Configuring Fabric and Interfaces

Fabric and Interface Configuration

To form the ACI fabric, Cisco Nexus 9000 Series ACI-mode switches are deployed in a leaf and spine topology managed by the APIC controller. Each leaf node is connected to all spine nodes with no connectivity between the leaf nodes. The interconnecting links between the leaf and spine nodes are called fabric links and the respective ports are called fabric ports. The fabric ports do not require user configuration for normal operation as these are auto discovered and factory default configuration is applied during fabric bring-up. All endpoint devices are connected to the leaf nodes through access ports. The access ports must be configured similar to those in NX-OS switches. Both fabric and access ports are represented as Interfaces as in NX-OS.

The leaf and spine nodes are considered different objects in the ACI model and support different sets of policies. In the CLI, these nodes are represented as leaf and spine respectively while both are commonly referred to as nodes. Leaf and spine node values are unique across all the pods in the fabric. FEX modules, if attached to the leaf nodes, will have fex-id values unique only within each leaf. For example, two leaf nodes can each have a FEX 101 attached.

Note Configuring FEX connections with FEX IDs `165` to `199` is not supported in the APIC GUI. To use one of these FEX IDs, configure the profile using the NX-OS style CLI. For more information, see *Configuring FEX Connections Using Interface Profiles with the NX-OS Style CLI*.

As of Cisco APIC, Release 3.0(1k), connections to FEX modules can be configured as profiles.

Interface Naming for Leaf and FEX Interfaces

In ACI fabric, most interface configuration is done for physical ports, port-channels, or vPCs (either directly connected to leaf nodes or connected through FEX modules). The general command syntax for each interface type is shown in the following table.

Interface Type	Command Syntax	Examples
Port	**interface ethernet** *slot/port*	interface eth 1/1
FEX Port	**interface ethernet** *fex-id/slot/port*	interface eth 101/1/1
Port-channel	**interface port-channel** *name*	interface port-channel foo
FEX Port-channel	**interface port-channel** *name* **fex** *fex-id*	interface port-channel foo fex 101
Virtual Port-channel (VPC)	**interface vpc** *name*	interface vpc foo
vPC over FEX	**interface vpc** *name* **fex** *fex-a fex-b*	interface vpc foo fex 101 102

Graceful Insertion and Removal (GIR) Mode

The Graceful Insertion and Removal (GIR) mode, or maintenance mode, allows you to isolate a switch from the network with minimum service disruption. In the GIR mode you can perform real-time debugging without affecting traffic.

You can use graceful insertion and removal to gracefully remove a switch and isolate it from the network in order to perform debugging operations. The switch is removed from the regular forwarding path with minimal traffic disruption. When you are finished performing the debugging operations, you can use graceful insertion to return the switch to its fully operational (normal) mode. In graceful removal, all external protocols are gracefully brought down except the fabric protocol (IS-IS) and the switch is isolated from the network. During maintenance mode, the maximum metric is advertised in IS-IS within the Cisco Application Centric Infrastructure (Cisco ACI) fabric and therefore the maintenance mode TOR does not attract traffic from the spine switches. In addition, all the front-panel interfaces are shutdown on the switch except the fabric interfaces. In graceful insertion, the switch is automatically decommissioned, rebooted, and recommissioned. When recommissioning is completed, all external protocols are restored and maximum metric in IS-IS is reset after 10 minutes.

The following protocols are supported:

- Border Gateway Protocol (BGP)
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Link Aggregation Control Protocol (LACP)

Important Notes

- Upgrading or downgrading a switch in maintenance mode is not supported.

- While the switch is in maintenance mode, the Ethernet port module stops propagating the interface related notifications. As a result, if the remote switch is rebooted or the fabric link is flapped during this time, the fabric link will not come up afterward unless the switch is manually rebooted (using the **acidiag touch clean** command), decommissioned, and recommissioned.
- For multi-pod, **IS-IS metric for redistributed routes** should be set to less than 63. To set the **IS-IS metric for redistributed routes**, choose **Fabric** > **Fabric Policies** > **Pod Policies** > **IS-IS Policy**.
- Existing GIR supports all Layer 3 traffic diversion. With LACP, all the Layer 2 traffic is also diverted to the redundant node. Once a node goes into maintenance mode, LACP running on the node immediately informs neighbors that it can no longer be aggregated as part of port-channel. All traffic is then diverted to the vPC peer node.
- For a GIR upgrade, Cisco Application Policy Infrastructure Controller (Cisco APIC)-connected leaf switches must be put into different maintenance groups such that the Cisco APIC-connected leaf switches get upgraded one at a time.

Removing a Switch to Maintenance Mode Using the CLI

Use this procedure to remove a switch to maintenance mode using the CLI.

Procedure

	Command or Action	Purpose
Step 1	[**no**]**debug-switch** *node_id or node_name*	Removes the switch to maintenance mode.

Inserting a Switch to Operation Mode Using CLI

Use this procedure to insert a switch to operational mode using the CLI.

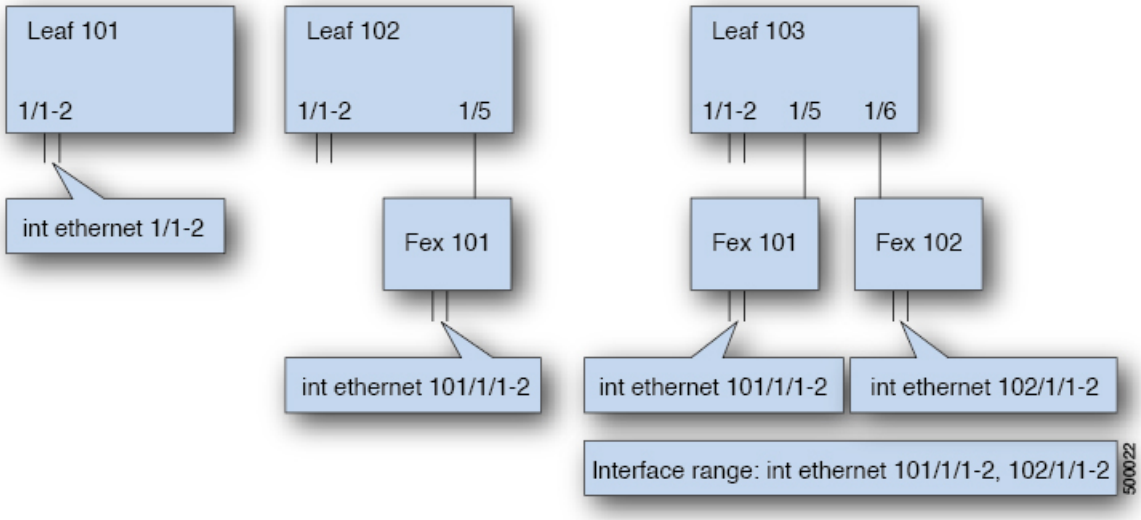
Procedure

	Command or Action	Purpose
Step 1	[**no**]**no debug-switch** *node_id or node_name*	Inserts the switch to operational mode.

Configuring Physical Ports in Leaf Nodes and FEX Devices Using the NX-OS CLI

The commands in the following examples create many managed objects (MOs) in the ACI policy model that are fully compatible with the REST API/SDK and GUI. However, the CLI user can focus on the intended network configuration instead of ACI model internals.

The following figure shows examples of Ethernet ports directly on leaf nodes or FEX modules attached to leaf nodes and how each is represented in the CLI. For FEX ports, the *fex-id* is included in the naming of the port itself as in **ethernet 101/1/1** . While describing an interface range, the **ethernet** keyword need not be repeated as in NX-OS. Example: **interface ethernet 101/1/1-2, 102/1/1-2** .



- Leaf node ID numbers are global.
- The *fex-id* numbers are local to each leaf.
- Note the space after the keyword **ethernet** .

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	**leaf** *node-id* **Example:** `apic1(config)# leaf 102`	Specifies the leaf or leafs to be configured. The *node-id* can be a single node ID or a range of IDs, in the form *node-id1 − node-id2* , to which the configuration will be applied.
Step 3	**interface** *type* **Example:** `apic1(config-leaf)# interface ethernet 1/2`	Specifies the interface that you are configuring. You can specify the interface type and identity. For an Ethernet port, use “ethernet slot / port.”
Step 4	(Optional) **fex associate** *node-id* **Example:** `apic1(config-leaf-if)# fex associate 101`	If the interface or interfaces to be configured are FEX interfaces, you must use this command to attach the FEX module to a leaf node before configuration. **Note** This step is required before creating a port-channel using FEX ports.

	Command or Action	Purpose
Step 5	**speed** *speed* **Example:** `apic1(config-leaf-if)# speed 10G`	The speed setting is shown as an example. At this point you can configure any of the interface settings shown in the table below.

The following table shows the interface settings that can be configured at this point.

Command	Purpose
[no] shut	Shut down physical interface
[no] speed *speedValue*	Set the speed for physical interface
[no] link debounce time *time*	Set link debounce
[no] negotiate auto	Configure negotiate
[no] cdp enable	Disable/enable Cisco Discovery Protocol (CDP)
[no] mcp enable	Disable/enable Mis-cabling Protocol (MCP)
[no] lldp transmit	Set the transmit for physical interface
[no] lldp receive	Set the LLDP receive for physical interface
spanning-tree {bpduguard | bpdufilter} {enable | disable}	Configure spanning tree BPDU
[no] storm-control level *percentage* [burst-rate *percentage*]	Storm-control configuration (percentage)
[no] storm-control pps *packets-per-second* burst-rate *packets-per-second*	Storm-control configuration (packets-per-second)

Examples

Configure one port in a leaf node. The following example shows how to configure the interface eth1/2 in leaf 101 for the following properties: speed, cdp, and admin state.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/2
apic1(config-leaf-if)# speed 10G
apic1(config-leaf-if)# cdp enable
apic1(config-leaf-if)# no shut
```

Configure multiple ports in multiple leaf nodes. The following example shows the configuration of speed for interfaces eth1/1-10 for each of the leaf nodes 101-103.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface eth 1/1-10
```

```
apic1(config-leaf-if)# speed 10G
```

Attach a FEX to a leaf node. The following example shows how to attach a FEX module to a leaf node. Unlike in NX-OS, the leaf port Eth1/5 is implicitly configured as fabric port and a FEX fabric port-channel is created internally with the FEX uplink port(s). In ACI, the FEX fabric port-channels use default configuration and no user configuration is allowed.

Note This step is required before creating a port-channel using FEX ports, as described in the next example.

```
apic1(config)# leaf 102
apic1(config-leaf)# interface eth 1/5
apic1(config-leaf-if)# fex associate 101
```

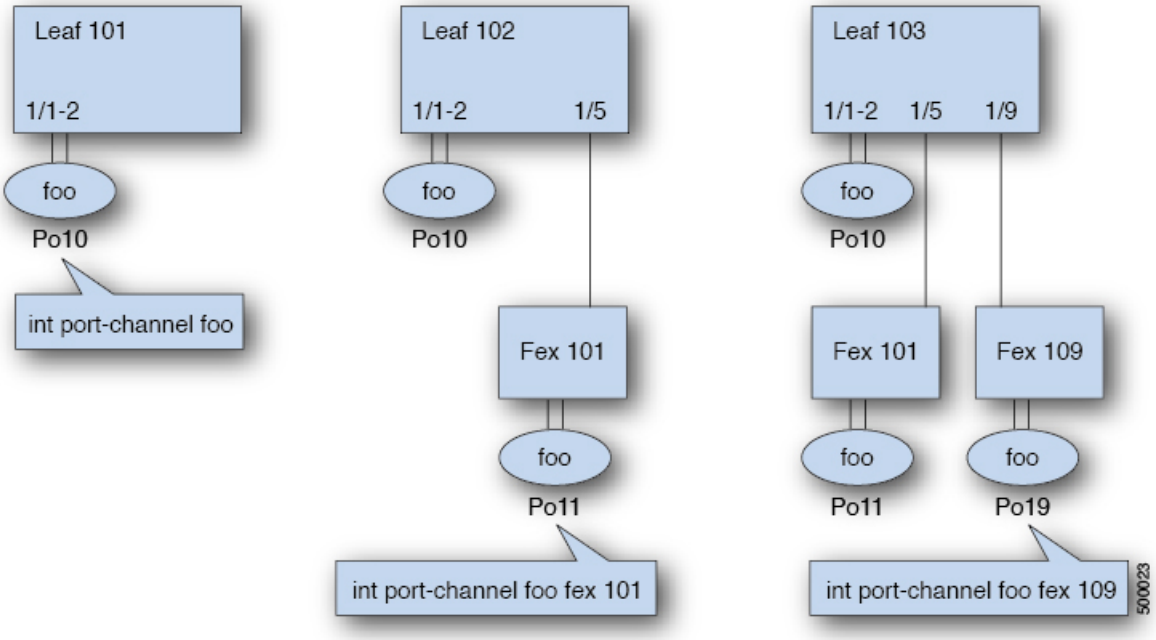
Configure FEX ports attached to leaf nodes. This example shows configuration of speed for interfaces eth1/1-10 in FEX module 101 attached to each of the leaf nodes 102-103. The FEX ID 101 is included in the port identifier. FEX IDs start with 101 and are local to a leaf.

```
apic1(config)# leaf 102-103
apic1(config-leaf)# interface eth 101/1/1-10
apic1(config-leaf-if)# speed 1G
```

Configuring Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI

Port-channels are logical interfaces in NX-OS used to aggregate bandwidth for multiple physical ports and also for providing redundancy in case of link failures. In NX-OS, port-channel interfaces are identified by user-specified numbers in the range 1 to 4096 unique within a node. Port-channel interfaces are either configured explicitly (using the **interface port-channel** command) or created implicitly (using the **channel-group** command). The configuration of the port-channel interface is applied to all the member ports of the port-channel. There are certain compatibility parameters (speed, for example) that cannot be configured on the member ports.

In the ACI model, port-channels are configured as logical entities identified by a name to represent a collection of policies that can be assigned to set of ports in one or more leaf nodes. Such assignment creates one port-channel interface in each of the leaf nodes identified by an auto-generated number in the range 1 to 4096 within the leaf node, which may be same or different among the nodes for the same port-channel name. The membership of these port-channels may be same or different as well. When a port-channel is created on the FEX ports, the same port-channel name can be used to create one port-channel interface in each of the FEX devices attached to the leaf node. Thus, it is possible to create up to N+1 unique port-channel interfaces (identified by the auto-generated port-channel numbers) for each leaf node attached to N FEX modules. This is illustrated with the examples below. Port-channels on the FEX ports are identified by specifying the *fex-id* along with the port-channel name (**interface port-channel foo fex 101** , for example).



- N+1 instances per leaf of port-channel foo are possible when each leaf is connected to N FEX nodes.
- Leaf ports and FEX ports cannot be part of the same port-channel instance.
- Each FEX node can have only one instance of port-channel foo.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	**template port-channel** *channel-name* **Example:** `apic1(config)# template port-channel foo`	Creates a new port-channel or configures an existing port-channel (global configuration).
Step 3	[**no**] **switchport access vlan** *vlan-id* **tenant** *tenant-name* **application** *application-name* **epg** *epg-name* **Example:** `apic1(config-po-ch-if)# switchport access vlan 4 tenant ExampleCorp application Web epg webEpg`	Deploys the EPG with the VLAN on all ports with which the port-channel is associated.

	Command or Action	Purpose

Step 4	**channel-mode active**
	Example:
	`apic1(config-po-ch-if)# channel-mode active`
	Note To enable symmetric hashing, enter the **lacp symmetric-hash** command:
	`apic1(config-po-ch-if)# lacp symmetric-hash`
		Note The **channel-mode** command is equivalent to the mode option in the channel-group command in NX-OS. In ACI, however, this is supported for the port-channel (not on a member port).
		Symmetric hashing is not supported on the following switches:
		• Cisco Nexus 93128TX
		• Cisco Nexus 9372PX
		• Cisco Nexus 9372PX-E
		• Cisco Nexus 9372TX
		• Cisco Nexus 9372TX-E
		• Cisco Nexus 9396PX
		• Cisco Nexus 9396TX

Step 5	**exit** **Example:** `apic1(config-po-ch-if)# exit`	Returns to configure mode.
Step 6	**leaf** *node-id* **Example:** `apic1(config)# leaf 101`	Specifies the leaf switches to be configured. The *node-id* can be a single node ID or a range of IDs, in the form *node-id1 − node-id2* , to which the configuration will be applied.
Step 7	**interface** *type* **Example:** `apic1(config-leaf)# interface ethernet 1/1-2`	Specifies the interface or range of interfaces that you are configuring to the port-channel.
Step 8	[**no**] **channel-group** *channel-name* **Example:** `apic1(config-leaf-if)# channel-group foo`	Assigns the interface or range of interfaces to the port-channel. Use the keyword **no** to remove the interface from the port-channel. To change the port-channel assignment on an interface, you can enter the **channel-group** command without first removing the interface from the previous port-channel.
Step 9	(Optional) **lacp port-priority** *priority* **Example:** `apic1(config-leaf-if)# lacp port-priority`	This setting and other per-port LACP properties can be applied to member ports of a port-channel at this point.

	Command or Action	Purpose
	1000 apic1(config-leaf-if)# **lacp rate fast**	**Note** In the ACI model, these commands are allowed only after the ports are member of a port channel. If a port is removed from a port channel, configuration of these per-port properties are removed as well.

The following table shows various commands for global configurations of port channel properties in the ACI model. These commands can also be used for configuring overrides for port channels in a specific leaf in the (config-leaf-if) CLI mode. The configuration made on the port-channel is applied to all member ports.

CLI Syntax	Feature
[no] speed <speedValue>	Set the speed for port-channel
[no] link debounce time <time>	Set Link Debounce for port-channel
[no] negotiate auto	Configure Negotiate for port-channel
[no] cdp enable	Disable/Enable CDP for port-channel
[no] mcp enable	Disable/Enable MCP for port-channel
[no] lldp transmit	Set the transmit for port-channel
[no] lldp receive	Set the lldp receive for port-channel
spanning-tree <bpduguard | bpdufilter> <enable | disable>	Configure spanning tree BPDU
[no] storm-control level <percentage> [burst-rate <percentage>]	Storm-control configuration (percentage)
[no] storm-control pps <packet-per-second> burst-rate <packets-per-second>	Storm-control configuration (packets-per-second)
[no] channel-mode { active | passive | on| mac-pinning }	LACP mode for the link in port-channel l
[no] lacp min-links <value>	Set minimum number of links
[no] lacp max-links <value>	Set maximum number of links
[no] lacp fast-select-hot-standby	LACP fast select for hot standby ports
[no] lacp graceful-convergence	LACP graceful convergence
[no] lacp load-defer	LACP load defer member ports
[no] lacp suspend-individual	LACP individual Port suspension
[no] lacp port-priority	LACP port priority

CLI Syntax	Feature
[no] lacp rate	LACP rate

Examples

Configure a port channel (global configuration). A logical entity foo is created that represents a collection of policies with two configurations: speed and channel mode. More properties can be configured as required.

Note The channel mode command is equivalent to the mode option in the channel group command in NX-OS. In ACI, however, this supported for the port-channel (not on member port).

```
apic1(config)# template port-channel foo
apic1(config-po-ch-if)# switchport access vlan 4 tenant ExampleCorp application Web epg
webEpg
apic1(config-po-ch-if)# speed 10G
apic1(config-po-ch-if)# channel-mode active
```

Configure ports to a port-channel in a FEX. In this example, port channel foo is assigned to ports Ethernet 1/1-2 in FEX 101 attached to leaf node 102 to create an instance of port channel foo. The leaf node will auto-generate a number, say 1002 to identify the port channel in the switch. This port channel number would be unique to the leaf node 102 regardless of how many instance of port channel foo are created.

Note The configuration to attach the FEX module to the leaf node must be done before creating port channels using FEX ports.

```
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 101/1/1-2
apic1(config-leaf-if)# channel-group foo
```

In Leaf 102, this port channel interface can be referred to as interface port-channel foo FEX 101.

```
apic1(config)# leaf 102
apic1(config-leaf)# interface port-channel foo fex 101
apic1(config-leaf)# shut
```

Configure ports to a port channel in multiple leaf nodes. In this example, port channel foo is assigned to ports Ethernet 1/1-2 in each of the leaf nodes 101-103. The leaf nodes will auto generate a number unique in each node (which may be same or different among nodes) to represent the port-channel interfaces.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo
```

Add members to port channels. This example would add two members eth1/3-4 to the port-channel in each leaf node, so that port-channel foo in each node would have members eth 1/1-4.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo
```

Remove members from port channels. This example would remove two members eth1/2, eth1/4 from the port channel foo in each leaf node, so that port channel foo in each node would have members eth 1/1, eth1/3.

```
 apic1(config)# leaf 101-103
apic1(config-leaf)# interface eth 1/2,1/4
apic1(config-leaf-if)# no channel-group foo
```

Configure port-channel with different members in multiple leaf nodes. This example shows how to use the same port-channel foo policies to create a port-channel interface in multiple leaf nodes with different member ports in each leaf. The port-channel numbers in the leaf nodes may be same or different for the same port-channel foo. In the CLI, however, the configuration will be referred as interface port-channel foo. If the port-channel is configured for the FEX ports, it would be referred to as interface port-channel foo fex <fex-id>.

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 103
apic1(config-leaf)# interface ethernet 1/5-8
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 101/1/1-2
apic1(config-leaf-if)# channel-group foo
```

Configure per port properties for LACP. This example shows how to configure member ports of a port-channel for per-port properties for LACP.

Note In ACI model, these commands are allowed only after the ports are member of a port channel. If a port is removed from a port channel, configuration of these per-port properties would be removed as well.

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# lacp port-priority 1000
apic1(config-leaf-if)# lacp rate fast
```

Configure admin state for port channels. In this example, a port-channel foo is configured in each of the leaf nodes 101-103 using the channel-group command. The admin state of port-channel(s) can be configured in each leaf using the port-channel interface. In ACI model, the admin state of the port-channel cannot be configured in the global scope.

```
// create port-channel foo in each leaf
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo

// configure admin state in specific leaf
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel foo
apic1(config-leaf-if)# shut
```

Override config is very helpful to assign specific vlan-domain, for example, to the port-channel interfaces in each leaf while sharing other properties.

```
// configure a port channel global config
apic1(config)# interface port-channel foo
apic1(config-if)# speed 1G
apic1(config-if)# channel-mode active

// create port-channel foo in each leaf
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# channel-group foo

// override port-channel foo in leaf 102
apic1(config)# leaf 102
apic1(config-leaf)# interface port-channel foo
apic1(config-leaf-if)#  speed 10G
apic1(config-leaf-if)#  channel-mode on
apic1(config-leaf-if)#  vlan-domain dom-foo
```

This example shows how to change port channel assignment for ports using the channel-group command. There is no need to remove port channel membership before assigning to other port channel.

```
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group foo
apic1(config-leaf-if)# channel-group bar
```

Configuring Virtual Port Channels in Leaf Nodes and FEX Devices Using the NX-OS CLI

A virtual port channel (vPC) is an enhancement to port-channels that allows connection of a host or switch to two upstream leaf nodes to improve bandwidth utilization and availability. In NX-OS, vPC configuration is done in each of the two upstream switches and configuration is synchronized using peer link between the switches.

Note When creating a vPC domain between two leaf switches, both switches must be in the same switch generation, one of the following:

- Generation 1 - Cisco Nexus N9K switches without “EX” on the end of the switch name; for example, N9K-9312TX
- Generation 2 – Cisco Nexus N9K switches with “EX” on the end of the switch model name; for example, N9K-93108TC-EX

Switches such as these two are not compatible vPC peers. Instead, use switches of the same generation.

The ACI model does not require a peer link and vPC configuration can be done globally for both the upstream leaf nodes. A global configuration mode called **vpc context** is introduced in ACI and vPC interfaces are represented using a type **interface vpc** that allows global configuration applicable to both leaf nodes.

Two different topologies are supported for vPC in the ACI model: vPC using leaf ports and vPC over FEX ports. It is possible to create many vPC interfaces between a pair of leaf nodes and similarly, many vPC interfaces can be created between a pair of FEX modules attached to the leaf node pairs in a straight-through topology.

vPC considerations include:

- The vPC name used is unique between leaf node pairs. For example, only one vPC 'corp' can be created per leaf pair (with or without FEX).
- Leaf ports and FEX ports cannot be part of the same vPC.
- Each FEX module can be part of only one instance of vPC corp.
- vPC context allows configuration
- The vPC context mode allows configuration of all vPCs for a given leaf pair. For vPC over FEX, the *fex-id* pairs must be specified either for the vPC context or along with the vPC interface, as shown in the following two alternative examples.

```
(config)# vpc context leaf 101 102
(config-vpc)# interface vpc Reg fex 101 101
```

or

```
(config)# vpc context leaf 101 102 fex 101 101
(config-vpc)# interface vpc Reg
```

In the ACI model, vPC configuration is done in the following steps (as shown in the examples below).

Note A VLAN domain is required with a VLAN range. It must be associated with the port-channel template.

1. VLAN domain configuration (global config) with VLAN range
2. vPC domain configuration (global config)

3. Port-channel template configuration (global config)
4. Associate the port-channel template with the VLAN domain
5. Port-channel configuration for vPC (global config)
6. Configure ports to vPC in leaf nodes
7. Configure L2, L3 for vPC in the vpc context

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	**vlan-domain** *name*[**dynamic**] [**type** *domain-type*] **Example:** `apic1(config)# vlan-domain dom1 dynamic`	Configures a VLAN domain for the virtual port-channel (here with a port-channel template).

Step 3	**vlan** *range*
	Example:
	`apic1(config-vlan)# vlan 1000-1999`
	`apic1(config-vlan)# exit`
		Configures a VLAN range for the VLAN domain and exits the configuration mode. The range can be a single VLAN or a range of VLANs.

Step 4	**vpc domain explicit** *domain-id* **leaf** *node-id1 node-id2* **Example:** `apic1(config)# vpc domain explicit 1 leaf 101 102`	Configures a vPC domain between a pair of leaf nodes. You can specify the vPC domain ID in the explicit mode along with the leaf node pairs. Alternative commands to configure a vPC domain are as follows: • **vpc domain** [**consecutive** | **reciprocal**] The consecutive and reciprocal options allow auto configuration of a vPC domain across all leaf nodes in the ACI fabric. • **vpc domain consecutive** *domain-start* **leaf** *start-node end-node* This command configures a vPC domain consecutively for a selected set of leaf node pairs.
Step 5	**peer-dead-interval** *interval* **Example:**	Configures the time delay the Leaf switch waits to restore the vPC before receiving a response from the peer. If it does not receive

	Command or Action	Purpose
	`apic1(config-vpc)# peer-dead-interval 10`	a response from the peer within this time, the Leaf switch considers the peer dead and brings up the vPC with the role as a master. If it does receive a response from the peer it restores the vPC at that point. The range is from 5 seconds to 600 seconds. The default is 200 seconds.
Step 6	**exit** **Example:** `apic1(config-vpc)# exit`	Returns to global configuration mode.
Step 7	**template port-channel** *channel-name* **Example:** `apic1(config)# template port-channel corp`	Creates a new port-channel or configures an existing port-channel (global configuration). All vPCs are configured as port-channels in each leaf pair. The same port-channel name must be used in a leaf pair for the same vPC. This port-channel can be used to create a vPC among one or more pairs of leaf nodes. Each leaf node will have only one instance of this vPC.
Step 8	**vlan-domain member** *vlan-domain-name* **Example:** `vlan-domain member dom1`	Associates the port channel template with the previously configured VLAN domain.
Step 9	**switchport access vlan** *vlan-id* **tenant** *tenant-name* **application** *application-name* **epg** *epg-name* **Example:** `apic1(config-po-ch-if)# switchport access vlan 4 tenant ExampleCorp application Web epg webEpg`	Deploys the EPG with the VLAN on all ports with which the port-channel is associated.
Step 10	**channel-mode active** **Example:** `apic1(config-po-ch-if)# channel-mode active`	**Note** A port-channel must be in active channel-mode for a vPC.
Step 11	**exit** **Example:** `apic1(config-po-ch-if)# exit`	Returns to configure mode.
Step 12	**leaf** *node-id1 node-id2* **Example:** `apic1(config)# leaf 101-102`	Specifies the pair of leaf switches to be configured.

	Command or Action	Purpose
Step 13	**interface** *type leaf/interface-range* **Example:** `apic1(config-leaf)# interface ethernet 1/3-4`	Specifies the interface or range of interfaces that you are configuring to the port-channel.
Step 14	[**no**] **channel-group** *channel-name* **vpc** **Example:** `apic1(config-leaf-if)# channel-group corp vpc`	Assigns the interface or range of interfaces to the port-channel. Use the keyword **no** to remove the interface from the port-channel. To change the port-channel assignment on an interface, you can enter the **channel-group** command without first removing the interface from the previous port-channel. **Note** The **vpc** keyword in this command makes the port-channel a vPC. If the vPC does not already exist, a vPC ID is automatically generated and is applied to all member leaf nodes.
Step 15	**exit** **Example:** `apic1(config-leaf-if)# exit`	
Step 16	**exit** **Example:** `apic1(config-leaf)# exit`	
Step 17	**vpc context leaf** *node-id1 node-id2* **Example:** `apic1(config)# vpc context leaf 101 102`	The vPC context mode allows configuration of vPC to be applied to both leaf node pairs.
Step 18	**interface vpc** *channel-name* **Example:** `apic1(config-vpc)# interface vpc blue fex 102 102`	
Step 19	(Optional) [**no**] **shutdown** **Example:** `apic1(config-vpc-if)# no shut`	Administrative state configuration in the vPC context allows changing the admin state of a vPC with one command for both leaf nodes.

Example

This example shows how to configure a basic vPC.

```
apic1# configure
apic1(config)# vlan-domain dom1 dynamic

apic1(config-vlan)# vlan 1000-1999
apic1(config-vlan)# exit
apic1(config)# vpc domain explicit 1 leaf 101 102
apic1(config-vpc)# peer-dead-interval 10

apic1(config-vpc)# exit
apic1(config)# template port-channel corp
apic1(config-po-ch-if)# vlan-domain member dom1

apic1(config-po-ch-if)# channel-mode active

apic1(config-po-ch-if)# exit
apic1(config)# leaf 101-102
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group corp vpc
apic1(config-leaf-if)# exit
apic1(config)# vpc context leaf 101 102
```

This example shows how to configure vPCs with FEX ports.

```
apic1(config-leaf)# interface ethernet 101/1/1-2
apic1(config-leaf-if)# channel-group Reg vpc
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# interface vpc corp
apic1(config-vpc-if)# exit
apic1(config-vpc)# interface vpc red fex 101 101
apic1(config-vpc-if)# switchport
apic1(config-vpc-if)# exit
apic1(config-vpc)# interface vpc blue fex 102 102
apic1(config-vpc-if)# shut
```

Configuring FEX Connections Using Profiles with the NX-OS Style CLI

Use this procedure to configure FEX connections to leaf nodes using the NX-OS style CLI.

Note Configuring FEX connections with FEX IDs `165` to `199` is not supported in the APIC GUI. To use one of these FEX IDs, configure the profile using the following commands.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`apic1# configure`	
Step 2	**leaf-interface-profile** *name* **Example:** `apic1(config)# leaf-interface-profile fexIntProf1`	Specifies the leaf interface profile to be configured.
Step 3	**leaf-interface-group** *name* **Example:** `apic1(config-leaf-if-profile)# leaf-interface-group leafIntGrp1`	Specifies the interface group to be configured.
Step 4	**fex associate** *fex-id* [**template** *template-type fex-template-name*] **Example:** `apic1(config-leaf-if-group)# fex associate 101`	Attaches a FEX module to a leaf node. Use the optional template keyword to specify a template to be used. If it does not exist, the system creates a template with the name and type you specified.

Example

This merged example configures a leaf interface profile for FEX connections with ID 101.

```
apic1# configure
apic1(config)# leaf-interface-profile fexIntProf1
apic1(config-leaf-if-profile)# leaf-interface-group leafIntGrp1
apic1(config-leaf-if-group)# fex associate 101
```

Reflective Relay (802.1Qbg)

Reflective relay is a switching option beginning with Cisco APIC Release 2.3(1). Reflective relay—the tagless approach of IEEE standard 802.1Qbg—forwards all traffic to an external switch, which then applies policy and sends the traffic back to the destination or target VM on the server as needed. There is no local switching. For broadcast or multicast traffic, reflective relay provides packet replication to each VM locally on the server.

One benefit of reflective relay is that it leverages the external switch for switching features and management capabilities, freeing server resources to support the VMs. Reflective relay also allows policies that you configure on the Cisco APIC to apply to traffic between the VMs on the same server.

In the Cisco ACI, you can enable reflective relay, which allows traffic to turn back out of the same port it came in on. You can enable reflective relay on individual ports, port channels, or virtual port channels as a Layer 2 interface policy using the APIC GUI, NX-OS CLI, or REST API. It is disabled by default.

The term *Virtual Ethernet Port Aggregator* (VEPA) is also used to describe 802.1Qbg functionality.

Reflective Relay Support

Reflective relay supports the following:

- IEEE standard 802.1Qbg tagless approach, known as reflective relay.

Cisco APIC Release 2.3(1) release does not support the IEE standard 802.1Qbg S-tagged approach with multichannel technology.

- Physical domains.

  Virtual domains are not supported.

- Physical ports, port channels (PCs), and virtual port channels (vPCs).

  Cisco Fabric Extender (FEX) and blade servers are not supported. If reflective relay is enabled on an unsupported interface, a fault is raised, and the last valid configuration is retained. Disabling reflective relay on the port clears the fault.

- Cisco Nexus 9000 series switches with *EX* or *FX* at the end of their model name.

Enabling Reflective Relay Using the NX-OS CLI

Reflective relay is disabled by default; however, you can enable it on a port, port channel, or virtual port channel as a Layer 2 interface policy on the switch. In the NX-OS CLI, you can use a template to enable reflective relay on multiple ports or you can enable it on individual ports.

Before you begin

This procedure assumes that you have set up the Cisco Application Centric Infrastructure (ACI) fabric and installed the physical switches.

Procedure

Enable reflective relay on one or multiple ports:

Example:

This example enables reflective relay on a single port:

```
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/2
apic1(config-leaf-if)# switchport vepa enabled
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Example:

This example enables reflective relay on multiple ports using a template:

```
apic1(config)# template policy-group grp1
apic1(config-pol-grp-if)# switchport vepa enabled
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/2-4
apic1(config-leaf-if)# policy-group grp1
```

Example:

This example enables reflective relay on a port channel:

```
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel po2
apic1(config-leaf-if)# switchport vepa enabled
apic1(config-leaf-if)# exit
```

```
apic1(config-leaf)# exit
apic1(config)#
```

Example:

This example enables reflective relay on multiple port channels:

```
apic1(config)# template port-channel po1
apic1(config-if)# switchport vepa enabled
apic1(config-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/3-4
apic1(config-leaf-if)# channel-group po1
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

Example:

This example enables reflective relay on a virtual port channel:

```
apic1(config)# vpc domain explicit 1 leaf 101 102
apic1(config-vpc)# exit
apic1(config)# template port-channel po4
apic1(config-if)# exit
apic1(config)# leaf 101-102
apic1(config-leaf)# interface eth 1/11-12
apic1(config-leaf-if)# channel-group po4 vpc
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# interface vpc po4
apic1(config-vpc-if)# switchport vepa enabled
```

Configuring Policy Groups for Interfaces

In data center networks, typically configuration of many interfaces is the same across multiple nodes. This can be achieved in the ACI Policy Model by creating policy-groups to be shared by groups of interfaces across multiple leaf nodes. The policy-group is identified by a name similar to the port-channel; however, in case of port-channel the policies shared with the group of ports create one logical interface in each leaf while in case of a policy-group, each of the ports sharing the policies are individual physical interfaces. The policy-group concept is very similar to a port-profile in NX-OS.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	**template policy-group** *policy-group-name* **Example:** `apic1(config)# template policy-group pg1`	Creates a new policy group or edits an existing policy group.

	Command or Action	Purpose
Step 3	[**no**] **switchport access vlan** *vlan-id* **tenant** *tenant-name* **application** *application-name* **epg** *epg-name* **Example:** `apic1(config-pol-grp-if)# switchport access vlan 4 tenant ExampleCorp application Web epg webEpg`	

Step 4	(Apply configuration commands)
	Example:
	`apic1(config-pol-grp-if)# speed 10G`
	`apic1(config-pol-grp-if)# cdp enable`
		The table at the end of these steps shows various commands for configurations of policy-group for interfaces.

Step 5	**exit** **Example:** `apic1(config-pol-grp-if)# exit`	Returns to configure mode.
Step 6	**leaf** *node-id* **Example:** `apic1(config)# leaf 101-103`	Specifies the leaf or leafs to be configured. The *node-id* can be a single node ID or a range of IDs, in the form *node-id1* − *node-id2* , to which the configuration will be applied.
Step 7	**interface** *type* **Example:** `apic1(config-leaf)# interface ethernet 1/1-24`	Specifies the interface or range of interfaces to which you will apply the policy group.
Step 8	[**no**] **policy-group** *policy-group-name* [**force**] **Example:** `apic1(config-leaf-if)# policy-group pg1`	Applies the policy-group to the interface or range of interfaces. Use the keyword **no** to remove the policy-group from the interface. Use the keyword **force** to delete any override configurations on the interfaces. If the specified policy-group was not configured prior to this command, this command would not implicitly create the policy-group. However, the policy-group would take effect on the interface after the policy-group has been configured in the global scope. To change the policy-group assignment on an interface, you can enter the **policy-group** command without first removing the previous policy-group from the interface.

	Command or Action	Purpose
		Note If you apply a policy-group to an interface and then assign the interface to a port-channel, the interface will lose the policy-group configuration and the policies in the port-channel will be applied.

The following table shows various commands for configurations of policy-group for interfaces.

CLI Syntax	Feature
[no] speed <speedValue>	Set the speed for Physical Interface
[no] link debounce time <time>	Set link debounce for Physical Interface
[no] negotiate auto	Configure Negotiate for Physical Interface
[no] cdp enable	Disable/Enable CDP for Physical Interface
[no] mcp enable	Disable/Enable MCP for Physical Interface
[no] lldp transmit	Set the LLDP transmit for Physical Interface
[no] lldp receive	Set the LLDP receive for Physical Interface
spanning-tree <bpduguard | bpdufilter> <enable | disable>	Configure spanning tree BPDU
[no] storm-control level <percentage> [burst-rate <percentage>]	Storm-control configuration (percentage)
[no] storm-control pps <packet-per-second> burst-rate <packets-per-second>	Storm-control configuration (packets-per-second)

Example

This example shows how to configure a policy-group and apply it to a range of ports in each of the leaf nodes 101-103. Each of the ports sharing the policy-group in each leaf will have the same configuration as defined in the policy-group pg1.

```
apic1# configure
apic1(config)# template policy-group pg1
apic1(config-pol-grp-if)# switchport access vlan 4 tenant ExampleCorp application Web epg
webEpg
apic1(config-pol-grp-if)# speed 10G
apic1(config-pol-grp-if)# cdp enable
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101-103
apic1(config-leaf)# interface ethernet 1/1-24
apic1(config-leaf-if)# policy-group pg1
```

Configuring Overrides for Interfaces

When policy-groups are used with large number of interfaces, it may be useful to have the option to configure a set of ports for specific properties that will override the configuration in the assigned policy-group. Override configuration is allowed only if the port is assigned to a policy-group. Override configuration is not allowed for member ports of a port-channel. When a port is added to a port-channel, the override configuration is automatically removed. However, during policy-group assignment to a port that has overrides configured, the override configuration is not removed automatically and the user can decide to remove the override configuration with the force option, if required, in the policy-group command.

When a policy-group assignment is removed from a port, the override config, if exists, does not change. Similarly, the override config does not change if the port is assigned to a different policy-group (without the force option). The override config takes effect once configured and it is not removed even if the user assigns default values to all the properties in the override. To remove the override config, the user can reapply the policy-group assignment with force option. The force option, however, is not displayed in the show running-config as it is used to just remove the override config in the ACI model.

In the ACI model, overrides can be configured for a policy which may contain one or more properties. If a policy has more than one property, it is not possible to override only one property within a policy. In the CLI framework, when the user intends to override a property for which the corresponding policy has more than one property, all other properties in the policy except the override property would be implicitly copied to the override configuration to avoid ambiguity. Such implicit copy of configuration would be reflected in the output of show running-config regardless of the value (including default values). Also, the copy is done only once during the configuration of the override policy and any subsequent change to the policy-group for any of the properties in that policy would have no effect on the port(s) on which the override is configured.

If the policy-group assigned to a port is not configured when the override is created, the implicit copy of properties noted above is not possible; instead, default values are assigned to properties in the override config for which the corresponding policy has more than one property. These properties shall not change for the override config when the policy-group is configured afterwards. It is recommended that user create overrides after configuring the policy-group itself or the user may need to configure the overrides in addition to the config in policy-group to get desired configuration if the config for properties in override are set to default implicitly before the configuration of the policy-group with non-default values for those properties.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	**leaf** *node-id* **Example:** `apic1(config)# leaf 102`	Specifies the leaf or leafs to be configured. The *node-id* can be a single node ID or a range of IDs, in the form *node-id1* **-** *node-id2* , to which the configuration will be applied.
Step 3	**interface** *type* **Example:** `apic1(config-leaf)# interface ethernet 1/2`	Specifies the interface or range of interfaces with an override configuration.

	Command or Action	Purpose
Step 4	**policy-group** *policy-group-name* **force** **Example:** `apic1(config-leaf-if)# policy-group pg1 force`	Forces the policy-group to the interface or range of interfaces, deleting any override configurations on the interfaces.

Examples

This example shows how to apply a policy-group and then override the speed configuration for port eth1/1 in leaf node 101. In the ACI model, speed is part of a policy that also contains properties autoneg and link debounce time. As a result, those properties are copied from the speed policy-group when the override of pg1 is configured.

```
apic1# configure
apic1(config)# interface policy-group pg1
apic1(config-pol-grp-if)# speed 10G
apic1(config-pol-grp-if)# cdp enable
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1-2
apic1(config-leaf-if)# policy-group pg1
apic1(config-pol-grp-if)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1
apic1(config-leaf-if)# speed 1G
apic1(config-leaf-if)# show running-config

leaf 101
     interface ethernet 1/1
         policy-group pg1
         speed 1G
         autoneg on
         link debounce time 100

     interface ethernet 1/2
         policy-group pg1
```

This example shows how to remove the override configuration from port eth1/1 in leaf node 101.

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/1
apic1(config-leaf-if)# policy-group pg1 force
apic1(config-leaf-if)# show running-config

leaf 101
     interface ethernet 1/1
         policy-group pg1
```

About Forwarding Error Correction

Forwarding Error Correction (FEC) is a method of obtaining error control in data transmission over an unreliable or noisy channel in which the source (transmitter) encodes the data in a redundant way using Error Correcting Code, and the destination (receiver) recognizes it and corrects the errors without requiring a retransmission. The available options are as follows:

- CL74-FC-FEC—Supports 25 Gbps speed.
- CL91-RS-FEC—Supports 25 and 100 Gbps speeds.
- Disable-FEC—Disables FEC.
- Inherit—The switch uses FEC based on the port transceiver type. All copper (CR4) transceivers have FC-FEC enabled on 25G. All interfaces with 100G transceivers have RS-FEC enabled.

The default is "Inherit".

Note FEC is only configurable on the front port and not on fabric ports.

Configuring FEC Using NX-OS Style CLI

Procedure

	Command or Action	Purpose
Step 1	Enter the configure mode. **Example:** `apic1# configure`	Enters the configuration mode.
Step 2	Enter the switch mode. **Example:** `apic1(config)# leaf 104`	Enters the switch mode.
Step 3	Specify the interface and port. **Example:** `apic1(config-leaf)# int eth 1/4`	Specifies the interface and port.
Step 4	Configure FEC. **Example:** `apic1(config-leaf-if)# forward-error-correction cl91-rs-fec`	Configures RS-FEC. **Note** The default forward-error-correction value is inherit.
Step 5	Exit the interface mode. **Example:** `apic1(config-leaf-if)# exit`	Exits the interface mode.