Configuring Flexible Command Line Interface Configuration Groups

This module describes how to configure and use flexible command line interface (CLI) configuration groups.

Table 1: Feature History for Configuring Flexible CLI Configuration Groups

Release	Modification
Release 4.3.1	Flexible CLI configuration groups were introduced.

This module contains these topics:

Information About Flexible CLI Configuration Groups

Flexible command line interface (CLI) configuration groups provide the ability to minimize repetitive configurations by defining a series of configuration statements in a configuration group, and then applying this group to multiple hierarchical levels in the router configuration tree.

Flexible CLI configuration groups utilize regular expressions that are checked for a match at multiple submodes of the configuration tree based on where the group is applied within the hierarchy. If a match is found at a configuration submode, the corresponding configuration defined in the group is inherited within the matched submode.

Flexible CLI configuration groups also provide an auto-inheritance feature. Auto-inheritance means that any change done to a CLI configuration group is automatically applied to the configuration in any matched submodes that have an apply-group at that hierarchical level. This allows you to make a configuration change or addition once, and have it applied automatically in multiple locations, depending on where you have applied the flexible CLI configuration group.

Flexible Configuration Restrictions

Note these restrictions while using flexible configuration groups:

- Flexible CLI configuration groups are not supported in administration configurations and corresponding apply-groups are not supported in administration configurations.

- Use of preconfigured interfaces in configuration groups is not supported.

- Downgrading from an image that supports configuration groups to an image that does not support them is not supported.

- Access lists, quality of service and route policy configurations do not support the use of configuration groups. Configurations such as these are not valid:

```
group g-not-supported
 ipv4 access-list ...
 !
 ipv6 access-list ...
 !
 ethernet-service access-list ...
 !
 class-map ...
 !
 policy-map ...
 !
 route-policy ...
 !
end-group
```

You can, however, reference such configurations, as shown in this example:

```
group g-reference-ok
 router bgp 6500
  neighbor 7::7
   remote-as 65000
   bfd fast-detect
   update-source Loopback300
   graceful-restart disable
   address-family ipv6 unicast
    route-policy test1 in
    route-policy test2 out
    soft-reconfiguration inbound always
   !
  !
 !
 interface Bundle-Ether1005
   bandwidth 10000000
   mtu 9188
   service-policy output input_1
   load-interval 30
 !
end-group
```

- Some regular expressions are not supported within groups. For example, '?', '|' and '$,' are not supported within groups. Also some characters such as /d and /w are not supported.

◦The choice operator "|" to express multiple match expressions within a regular expression is not supported. For example, these expressions are not supported:

`Gig.*|Gig.*\..*`—To match on either Gigabit Ethernet main interfaces or Gigabit Ethernet sub-interfaces.

`Gig.*0/0/0/[1-5]|Gig.*0/0/0/[10-20]`—To match on either Gig.*0/0/0/[1-5] or Gig.*0/0/0/[10-20].

`'TenGigE.*|POS.*`—To match on either TenGigE.* or POS.* .

• Commands that require a node identifier for the **location** keyword are not supported. For example, this configuration is not supported:
```
lpts pifib hardware police location 0/0/CPU0
```

• Overlapping regular expressions within a configuration group for the same configuration are not supported. For example:

```
group G-INTERFACE
interface 'gig.*a.*'
  mtu 1500
!
interface 'gig.*e.* '
  mtu 2000
!
end-group

interface gigabitethernet0/4/1/0
  apply-group G-INTERFACE
```

This configuration is not permitted because it cannot be determined whether the `interface gigabitethernet0/4/1/0` configuration inherits `mtu 1500` or `mtu 2000`. Both expressions in the configuration group match gigabitethernet0/4/1/0.

• Up to eight configuration groups are permitted on one apply-group command.

• Use multi-line configuration style to configure Flexible CLI configuration groups (like group or apply-group commands) by entering each configuration mode in a separate line, one configuration per line. This is important so that the configuration properties are fully inherited and for better readability during troubleshooting.

Example for a correct configuration style is:

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# router isis IGP
RP/0/RP0/CPU0:router(config-isis)# interface Ten 0/4/0/0
RP/0/RP0/CPU0:router(config-isis-if) # address-family ipv4 unicast
RP/0/RP0/CPU0:router (config-isis-if-af) # metric 123
```

Configuring a Configuration Group

A configuration group includes a series of configuration statements that can be used in multiple hierarchical levels in the router configuration tree. By using regular expressions in a configuration group, you can create generic commands that can be applied in multiple instances.

Use this task to create and use a configuration group.

✎

Note Flexible CLI configurations are not available through the XML interface.

SUMMARY STEPS

1. **configure**
2. **group** *group-name*
3. Enter configuration commands, starting from global configuration mode. Use regular expressions for interface names and other variable instances.
4. **end-group**
5. **apply-group**

DETAILED STEPS

	Command or Action	**Purpose**
Step 1	**configure**	

Step 2	**group** *group-name*
	
	Example:
	`RP/0/RP0/CPU0:router(config)# group`
	`g-interf`
		Specifies a name for a configuration group and enters group configuration mode to define the group. The *group-name* argument can have up to 32 characters and cannot contain any special characters. For information regarding special characters, refer to the *Understanding Regular Expressions, Special Characters, and Patterns* module in the *Cisco IOS XR Getting Started Guide for the Cisco CRS Router*.

Step 3	Enter configuration commands, starting from global configuration mode. Use regular expressions for interface names and other variable instances.
	
	Example:
	`RP/0/RP0/CPU0:router(config)# group`
	`g-interf`
	`RP/0/RP0/CPU0:router(config-GRP)#`
	`interface 'GigabitEthernet.*'`
	`RP/0/RP0/CPU0:router(config-GRP-if)# mtu`
	` 1500`
		Specifies the configuration statements that you want included in this configuration group.
		
		For more information regarding the use of regular expressions, see Regular Expressions in Configuration Groups, on page 8. This example is applicable to all Gigabit Ethernet interfaces.

Step 4	**end-group**
	
	Example:
	`RP/0/RP0/CPU0:router(config-GRP-if)#`
	`end-group`
		Completes the configuration of a configuration group and exits to global configuration mode.

Step 5	**apply-group**
	
	Example:
	`RP/0/RP0/CPU0:router(config)# interface`
	` GigabitEthernet0/2/0/0`
	`RP/0/RP0/CPU0:router(config-if)#`
		Adds the configuration of the configuration group into the router configuration applicable at the location that the group is applied. Groups can be applied in multiple locations, and their effect depends on the location and context.
		
		The MTU value from the group g-interf is applied to the interface GigabitEthernet0/2/0/0. If this group is applied in global configuration

Command or Action	Purpose
`apply-group g-interf`	mode, the MTU value is inherited by all Gigabit Ethernet interfaces that do not have an MTU value configured.

Simple Configuration Group: Example

This example shows how to use configuration groups to add a global configuration to the system:

```
RP/0/RP0/CPU0:router(config)# group g-logging
RP/0/RP0/CPU0:router(config-GRP)# logging trap notifications
RP/0/RP0/CPU0:router(config-GRP)# logging console debugging
RP/0/RP0/CPU0:router(config-GRP)# logging monitor debugging
RP/0/RP0/CPU0:router(config-GRP)# logging buffered 10000000
RP/0/RP0/CPU0:router(config-GRP)# end-group

RP/0/RP0/CPU0:router(config)# apply-group g-logging
```

When this configuration is committed, all commands contained in the g-logging configuration group are committed.

Configuration Group Applied to Different Places: Example

Configuration groups can be applied to different places, and their effect depends on the context within which they are applied. Consider this configuration group:

```
RP/0/RP0/CPU0:router(config)# group g-interfaces
RP/0/RP0/CPU0:router(config-GRP)# interface 'FastEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 1500
RP/0/RP0/CPU0:router(config-GRP-if)# exit
RP/0/RP0/CPU0:router(config-GRP)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 1000
RP/0/RP0/CPU0:router(config-GRP-if)# exit
RP/0/RP0/CPU0:router(config-GRP)# interface 'POS.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 2000
RP/0/RP0/CPU0:router(config-GRP-if)# end-group
```

This group can be applied to Fast Ethernet, Gigabit Ethernet or POS interfaces, and in each instance the applicable MTU is applied. For instance, in this example, the Gigabit Ethernet interface is configured to have an MTU of 1000:

```
RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/2/0/0
RP/0/RP0/CPU0:router(config-if)# apply-group g-interfaces
RP/0/RP0/CPU0:router(config-if)# ipv4 address 2.2.2.2 255.255.255.0
```

In this example, the Fast Ethernet interface is configured to have an MTU of 1500:

```
RP/0/RP0/CPU0:router(config)# interface FastEthernet0/2/0/0
RP/0/RP0/CPU0:router(config-if)# apply-group g-interfaces
RP/0/RP0/CPU0:router(config-if)# ipv4 address 3.3.3.3 255.255.255.0
```

The same configuration group is used in both cases, but only the applicable configuration statements are used.

Verifying the Configuration of Configuration Groups

Use this task to verify the router configuration using configuration groups:

SUMMARY STEPS

1. **show running-config group** [*group-name*]
2. **show running-config**
3. **show running-config inheritance**
4. **show running-config interface x/y/z inheritance** *config-command*

DETAILED STEPS

	Command or Action	**Purpose**

Step 1	**show running-config group** [*group-name*]
	
	Example:
	
	`RP/0/RP0/CPU0:router# show running-config group`
	
	`group g-int-ge`
	` interface 'GigabitEthernet.*'`
	`  mtu 1000`
	`   negotiation auto`
	` !`
	`end-group`
		Displays the contents of a specific or all configured configuration groups.

Step 2	**show running-config**
	
	Example:
	
	`RP/0/RP0/CPU0:router# show running-config`
	
	`group G-INTERFACE-MTU`
	` interface 'POS.*'`
	`  mtu 1500`
	`  !`
	`end-group`
	
	`interface POS0/4/1/0`
	` apply-group G-INTERFACE-MTU`
	`!`
	`interface POS0/4/1/1`
	` apply-group G-INTERFACE-MTU`
	` mtu 2000`
	`!`
		Displays the running configuration. Any applied groups are displayed. There is no indication as to whether these configuration groups affect the actual configuration or not. In this example, although the group G-INTERFACE-MTU is applied to POS0/4/1/1, the configured MTU value is 2000 and not 1500. This happens if the command **mtu 2000** is configured directly on the interface. An actual configuration overrides a configuration group configuration if they are the same.

Step 3	**show running-config inheritance**
	
	Example:
	
	`RP/0/RP0/CPU0:router# show running-config inheritance`
	`.`
	`.`
	`group G-INTERFACE-MTU`
		Displays the inherited configuration where ever a configuration group has been applied.

Command or Action	Purpose

<pre>interface 'POS.*'
 mtu 1500
 !
end-group
.
.
interface POS0/4/1/0
 ## Inherited from group G-INTERFACE-MTU
 mtu 1500
!
interface POS0/4/1/1
 mtu 2000
!
.
.</pre>	

Step 4 **show running-config interface x/y/z inheritance** *config-command* **Example:** <pre>RP/0/RP0/CPU0:router# show running-config interface pos0/4/1/0 inheritance [detail]
 	
interface POS0/4/1/0
 ## Inherited from group G-INTERFACE-MTU
 mtu 1500</pre>
	Displays the inherited configuration for a specific configuration command.

Apply Groups Priority Inheritance

The inheritance is supported according to the priority.

Apply groups priority inheritance helps flexible configuration groups handle common configuration statements between groups. When multiple configuration groups have common configuration statements, the inheritance priority is configuration statements present in inner groups have precedence over configuration statements present in outer groups. Tiebreaker is determined by the system order (lexicographical) of the regular expressions. User defined order of commands are not accepted.

For example, a configuration statement in configuration group ONE has precedence over any other group. A configuration statement in configuration group SEVEN is used only if it is not contained in any other group. Within a configuration group, inheritance priority is lengthiest match.

```
apply-group SIX SEVEN
   router ospf 0
   apply-group FOUR FIVE
   area 0
   apply-group THREE
   interface GigabitEthernet 0/0/0/0
      apply-group ONE TWO
  !
 !
!
```

The above example states two scenarios. Inner most group (**apply-group ONE TWO**) has the highest priority.

Case 1

In the first scenario it shows which group gets the first priority. The example states which group is applied between different configuration groups (different groups- nothing in common between them). While applying the group one (ONE TWO), all the seven groups that matches to the interface `interface GigabitEthernet 0/0/0/0` will be applied.

Case 2

In the case when all these groups (mentioned above) have same (common) configuration, group one will be active. The `apply-group ONE TWO` will be active. If group ONE is deleted then group TWO will be active.

Regular Expressions in Configuration Groups

Regular expressions are used in configuration groups to make them widely applicable. Portable Operating System Interface for UNIX (POSIX) 1003.2 regular expressions are supported in the names of configuration statements. Single quotes must be used to delimit a regular expression.

For general information regarding regular expressions, refer to the *Understanding Regular Expressions, Special Characters, and Patterns* module in the *Cisco IOS XR Getting Started Guide for the Cisco CRS Router*.

Note
Not all POSIX regular expressions are supported. Refer to for more information.

Regular Expressions for Interface Identifiers

Configuration groups do not accept exact interface identifiers. You must use a regular expression to identify a group of interfaces that are applicable to the configuration group. The regular expression '.*' is not allowed. You must begin the regular expression for an interface identifier with an unambiguous word, followed by the regular expression. For example, to configure Gigabit Ethernet interfaces, use the regular expression 'GigabitEthernet.*'.

To display a list of available interface types for your router configuration, enter **interface ?** at the configuration group prompt:

```
RP/0/RP0/CPU0:router(config-GRP)# interface ?

  ATM             'RegExp': ATM Network Interface(s)
  BVI             'RegExp': Bridge-Group Virtual Interface
  Bundle-Ether    'RegExp': Aggregated Ethernet interface(s)
  Bundle-POS      'RegExp': Aggregated POS interface(s)
  GigabitEthernet 'RegExp': GigabitEthernet/IEEE 802.3 interface(s)
  IMA             'RegExp': ATM Network Interface(s)
  Loopback        'RegExp': Loopback interface(s)
  MgmtEth         'RegExp': Ethernet/IEEE 802.3 interface(s)
  Multilink       'RegExp': Multilink network interface(s)
  Null            'RegExp': Null interface
  POS             'RegExp': Packet over SONET/SDH network interface(s)
  PW-Ether        'RegExp': PWHE Ethernet Interface
  PW-IW           'RegExp': PWHE VC11 IP Interworking Interface
  Serial          'RegExp': Serial network interface(s)
  tunnel-ip       'RegExp': GRE/IPinIP Tunnel Interface(s)
  tunnel-mte      'RegExp': MPLS Traffic Engineering P2MP Tunnel interface(s)
  tunnel-te       'RegExp': MPLS Traffic Engineering Tunnel interface(s)
  tunnel-tp       'RegExp': MPLS Transport Protocol Tunnel interface
```

✎

Note Although you are required to enter only enough characters for the interface type to be unique, it is recommended that you enter the entire phrase. All interface types used in regular expressions are case-sensitive.

To specify a subinterface, prefix the expression with the characters \. (backslash period). For example, use `interface 'GigabitEthernet.*\..*'` to configure all Gigabit Ethernet subinterfaces.

You can specify Layer 2 transport interfaces or point-to-point interfaces as shown in these examples:

```
group g-l2t
  interface 'Gi.*\..*' l2transport
.
.
end-group
group g-ptp
  interface 'Gi.*\..*' point-to-point
.
.
end-group
```

Regular Expressions for an OSPF Configuration

Exact router process names and OSPF areas cannot be used. You must use a regular expression to specify a process name or group of OSPF areas. To specify that the OSFP area can be either a scalar value or an IP address, use the regular expression '.*', as in this example:

```
group g-ospf
router ospf '.*'
area '.*'
mtu-ignore enable
!
!
end-group
```

To specify that the OSPF area must be an IP address, use the expression '\.' as in this example:

```
group g-ospf-ipaddress
router ospf '.*\..*\..*\..*'
area '.*'
passive enable
!
!
 end-group
```

To specify that the OSPF area must be a scalar value, use the expression '1.*', as in this example:

```
group g-ospf-match-number
router ospf '.*'
area '1.*'
passive enable
!
!
end-group
```

Regular Expressions for a BGP AS

Exact BGP AS values cannot be used in configuration groups. Use a regular expression to specify either AS plain format, or AS dot format as in the format X.Y. To match AS plain format instances, use a simple regular

expression. To match AS dot format instances, use two regular expressions separated by a dot, as shown in this example:

```
group g-bgp
router bgp '*'.'*'
address-family ipv4 unicast
!
!
end-group
```

Regular Expressions for ANCP

Exact Access Node Control Protocol (ANCP) sender-name identifiers cannot be used in configuration groups. Because the sender name argument can be either an IP address or a MAC address, you must specify in the regular expression which one is being used. Specify an IP address as `'.*\..*\..*\..*'`; specify a MAC address as `'.*\..*\..*'`.

Resolving to a Uniform Type

Regular expressions must resolve to a uniform type. This is an example of an illegal regular expression:

```
group g-invalid
 interface '.*'
  bundle port-priority 10
 !
interface '.*Ethernet.*'
  bundle port-priority 10
 !
end-group
```

In this example, the **bundle** command is supported for interface type GigabitEthernet but not for interface type 'FastEthernet'. The regular expressions '.*' and '.*Ethernet.*' match both GigabitEthernet and FastEthernet types. Because the **bundle** command is not applicable to both these interface types, they do not resolve to a uniform type and therefore the system does not allow this configuration.

Note If the system cannot determine from the regular expression what the configuration should be, the expression is not considered valid.

Note The regular expression '.*' is not allowed when referring to an interface identifier. You must begin the regular expression for an interface identifier with an unambiguous word, followed by the regular expression. Refer to *Regular Expressions for Interface Identifiers* in this section for more information.

Configuration Examples Using Regular Expressions

Configuration Group with Regular Expression: Example

This example shows the definition of a configuration group for configuring Gigabit Ethernet interfaces with ISIS routing parameters, using regular expressions for the exact interface:

```
RP/0/RP0/CPU0:router(config)# group g-isis-gige
RP/0/RP0/CPU0:router(config-GRP)# router isis '.*'
RP/0/RP0/CPU0:router(config-GRP-isis)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-isis-if)# lsp-interval 20
RP/0/RP0/CPU0:router(config-GRP-isis-if)# hello-interval 40
RP/0/RP0/CPU0:router(config-GRP-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-GRP-isis-if-af)# metric 10
RP/0/RP0/CPU0:router(config-GRP-isis-if-af)# end-group
RP/0/RP0/CPU0:router(config)#
```

To illustrate the use of this configuration group, assume that you want to configure these Gigabit Ethernet interfaces with the ISIS routing parameters:

```
router isis green
 interface GigabitEthernet0/0/0/0
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/1
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/2
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/3
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
!
```

There are three possible ways to use the configuration group to configure these interfaces. The first is by applying the group within the interface configuration, as shown here:

```
router isis green
 interface GigabitEthernet0/0/0/0
  apply-group g-isis-gige
  !
 !
 interface GigabitEthernet0/0/0/1
  apply-group g-isis-gige
  !
 !
 interface GigabitEthernet0/0/0/2
```

```
 apply-group g-isis-gige
 !
!
interface GigabitEthernet0/0/0/3
 apply-group g-isis-gige
 !
 !
```

In this situation, only the interfaces to which you apply the configuration group inherit the configuration.

The second way to configure these interfaces using the configuration group is to apply the configuration group within the **router isis** configuration, as shown here:

```
router isis green
 apply-group g-isis-gige
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/1
 !
 interface GigabitEthernet0/0/0/2
 !
 interface GigabitEthernet0/0/0/3
 !
!
```

In this way, any other Gigabit Ethernet interfaces that you configure in the ISIS green configuration also inherit these configurations.

The third way to configure these interfaces using the configuration group is to apply the group at the global level as shown here:

```
apply-group g-isis-gige
router isis green
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/1
 !
 interface GigabitEthernet0/0/0/2
 !
 interface GigabitEthernet0/0/0/3
 !
!
```

In this example, the configuration of the group is applied to all Gigabit Ethernet interfaces configured for ISIS.

Configuration Group Inheritance with Regular Expressions: Example

Local Configuration Has Precedence Over Configuration Group

An explicit configuration takes precedence over a configuration applied from a configuration group. For example, assume that this configuration is running on the router:

```
router ospf 100
 packet-size 1000
!
```

You configure this configuration group, apply it, and commit it to the configuration.

```
RP/0/RP0/CPU0:router(config)# group g-ospf
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# nsf cisco
RP/0/RP0/CPU0:router(config-GRP-ospf)# packet-size 3000
RP/0/RP0/CPU0:router(config-GRP-ospf)# end-group
```

```
RP/0/RP0/CPU0:router(config)# apply-group g-ospf
```

The result is effectively this configuration:

```
router ospf 100
 packet-size 1000
 nsf cisco
```

Note that `packet-size 3000` is not inherited from the configuration group because the explicit local configuration has precedence.

Compatible Configuration Is Inherited

The configuration in the configuration group must match the configuration on the router to be inherited. If the configuration does not match, it is not inherited. For example, assume that this configuration is running on the router:

```
router ospf 100
 auto-cost disable
!
```

You configure this configuration and commit it to the configuration.

```
RP/0/RP0/CPU0:router(config)# group g-ospf
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# area '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# packet-size 2000
RP/0/RP0/CPU0:router(config-GRP-ospf)# end-group

RP/0/RP0/CPU0:router(config)# apply-group g-ospf

RP/0/RP0/CPU0:router(config)# router ospf 200
RP/0/RP0/CPU0:router(config-ospf)# area 1
```

The result is effectively this configuration:

```
router ospf 100
  auto-cost disable

router ospf 200
  area 1
  packet-size 2000
```

The packet size is inherited by the ospf 200 configuration, but not by the ospf 100 configuration because the area is not configured.

Layer 2 Transport Configuration Group: Example

This example shows how to configure and apply a configuration group with Layer 2 transport subinterfaces:

```
RP/0/RP0/CPU0:router(config)# group g-l2trans-if
RP/0/RP0/CPU0:router(config-GRP)# interface 'TenGigE.*\..*' l2transport
RP/0/RP0/CPU0:router(config-GRP)# mtu 1514
RP/0/RP0/CPU0:router(config-GRP)# end-group

RP/0/RP0/CPU0:router(config)# interface TenGigE0/0/0/0.1 l2transport
RP/0/RP0/CPU0:router(config-if)# apply-group g-l2trans-if
```

When this configuration is committed, the Ten Gigabit Ethernet interface 0/0/0/0.1 inherits the 1514 MTU value. This is the output displayed from the **show running-config inheritence** command for the Ten Gigabit Ethernet interface:

```
interface TenGigE0/0/0/0.1 l2transport
 ## Inherited from group g-l2trans-if
 mtu 1514
!
```

Configuration Group Precedence: Example

When similar configuration statements are contained in multiple configuration groups, groups applied in inner configuration modes take precedence over groups applied in outer modes. This example shows two configuration groups that configure different cost values for OSPF.

```
RP/0/RP0/CPU0:router(config)# group g-ospf2
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# area '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# cost 2
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# end-group

RP/0/RP0/CPU0:router(config)# group g-ospf100
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# area '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# cost 100
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# end-group
```

If these configuration groups are applied as follows, the cost 2 specified in g-ospf2 is inherited by OSPF area 0 because the group is applied in a more inner configuration mode. In this case, the configuration in group g-ospf100 is ignored.

```
RP/0/RP0/CPU0:router(config)# router ospf 0
RP/0/RP0/CPU0:router(config-ospf)# apply-group g-ospf100
RP/0/RP0/CPU0:router(config-ospf)# area 0
RP/0/RP0/CPU0:router(config-ospf-ar)# apply-group g-ospf2
```

Changes to Configuration Group are Automatically Inherited: Example

When you make changes to a configuration group that is committed and applied to your router configuration, the changes are automatically inherited by the router configuration. For example, assume that this configuration is committed:

```
group g-interface-mtu
 interface 'POS.*'
  mtu 1500
 !
end-group

interface POS0/4/1/0
 apply-group g-interface-mtu
!
```

Now you change the configuration group as in this example:

```
RP/0/RP0/CPU0:router(config)# group g-interface-mtu
RP/0/RP0/CPU0:router(config-GRP)# interface 'POS.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 2000
```

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

When this configuration group is committed, the MTU configuration for interface POS0/4/1/0 is automatically updated to 2000.

Configuration Examples for Flexible CLI Configuration

Basic Flexible CLI Configuration: Example

This example shows that the Media Access Control (MAC) accounting configuration from the gd21 configuration group is applied to all Gigabit Ethernet interfaces in slot 2, ports 1 to 9.

1 Configure the configuration group that configures MAC accounting:

```
RP/0/RP0/CPU0:router# show running group gd21

group gd21
interface 'GigabitEthernet0/0/0/2[1-9]'
description general interface inheritance check
load-interval 30
mac-accounting ingress
mac-accounting egress
!
end-group
```

2 Check that the corresponding apply-group is configured in global configuration or somewhere in the hierarchy:

```
RP/0/RP0/CPU0:router# show running | in apply-group gd21

Building configuration...
apply-group gd21
```

3 Check the concise local view of the configuration of some of the interfaces:

```
RP/0/RP0/CPU0:router# show running interface

interface GigabiEthernet0/0/0/21
!
interface GigabitEthernet0/0/0/22
!
```

4 Verify that the match and inheritance occur on these interfaces:

```
RP/0/RP0/CPU0:router# show running inheritance interface

interface GigabitEthernet0/0/0/21
## Inherited from group gd21
description general interface inheritance check
## Inherited from group gd21
load-interval 30
## Inherited from group gd21
mac-accounting ingress
## Inherited from group gd21
mac-accounting egress
!
Interface GigabitEthernet0/0/0/22
## Inherited from group gd21
description general interface inheritance check
## Inherited from group gd21
load-interval 30
```

```
                    ## Inherited from group gd21
                    mac-accounting ingress
                    ## Inherited from group gd21
                    mac-accounting egress
                    !
                    !
```

5 Verify that the inherited configuration actually takes effect:

```
RP/0/RP0/CPU0:router# show mac gigabitEthernet0/0/0/21

GigabitEthernet0/0/0/21
  Input (96 free)
    6c9c.ed35.90fd:  1271 packets, 98426 bytes
        Total: 1271 packets, 98426 bytes
  Output (96 free)
    6c9c.ed35.90fd:   774 packets, 63265 bytes
        Total: 774 packets, 63264 bytes
```

Interface MTU Settings for Different Interface Types: Example

This example shows that an MTU value is configured on different interface types.

1 Configure an interface MTU configuration group and apply this group:

```
RP/0/RP0/CPU0:router# show running group l2tr

group l2tr
interface 'GigabitEthernet0/0/0/3.*'
mtu 1500
!
interface 'GigabitEthernet0/0/0/9\..*'
mtu 1400
!
interface 'GigabitEthernet0/0/0/9\..*' l2transport
mtu 1400
!
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group l2tr
```

2 Check the concise view and the inheritance view of the various interfaces:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/30

interface GigabitEthernet0/0/0/30
!
RP/0/RP0/CPU0:router# show running inheritance interface gigabitEthernet0/0/0/30

interface GigabitEthernet0/0/0/30
## Inherited from group l2tr
mtu 1500
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/9.800

interface GigabitEthernet0/0/0/9.800
  encapsulation dot1q 800
!

RP/0/RP0/CPU0:router# show running inheritance interface gigabitEthernet0/0/0/9.800
```

```
interface GigabitEthernet0/0/0/9.800
## Inherited from group l2tr
mtu 1400
encapsulation dot1q800
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/9.250

interface GigabitEthernet0/0/0/9.250 l2transport
  encapsulation dot1q 250
!

RP/0/RP0/CPU0:router# show running inheritance interface gigabitEthernet0/0/0/9.800

interface GigabitEthernet0/0/0/9.250 l2transport
encapsulation dot1q250
## Inherited from group l2tr
mtu 1400
!
```

3 Verify that the correct values from the group do take effect:

```
RP/0/RP0/CPU0:router# show interface gigabitEthernet 0/0/0/30

GigabitEthernet0/0/0/30 is down, line protocol is down
  Interface state transitions: 0
  Hardware is GigabitEthernet, address is 0026.9824.ee56 (bia 0026.9824.ee56)
  Internet address is Unknown
  MTU 1500 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 1000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  loopback not set,
  Last input never, output never
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
           0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out

RP/0/RP0/CPU0:router# show interface gigabitEthernet 0/0/0/9.801

GigabitEthernet0/0/0/9.801 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0026.9824.ee41
  Internet address is Unknown
  MTU 1400 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 801,  loopback not set,
  Last input never, output never
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets


RP/0/RP0/CPU0:router# show interface gigabitEthernet 0/0/0/9.250

GigabitEthernet0/0/0/9.250 is up, line protocol is up
```

```
   Interface state transitions: 1
   Hardware is VLAN sub-interface(s), address is 0026.9824.ee41
   Layer 2 Transport Mode
   MTU 1400 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
      reliability Unknown, txload Unknown, rxload Unknown
   Encapsulation 802.1Q Virtual LAN,
     Outer Match: Dot1Q VLAN 250
     Ethertype Any, MAC Match src any, dest any
   loopback not set,
   Last input never, output never
   Last clearing of "show interface" counters never
      0 packets input, 0 bytes
      0 input drops, 0 queue drops, 0 input errors
      0 packets output, 0 bytes

      0 output drops, 0 queue drops, 0 output errors
```

ACL Referencing: Example

This example shows how to reference access-lists on a number of interfaces using configuration groups.

1 Configure the configuration group and apply-group:

```
RP/0/RP0/CPU0:router# show running group acref

group acref
 interface 'GigabitEthernet0/0/0/3.*'
  ipv4 access-group adem ingress
  ipv4 access-group adem egress
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group isis l2tr isis2 mpp bundle1 acref
```

2 Check the concise and inheritance view of the matching configurations:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/30

interface GigabitEthernet0/0/0/30
!

RP/0/RP0/CPU0:router# show running inheritance interface GigabitEthernet 0/0/0/30

interface GigabitEthernet0/0/0/30
 ## Inherited from group l2tr
 mtu 1500
 ## Inherited from group acref
 ipv4 access-group adem ingress
 ## Inherited from group acref
 ipv4 access-group adem egress
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/31

interface GigabitEthernet0/0/0/31
!

RP/0/RP0/CPU0:router# show running inheritance interface GigabitEthernet 0/0/0/31

interface GigabitEthernet0/0/0/31
 ## Inherited from group l2tr
 mtu 1500
```

```
                        ## Inherited from group acref
                        ipv4 access-group adem ingress
                        ## Inherited from group acref
                        ipv4 access-group adem egress
```

3 Check that the ACL group configuration actually got configured by using a traffic generator and watching that denied traffic is dropped.

Local Configuration Takes Precedence: Example

This example illustrates that local configurations take precedence when there is a discrepancy between a local configuration and the configuration inherited from a configuration group.

1 Configure a local configuration in a configuration submode with an access list:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/39

interface GigabitEthernet0/0/0/39
 ipv4 access-group smany ingress
 ipv4 access-group smany egress
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/38

interface GigabitEthernet0/0/0/38
!

RP/0/RP0/CPU0:router# show running ipv4 access-list smany

ipv4 access-list smany
 10 permit ipv4 any any
!

RP/0/RP0/CPU0:router# show running ipv4 access-list adem

ipv4 access-list adem
 10 permit ipv4 21.0.0.0 0.255.255.255 host 55.55.55.55
 20 deny ipv4 any any
!
```

2 Configure and apply the access list group configuration:

```
RP/0/RP0/CPU0:router# show running group acref

group acref
 interface 'GigabitEthernet0/0/0/3.*'
  ipv4 access-group adem ingress
  ipv4 access-group adem egress
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...
apply-group isis l2tr isis2 mpp bundle1 acref
```

3 Check the concise and inheritance views for the matching interface where the access list reference is configured locally:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/39

interface GigabitEthernet0/0/0/39
 ipv4 access-group smany ingress
 ipv4 access-group smany egress
!
```

```
RP/0/RP0/CPU0:router# show running inheritance interface gigabitEthernet 0/0/0/39

interface GigabitEthernet0/0/0/39
 ## Inherited from group l2tr
 mtu 1500
 ipv4 access-group smany ingress
 ipv4 access-group smany egress      << no config inherited, local config prioritized
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/38

interface GigabitEthernet0/0/0/38
!

RP/0/RP0/CPU0:router# show running inheritance interface gigabitEthernet 0/0/0/38

interface GigabitEthernet0/0/0/38
 ## Inherited from group l2tr
 mtu 1500
 ## Inherited from group acref
 ipv4 access-group adem ingress
 ## Inherited from group acref
 ipv4 access-group adem egress
!
```

4 Use a traffic generator to verify that the traffic pattern for interface GigabitEthernet0/0/0/39 gets acted on by the access list in the local configuration (smany) and not according to the inherited referenced access list (adem).

ISIS Hierarchical Configuration: Example

This example illustrates inheritance and priority handling with two ISIS groups using an ISIS configuration.

1 Configure the local ISIS configuration:

```
RP/0/RP0/CPU0:router# show running router isis

router isis vink
 net 49.0011.2222.2222.2222.00
 address-family ipv4 unicast
  mpls traffic-eng level-1-2
  mpls traffic-eng router-id Loopback0
  redistribute connected
 !
 interface Bundle-Ether1
  address-family ipv4 unicast
  !
 !
 interface Bundle-Ether2
 !
 interface Loopback0
 !
 interface TenGigE0/2/0/0.3521
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3522
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3523
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3524
  address-family ipv4 unicast
```

```
  !
 !
 interface TenGigE0/2/0/0.3525
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3526
 !
 interface TenGigE0/2/0/0.3527
 !
 interface TenGigE0/2/0/0.3528
 !
 interface TenGigE0/2/0/1
  address-family ipv4 unicast
  !
 !
!
```

2 Configure two ISIS groups and apply these to the configuration:

```
RP/0/RP0/CPU0:router# show running group isis

group isis
 router isis '.*'
  address-family ipv4 unicast
   mpls traffic-eng level-1-2
   mpls traffic-eng router-id Loopback0
   redistribute connected
   redistribute ospf 1 level-1-2
  !
  interface 'TenGig.*'
   lsp-interval 40
   hello-interval 15
   address-family ipv4 unicast
    metric 50
   !
  !
  interface 'Bundle-Ether.*'
   address-family ipv4 unicast
    metric 55
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running group isis2

group isis2
 router isis '.*'
 !
 router isis '^(vink)'
  address-family ipv4 unicast
  !
  interface '(^Ten)Gig.*'
  !
  interface '^(Ten)Gig.*'
   address-family ipv4 unicast
    metric 66
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group isis  l2tr  isis2  mpp  bundle1  acref
```

3   Check the inheritance view of the ISIS configuration:

```
RP/0/RP0/CPU0:router# show running inheritance router isis

router isis vink
 net 49.0011.2222.2222.2222.00
 address-family ipv4 unicast
  mpls traffic-eng level-1-2
  mpls traffic-eng router-id Loopback0
  redistribute connected
  ## Inherited from group isis
  redistribute ospf 1 level-1-2
 !
 interface Bundle-Ether1
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 55
  !
 !
 interface Bundle-Ether2
  ## Inherited from group isis
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 55
  !
 !
 interface Loopback0
 !
 interface TenGigE0/2/0/0.3521
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/0.3522
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/0.3523
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/0.3524
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/0.3525
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
```

```
  hello-interval 15
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/0.3526
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  ## Inherited from group isis
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/0.3527
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  ## Inherited from group isis
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/0.3528
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  ## Inherited from group isis
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
 interface TenGigE0/2/0/1
  ## Inherited from group isis
  lsp-interval 40
  ## Inherited from group isis
  hello-interval 15
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 50
  !
 !
!
```

4 Verify the actual functionality:

```
RP/0/RP0/CPU0:router# show isis interface TenGigE0/2/0/0.3528 | inc Metric

Metric (L1/L2):        50/50
```

OSPF Hierarchy: Example

This example illustrates hierarchical inheritance and priority. The configuration that is lower in hierarchy gets the highest priority.

1 Configure a local OSPF configuration:

```
RP/0/RP0/CPU0:router# show running router ospf

router ospf 1
```

```
   apply-group go-c
   nsr
   router-id 121.121.121.121
   nsf cisco
   redistribute connected
   address-family ipv4 unicast
   area 0
    apply-group go-b
    interface GigabitEthernet0/0/0/0
     apply-group go-a
    !
    interface GigabitEthernet0/0/0/1
    !
    interface GigabitEthernet0/0/0/3
    !
    interface GigabitEthernet0/0/0/4
    !
    interface GigabitEthernet0/0/0/21
     bfd minimum-interval 100
     bfd fast-detect
     bfd multiplier 3
    !
    interface TenGigE0/2/0/0.3891
    !
    interface TenGigE0/2/0/0.3892
    !
    interface TenGigE0/2/0/0.3893
    !
    interface TenGigE0/2/0/0.3894
    !
   !
  !
  router ospf 100
  !
  router ospf 1000
  !
  router ospf 1001
  !
```

2   Configure a configuration group and apply it in a configuration submode:

```
RP/0/RP0/CPU0:router# show running group go-a

group go-a
 router ospf '.*'
  area '.*'
   interface 'Gig.*'
    cost 200
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running group go-b

group go-b
 router ospf '.*'
  area '.*'
   interface 'Gig.*'
    cost 250
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running group go-c

group go-c
 router ospf '.*'
  area '.*'
   interface 'Gig.*'
    cost 300
   !
```

```
   !
  !
 end-group
```

3 Check the inheritance view and verify that the apply-group in the lowest configuration submode gets the highest priority:

```
RP/0/RP0/CPU0:router# show running inheritance router ospf 1

router ospf 1
 nsr
 router-id 121.121.121.121
 nsf cisco
 redistribute connected
 address-family ipv4 unicast
 area 0
  interface GigabitEthernet0/0/0/0
   ## Inherited from group go-a
   cost 200                        << apply-group in lowest submode  gets highest priority
   !
  interface GigabitEthernet0/0/0/1
   ## Inherited from group go-b
   cost 250
   !
  interface GigabitEthernet0/0/0/3
   ## Inherited from group go-b
   cost 250
   !
  interface GigabitEthernet0/0/0/4
   ## Inherited from group go-b
   cost 250
   !
  interface GigabitEthernet0/0/0/21
   bfd minimum-interval 100
   bfd fast-detect
   bfd multiplier 3
   ## Inherited from group go-b
   cost 250
   !
  interface TenGigE0/2/0/0.3891
   !
  interface TenGigE0/2/0/0.3892
   !
  interface TenGigE0/2/0/0.3893
   !
  interface TenGigE0/2/0/0.3894
   !
  !
 !
```

4 Check the functionality of the cost inheritance through the groups:

```
RP/0/RP0/CPU0:router# show ospf 1 interface GigabitEthernet 0/0/0/0

GigabitEthernet0/0/0/0 is up, line protocol is up
  Internet Address 1.0.1.1/30, Area 0
  Process ID 1, Router ID 121.121.121.121, Network Type BROADCAST, Cost: 200
  Transmit Delay is 1 sec, State DR, Priority 1, MTU 1500, MaxPktSz 1500
  Designated Router (ID) 121.121.121.121, Interface address 1.0.1.1
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  Non-Stop Forwarding (NSF) enabled
    Hello due in 00:00:02
  Index 5/5, flood queue length 0
  Next 0(0)/0(0)
  Last flood scan length is 1, maximum is 40
  Last flood scan time is 0 msec, maximum is 7 msec
  LS Ack List: current length 0, high water mark 0
  Neighbor Count is 1, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
```

```
        Multi-area interface Count is 0
```

Link Bundling Usage: Example

This example shows how to configure interface membership in a bundle link:

1 Configure the configuration groups:

```
RP/0/RP0/CPU0:router# show running group bundle1

group bundle1
 interface 'GigabitEthernet0/1/0/1[1-6]'
  bundle id 1 mode active
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group isis l2tr  isis2  mpp  bundle1
```

2 Check the local configuration:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/1/0/11

interface GigabitEthernet0/1/0/11
!

RP/0/RP0/CPU0:router# show running interface Bundle-Ether1

interface Bundle-Ether1
 ipv4 address 108.108.1.1 255.255.255.0
 bundle maximum-active links 10
 bundle minimum-active links 5
!
```

3 Check the inheritance configuration view:

```
RP/0/RP0/CPU0:router# show running inheritance interface GigabitEthernet 0/1/0/11

interface GigabitEthernet0/1/0/11
 ## Inherited from group bundle1
 bundle id 1 mode active
!
```

4 Check that the inheritance configuration took effect:

```
RP/0/RP0/CPU0:router# show interface Bundle-Ether1

Bundle-Ether1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is Aggregated Ethernet interface(s), address is 0024.f71f.4bc3
  Internet address is 108.108.1.1/24
  MTU 1514 bytes, BW 6000000 Kbit (Max: 6000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 6000Mb/s
  loopback not set,
  ARP type ARPA, ARP timeout 04:00:00
    No. of members in this bundle: 6
      GigabitEthernet0/1/0/11    Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/12    Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/13    Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/14    Full-duplex  1000Mb/s     Active
```

```
        GigabitEthernet0/1/0/15      Full-duplex  1000Mb/s     Active
        GigabitEthernet0/1/0/16      Full-duplex  1000Mb/s     Active
Last input 00:00:00, output 00:00:00
Last clearing of "show interface" counters never
5 minute input rate 8000 bits/sec, 1 packets/sec
5 minute output rate 3000 bits/sec, 1 packets/sec
    2058 packets input, 1999803 bytes, 426 total input drops
    0 drops for unrecognized upper-level protocol
    Received 1 broadcast packets, 2057 multicast packets
            0 runts, 0 giants, 0 throttles, 0 parity
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
    1204 packets output, 717972 bytes, 0 total output drops
    Output 2 broadcast packets, 1202 multicast packets
    0 output errors, 0 underruns, 0 applique, 0 resets
    0 output buffer failures, 0 output buffers swapped out
    0 carrier transitions
```