Vlan Domain 0x33 != GigabitEthernet0_0_0_2 Vlan Domain 0xa Logs: /var/log/bcc_inconsistencies.log /var/log/bcc_programming_error.log /var/log/bcc_stale_entries.log /var/log/bcc_delay_delete.log /var/log/bcc_bundle_health.log /var/log/bcc_exception.log /var/log/bcc_debug.log

Information About Configuring Link Bundling

To configure link bundling, you must understand the following concepts:

IEEE 802.3ad Standard

The IEEE 802.3ad standard typically defines a method of forming Ethernet link bundles.

For each link configured as bundle member, the following information is exchanged between the systems that host each end of the link bundle:

- A globally unique local system identifier
- An identifier (operational key) for the bundle of which the link is a member
- An identifier (port ID) for the link
- The current aggregation status of the link

This information is used to form the link aggregation group identifier (LAG ID). Links that share a common LAG ID can be aggregated. Individual links have unique LAG IDs.

The system identifier distinguishes one router from another, and its uniqueness is guaranteed through the use of a MAC address from the system. The bundle and link identifiers have significance only to the router assigning them, which must guarantee that no two links have the same identifier, and that no two bundles have the same identifier.

The information from the peer system is combined with the information from the local system to determine the compatibility of the links configured to be members of a bundle.

The system assigns the MAC address of a Bundle-Ethernet interface from a chassis-wide MAC address pool. All member links use the bundle MAC address when passing bundle traffic. The system also sets any unicast or multicast addresses configured on the bundle on all member links.



Note The bundle MAC address operates independently of individual member links. System events, such as reloads, upgrades, or bundle membership changes, can modify this address. A change in the bundle MAC address may affect packet forwarding and can cause ARP/VRRP inconsistencies on downstream devices. This is especially critical in environments using LACP or port-channel interfaces.

Link Bundle Configuration Overview

The following steps provide a general overview of the link bundle configuration. Keep in mind that a link must be cleared of all previous network layer configuration before it can be added to a bundle:

1. In global configuration mode, create a link bundle. To create an Ethernet link bundle, enter the **interface Bundle-Ether** command.
2. Assign an IP address and subnet mask to the virtual interface using the **ipv4 address** command.
3. Add interfaces to the bundle you created in Step 1 with the **bundle id** command in the interface configuration submode.

   You can add up to 32 links to a single bundle.
4. You can optionally implement 1:1 link protection for the bundle by setting the **bundle maximum-active links** command to 1. Performing this configuration causes the highest-priority link in the bundle to become active and the second-highest-priority link to become the standby. (The link priority is based on the value of the **bundle port-priority** command.) If the active link fails, the standby link immediately becomes the active link.



Note A link is configured as a member of a bundle from the interface configuration submode for that link.

Link Switchover

By default, a maximum of 64 links in a bundle can actively carry traffic. If one member link in a bundle fails, traffic is redirected to the remaining operational member links.

You can optionally implement 1:1 link protection for a bundle by setting the **bundle maximum-active links** command to 1. By doing so, you designate one active link and one or more dedicated standby links. If the active link fails, a switchover occurs and a standby link immediately becomes active, thereby ensuring uninterrupted traffic.

If the active and standby links are running LACP, you can choose between an IEEE standard-based switchover (the default) or a faster proprietary optimized switchover. If the active and standby links are not running LACP, the proprietary optimized switchover option is used.

Regardless of the type of switchover you are using, you can disable the wait-while timer, which expedites the state negotiations of the standby link and causes a faster switchover from a failed active link to the standby link.

To do so, you can use the **lacp fast-switchover** command.

LACP Fallback

The LACP Fallback feature allows an active LACP interface to establish a Link Aggregation Group (LAG) port-channel before the port-channel receives the Link Aggregation and Control Protocol (LACP) protocol data units (PDU) from its peer. With the LACP Fallback feature configured, the router allows the server to bring up the LAG, before receiving any LACP PDUs from the server, and keeps one port active. This allows the server to establish a connection to PXE server over one Ethernet port, download its boot image and then continue the booting process. When the server boot process is complete, the server fully forms an LACP port-channel.

Link flap err-disables

A link flap error disable is a network reliability feature that

- automatically shuts down an interface if it frequently toggles between "up" and "down" states
- ensures network reliability by allowing traffic to reroute to stable backup paths when a port becomes unstable and
- is optimized for short-distance links within data centers, such as those connecting servers and Top-of-Rack (ToR) switches.

Supported interface types:

- Physical interfaces
- Breakout interfaces
- Bundle member interfaces

Table 2: Feature History Table

Feature Name	Release information	Feature Description
Link flap err-disables	Release 26.2.1	Introduced in this release on: NCS 5500 fixed port routersNCS 5500 modular routersNCS 5700 fixed port routersNCS 5500 line cardsNCS 5700 line cards [Mode: Native] You can achieve reliable network performance and a stable system state by isolating unstable interfaces that flap excessively, preventing routing protocol disruptions and ensuring effective workload failover. With this feature, the system monitors link transitions and automatically moves physical, breakout, or bundle member interfaces into an error-disabled state if a defined flap-count threshold is breached within a specific time window. This automated mechanism guarantees traffic redirection to stable paths while providing the flexibility of manual restoration or a timer-based automatic recovery once the link stabilizes. This feature introduces these changes: CLI: • **fault management link-flap error-disable** • **error-disable recovery cause** • **clear error-disable** • **show error-disable**

Syslog Messages

When an interface is disabled due to excessive flapping, the system generates this syslog message:

```
%PKT_INFRA-ERRDIS-6-ERROR_DISABLE : GigabitEthernet0/0/0/0: Error disabling due to reason:
 link-flap-exceeded
```

Limitations for configuring link-flap error-disable

Consider these limitations while configuring link flap error-disable.

- Only hardware-level link state changes are counted. Upper-layer protocol changes, such as Layer 2 protocols or line protocols, do not trigger a flap count.
- The link-flap error-disable feature works independently of the interface dampening feature. If both are configured, the first feature to reach its threshold will take action.
- To ensure optimal coexistence, review interactions with carrier-delay and object-tracking, as these may impact feature functionality.
- Do not enable global link-flap err-disable recovery on interfaces controlled by EVPN peer-track–based error-disable if your requirement is to keep the interface down while the EVPN peer-track condition remains down. The global recovery timer does not check the peer-track state before bringing the interface back up, which may cause the interface to recover unexpectedly. Review the global link-flap recovery timer and peer-track conditions before enabling global recovery in such scenarios.

Configure link-flap error-disable

Prevent repeated link flaps from destabilizing the network by automatically disabling affected interfaces and providing options for recovery.

Link-flap error-disable is a feature that detects frequent interface flaps and disables the interface to maintain network stability. Recovery methods allow administrators to restore normal operation as needed.

Before you begin

- Determine the target interface and the desired recovery method - manual or automatic.

Procedure

Step 1 Configure link-flap error-disable with a specific count and interval.

Example:

```
Router#configure
Router(config)#interface GigabitEthernet0/0/0/0
Router(config-if)#fault-management link-flap error-disable count 30 interval 420
Router(config-if)#commit
```

Step 2 Verify which interfaces are error-disabled.

Example:

```
Router#show error-disable
Interface            Error-Disable reason            Retry (s)  Time disabled
-------------------  ------------------------------  ---------  ---------------
Fo0/1/0/25           link-flap-exceeded                    ---         10:43:18
Fo0/1/0/26           link-flap-exceeded                    ---         10:43:18
```

Recover from err-disable.

Once an interface is in the `error-disabled` state, it can be recovered manually or automatically.

Step 3 Recover from err-disable.

- Configure automatic recovery to bring up the interface after a specific time.

```
Router(config)#error-disable recovery cause link-flap-exceeded interval 30
```

The default timer of 300 seconds for **error-disable recovery cause linked-flap exceeded interval**

- Manually recover an interface, reset the interface state.

```
Router(config)#interface <interface-type> <interface-path>
   RP/0/0/CPU0:ios(config-if)#shut
   RP/0/0/CPU0:ios(config-if)#no shut
   RP/0/0/CPU0:ios(config-if)#commit
```



Note While shut/no shut resets interface state, the recommended CLI to clear error-disable are **clear error-disable interface** and **clear error-disable location**.

Step 4 Verify the error-disable interface.

Example:

```
Router# show error-disable interface TenGigE 0/0/0/1
Interface            Error-Disable reason            Retry (s)  Time disabled
-------------------  ------------------------------  ---------  ---------------
Te0/0/0/1            link-flap-exceeded                    200         11:35:07
```

Step 5 Manually recover an interface, reset the interface state.

Example:



Note While shut/no shut resets interface state, the recommended CLI to clear error-disable are **clear error-disable interface** and **clear error-disable location**.

Step 6 Clear error-disable state from an interface explicitly.

Example:

```
Router#clear error-disable interface Hu0/0/0/0
```

Step 7 Clear error-disable state from all interfaces in a specific location.

Example:

```
Router#clear error-disable location interface TenGigE 0/0/0/1
```