# Configuration Groups

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.

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


- Configuration Groups Overview, on page 1
- Guidelines and Restrictions for Configurations Group, on page 1
- Create a Configuration Group, on page 3
- Verify Configuration Groups Creation, on page 5
- Priority Inheritance for Configuration Groups, on page 6
- Regular Expressions in Configuration Groups , on page 10
- Use Cases of Configuration Groups, on page 20
- Replacing Configuration Elements, on page 31


# Configuration Groups Overview

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.

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.

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 configuration group.

# Guidelines and Restrictions for Configurations Group

These are the restrictions for configuration groups:

- Configuration groups aren't supported in administration configurations. And corresponding apply-groups aren't supported in administration configurations.
- Use of preconfigured interfaces in configuration groups isn't supported.
- Downgrading from an image that supports configuration groups to an image that doesn't support them isn't supported.
- Access lists (ACLs), Quality of Service (QoS) and route policy configurations don't support the use of configuration groups. Configurations such as these aren't 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
```

- Configuration groups are not available through the XML interface.
- Commands that require a node identifier for the **location** keyword aren't supported. For example, this configuration isn't supported:

  ```
  lpts pifib hardware police location 0/0/CPU0
  ```

- Overlapping regular expressions within a configuration group for the same configuration aren't 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 isn't permitted because it can't be determined whether the **interface gigabitethernet0/4/1/0** configuration inherits **mtu 1500** or **mtu 2000**. Both expressions in the configuration group match gigabit ethernet 0/4/1/0.

- Up to eight configuration groups are permitted on one **apply-group** command.
- Use a multiline configuration style to configure configuration groups (like **group** or **apply-group** commands) by entering each configuration mode in a separate line, one configuration per line. Using a multiline configuration style helps configuration properties to be fully inherited and provides better readability during troubleshooting.

An example for a correct configuration style is:

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

# Create a Configuration Group

The steps of creating a configuration group involves two sub-tasks:

- Creating a configuration group
- Applying a configuration group

Use the following steps to create a configuration group:

### Procedure

**Step 1** Enter global configuration mode.

**Example:**

```
Router# configure
```

**Step 2** Specify a name for a configuration group and enter group configuration mode to define the group.The *group-name* argument can have up to 32 characters and cannot contain any special characters.

**Example:**

```
Router(config)# group g-interf
```

**Step 3** Specify 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 10. This example is applicable to all Gigabit Ethernet interfaces.

**Example:**

```
Router(config-GRP)# interface 'GigabitEthernet.*'
Router(config-GRP-if)# mtu 1500
```

**Step 4** Complete the configuration of a configuration group and exits to global configuration mode.

**Example:**

```
Router(config-GRP-if)# end-group
```

**Step 5** Add 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 mode, the MTU value is inherited by all Gigabit Ethernet interfaces that don't have an MTU value configured.

**Example:**

```
Router(config)# interface GigabitEthernet0/2/0/0
Router(config-if)# apply-group g-interf
```

# Simple Configuration Group:Example

**Procedure**

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

**Example:**

```
Router(config)# group g-logging
Router(config-GRP)# logging trap notifications
Router(config-GRP)# logging console debugging
Router(config-GRP)# logging monitor debugging
Router(config-GRP)# logging buffered 10000000 R
Router(config-GRP)# end-group
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 locations, and their effect depends on the context within which they are applied.

**Procedure**

**Step 1** Consider this configuration group:

**Example:**

```
Router(config)# group g-interfaces
Ruter(config-GRP)# interface 'FastEthernet.*'
Router(config-GRP-if)# mtu 1500
Router(config-GRP-if)# exit
Router(config-GRP)# interface 'GigabitEthernet.*'
Router(config-GRP-if)# mtu 1000
Router(config-GRP-if)# exit
Router(config-GRP)# interface 'POS.*'
Router(config-GRP-if)# mtu 2000
Router(config-GRP-if)# end-group
```

**Step 2** 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:

**Example:**

```
Router(config)# interface GigabitEthernet0/2/0/0
Router(config-if)# apply-group g-interfaces
Router(config-if)# ipv4 address 192.0.2.1
```

**Step 3** In this example, the Fast Ethernet interface is configured to have an MTU of 1500:

**Example:**

```
Router(config)# interface FastEthernet0/2/0/0
Router(config-if)# apply-group g-interfaces
Router(config-if)# ipv4 address 192.0.2.1
```

**Step 4** The same configuration group is used in both cases, but only the applicable configuration statements are used.

# Verify Configuration Groups Creation

To verify the router configuration using configuration groups:

**Procedure**

**Step 1** Use the **show running-config group** command to display the contents of a specific or all configured configuration groups.

**Example:**

```
Router# show running-config group
group g-int-ge
 interface 'GigabitEthernet.*'
  mtu 1000
   negotiation auto
 !
```

**Step 2** Use the **show running-confi**g command to display 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.

**Example:**

```
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
 !
```

**Step 3** You can use the **show running-config inheritance** command to check the inherited configuration whereever a configuration group has been applied.

**Example:**

```
Router# show running-config inheritance
.
.
group G-INTERFACE-MTU
 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
!
.
.
```

**Step 4** Use the **show running-config interfcae inheritance** command to verify inherited configuration for a specific configuration command.

**Example:**

```
Router# show running-config interface pos0/4/1/0 inheritance [detail]


interface POS0/4/1/0
 ## Inherited from group G-INTERFACE-MTU
  mtu 1500
```

# Priority Inheritance for Configuration Groups

The configuration groups inheritance is supported according to the priority.

Apply groups priority inheritance helps configuration groups handle common configuration statements between groups. When multiple configuration groups have common configuration statements, the inheritance priority is that configuration statements present in inner groups have precedence over configuration statements present in outer groups. The tiebreaker is determined by the system order (lexicographical) of the regular expressions. User-defined order of commands isn't 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's not contained in any other group. Within a configuration group, inheritance priority is the 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
         !
        !
       !
```

This 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` are applied.

Case 2

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

**Note** From the Cisco IOS XR Software Release 6.3.1 onwards, you're able to enter the configuration group definition, **apply-group** and **exclude-group** command in any order as long as the entire commit has all the group definitions needed.

**Exclude-group** command can be used to exclude a configuration group (or groups) to be inherited by the router configuration, in the appropriate configuration mode.

# Configuration Group Precedence

**Procedure**

**Step 1** 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.

**Example:**

```
 Router(config)# group g-ospf2
Router(config-GRP)# router ospf '.*'
Router(config-GRP-ospf)# area '.*'
Router(config-GRP-ospf-ar)# cost 2
Router(config-GRP-ospf-ar)# end-group

Router(config)# group g-ospf100
Router(config-GRP)# router ospf '.*'
Router(config-GRP-ospf)# area '.*'
Router(config-GRP-ospf-ar)# cost 100
Router(config-GRP-ospf-ar)# end-group
```

**Step 2** 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.

**Example:**

```
Router(config)# router ospf 0
Router(config-ospf)# apply-group g-ospf100
Router(config-ospf)# area 0
Router(config-ospf-ar)# apply-group g-ospf2
```

# Precedence for Local Configuration

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.

**Procedure**

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

**Example:**

```
Router# show running interface gigabitEthernet 0/0/0/39

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

Router# show running interface gigabitEthernet 0/0/0/38
 interface GigabitEthernet0/0/0/38
!

Router# show running ipv4 access-list smany
ipv4 access-list smany
 10 permit ipv4 any any
!

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
!
```

**Step 2** Configure and apply the access list group configuration:

**Example:**

```
Router# show running group acref
group acref
interface 'GigabitEthernet0/0/0/3.*'
 ipv4 access-group adem ingress
 ipv4 access-group adem egress
!
end-group


Router# show running | inc apply-group

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

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

**Example:**

```
Router# show running interface gigabitEthernet 0/0/0/39
 interface GigabitEthernet0/0/0/39 ipv4 access-group smany ingress ipv4 access-group smany egress
!

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
!

Router# show running interface gigabitEthernet 0/0/0/38
interface GigabitEthernet0/0/0/38
!

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
!
```

**Step 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).

# Automatic Inheritance of Modifications to Configuration Group

**Procedure**

**Step 1** 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:

**Example:**

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

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

**Step 2** Now you change the configuration group as in this example:

**Example:**

```
Router(config)# group g-interface-mtu
Router(config-GRP)# interface 'POS.*'
Router(config-GRP-if)# mtu 2000
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.

# 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.

## Restrictions for Regular Expressions in Configuration Groups

The following are the restrictions for regular expressions:

- Some regular expressions aren't supported within groups. For example, '?', '|' and '$,' aren't supported within groups. Also some characters such as /d and /w aren't supported.
- The choice operator "|" to express multiple match expressions within a regular expression isn't supported. For example, these expressions aren't supported:

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

  `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.*
- Not all POSIX regular expressions are supported.
- If the system can't determine from the regular expression what the configuration should be, the expression is not considered valid.
- The regular expression '.*' isn't 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.

# Regular Expressions for Interface Identifiers

Configuration groups don't 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 '.*' isn't 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.*'.

### Procedure

**Step 1** Display a list of available interface types for your router configuration, use the **interface?** at the configuration group prompt:

**Example:**

```
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
```

**Step 2** 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 this example:

**Example:**

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

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

# Regular Expressions for an OSPF Configuration

**Procedure**

**Step 1** Exact router process names and OSPF areas can't 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:

**Example:**

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

**Step 2** Use the expression '\.', to specify that the OSPF area must be an IP address.

**Example:**

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

**Step 3** Use the expression '1.*', to specify that the OSPF area must be a scalar value.

**Example:**

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

# Regular Expressions for ISIS Configuration

**Procedure**

**Step 1** 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:

**Example:**

```
RP/0/RSP0/CPU0:router(config)# group g-isis-gige
RP/0/RSP0/CPU0:router(config-GRP)# router isis '.*'
RP/0/RSP0/CPU0:router(config-GRP-isis)# interface 'GigabitEthernet.*'
RP/0/RSP0/CPU0:router(config-GRP-isis-if)# lsp-interval 20
RP/0/RSP0/CPU0:router(config-GRP-isis-if)# hello-interval 40
RP/0/RSP0/CPU0:router(config-GRP-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-GRP-isis-if-af)# metric 10
RP/0/RSP0/CPU0:router(config-GRP-isis-if-af)# end-group
RP/0/RSP0/CPU0:router(config)#
```

**Step 2** To illustrate the use of this configuration group, assume that you want to configure these Gigabit Ethernet interfaces with the ISIS routing parameters:

**Example:**

```
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
  !
 !
!
```

**Step 3** 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:

**Example:**

```
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
 !

!
```

**Step 4** 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:

**Example:**

```
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
 !
!
```

**Step 5** 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:

**Example:**

```
 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.

# Regular Expressions for a BGP AS

Exact BGP AS values can’t 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.

### Procedure

Use two regular expressions separated by a dot, to match AS dot format instances as shown in this example:

**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 Regular Expressions to a Uniform Type

**Procedure**

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

**Example:**

```
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 isn't applicable to both these interface types, they don't resolve to a uniform type and therefore the system doesn't allow this configuration.

# Overlapping Regular Expressions

Regular expressions are used in the names of configuration statements within a configuration group. This permits inheritance by the configuration when applied to matching names. Single quotes are used to delimit the regular expression. Overlapping regular expression within a configuration group for the same configuration is permitted.

**Procedure**

**Step 1** The example, given below, illustrates the process of creating and applying multiple configuration groups:

**Example:**

```
Router(config)#group FB_flexi_snmp
Router(config-GRP)# snmp-server vrf '.*'
```

```
Router(config-GRP-snmp-vrf)# host 10.0.0.1 traps version 2c group_1
Router(config-GRP-snmp-vrf)# host 10.0.0.1 informs version 2c group_1
Router(config-GRP-snmp-vrf)# context group_1
Router(config-GRP-snmp-vrf)#commit

Router(config-GRP-snmp-vrf)#root
Router(config)#snmp-server vrf vrf1
Router(config-snmp-vrf)#snmp-server vrf vrf10
Router(config-snmp-vrf)#!
Router(config-snmp-vrf)#snmp-server vrf vrf100
Router(config-snmp-vrf)#commit

Router(config-snmp-vrf)#root
Router(config)#apply-group FB_flexi_snmp
Router(config)#do sh running-config group
group FB_flexi_snmp
 snmp-server vrf '.*'
  host 10.0.0.1 traps version 2c group_1
  host 10.0.0.1 informs version 2c group_1
  context group_1
 !
end-group

apply-group FB_flexi_snmp
snmp-server vrf vrf1
!
snmp-server vrf vrf10
!
snmp-server vrf vrf100
!

Router#show running-config inheritance detail

group FB_flexi_snmp
 snmp-server vrf
 '.*'
  host 10.0.0.1 traps version 2c group_1
  host 10.0.0.1 informs version 2c group_1
  context group_1
  !
end-group

snmp-server vrf vrf1

## Inherited from group FB_flexi_snmp host 10.0.0.1 traps version 2c group_1 ## Inherited from group
 FB_flexi_snmp host 10.0.0.1 informs version 2c group_1 ## Inherited from group FB_flexi_snmp context
 group_1

!

snmp-server vrf vrf10

## Inherited from group FB_flexi_snmp host
10.0.0.1 traps version 2c group_1
## Inherited from group FB_flexi_snmp host
10.0.0.1 informs version 2c group_1
## Inherited from group FB_flexi_snmp
context group_1
!

snmp-server vrf vrf100
## Inherited from group FB_flexi_snmp
host 10.0.0.1 traps version 2c group_1
## Inherited from group FB_flexi_snmp
```

```
host 10.0.0.1 informs version 2c group_1 ##
Inherited from group FB_flexi_snmp context group_1
```

**Step 2** The example given below demonstrates the regular expression. In this example `snmp-server vrf '.*'` and `snmp-server vrf '[\w]+` are two different regular expressions.

**Example:**

```
group FB_flexi_snmp
snmp-server vrf
'.*'
host 10.0.0.1 traps version 2c group_1
host 10.0.0.1 informs version 2c group_1
context group_1
!

snmp-server vrf '[\w]+'
host 172.16.0.1 traps version 2c group_2
host 172.16.0.1 informs version 2c group_2
context group_2
!
end-group
```

**Step 3** This individual regular expression gets combined to all the three expressions - `snmp-server vrf vrf1`, `snmp-server vrf vrf10` and `snmp-server vrf vrf100` as given below.

**Example:**

```
apply-group FB_flexi_snmp
snmp-server vrf vrf1
!
snmp-server vrf vrf10
!
snmp-server vrf vrf100
!
```

**Step 4** In a configuration group, there can be instances of regular expressions overlap. In such cases, the regular expression with the highest priority is activated and inherited, when applied. It has that regular expression, which comes first in the lexicographic order that has the highest priority. The following example shows how to use overlapping regular expressions and how the expression with higher priority is applied:

**Example:**

```
group FB_flexi_snmp
snmp-server vrf
'.*'
 host 10.0.0.1 traps version 2c group_1
 host 10.0.0.1 informs version 2c group_1
 context group_1
!

snmp-server vrf '[\w]+'

 host 172.16.0.1 traps version 2c group_2
 host 172.16.0.1 informs version 2c group_2
 context group_2
!
end-group
```

**Step 5** The expression shown below has the highest priority:

**Example:**

```
group FB_flexi_snmp snmp-server vrf '.*'
```

```
host 10.0.0.1 traps version 2c group_1
host 10.0.0.1 informs version 2c group_1
context group_1
```

**Step 6** The examples given above, show two different regular expression `snmp-server vrf '.*'` and `snmp-server vrf '[\w]+'`. The expression below, shows how these two expressions get merged together:

**Example:**

```
apply-group FB_flexi_snmp
snmp-server vrf vrf1
!
snmp-server vrf vrf10
!
snmp-server vrf vrf100
!
```

**Step 7** Any change in a regular expression with lower priority will not affect the inheritance.Any changes made to an existing regular expression, which is of less (non-top) priority, it will not have any effect on the inheritance.

**Example:**

```
snmp-server vrf '[\w]+'
host 172.16.0.1 traps version 2c group_2
host 172.16.0.1 informs version 2c group_2
context group_2
```

**Step 8** The expression with the higher priority gets inherited, as shown below:

**Example:**

```
group FB_flexi_snmp
snmp-server vrf
'.*'
host 10.0.0.1 traps version 2c group_1
host 10.0.0.1 informs version 2c group_1
context group_1
```

# Configuration Group Inheritance with Regular Expressions

The following guidelines apply to configuration group inheritance with regular expressions:

- 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.

  ```
  Router(config)# group g-ospf
  Router(config-GRP)# router ospf '.*'
  Router(config-GRP-ospf)# nsf cisco
  Router(config-GRP-ospf)# packet-size 3000
  Rrouter(config-GRP-ospf)# end-group
  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.

```
Router(config)# group g-ospf
Router(config-GRP)# router ospf '.*'
Router(config-GRP-ospf)# area '.*'
Router(config-GRP-ospf-ar)# packet-size 2000
Router(config-GRP-ospf)# end-group
Router(config)# apply-group g-ospf
Router(config)# router ospf 200
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

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

```
Router(config)# group g-l2trans-if
Router(config-GRP)# interface 'TenGigE.*\..*' l2transport
Router(config-GRP)# mtu 1514
Router(config-GRP)# end-group
Router(config)# interface TenGigE0/0/0/0.1 l2transport
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
!
```

# Use Cases of Configuration Groups

## Basic Configuration

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.

### Procedure

**Step 1** Configure the configuration group that configures MAC accounting:

**Example:**

```
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
```

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

**Example:**

```
Router# show running | in apply-group gd21
Building configuration...
apply-group gd21
```

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

**Example:**

```
Router# show running interface
interface GigabiEthernet0/0/0/21
!
interface GigabitEthernet0/0/0/22
!
```

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

**Example:**

```
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
!

!
```

**Step 5** Verify that the inherited configuration actually takes effect:

**Example:**

```
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

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

**Procedure**

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

**Example:**

```
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
```

```
Router# show running | inc apply-group
Building configuration...
apply-group l2tr
```

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

**Example:**

```
 Router# show running interface gigabitEthernet0/0/0/30
interface GigabitEthernet0/0/0/30
!
## Inherited from group l2tr mtu 1500

!

Router# show running interface gigabitEthernet0/0/0/9.800
interface GigabitEthernet0/0/0/9.800 encapsulation dot1q 800
!

Router# show running inheritance interface gigabitEthernet0/0/0/9.800
interface GigabitEthernet0/0/0/9.800 ## Inherited from group l2tr
mtu 1400 encapsulation dot1q800
!

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
!
```

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

**Example:**

```
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

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

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

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

### Procedure

**Step 1** Configure the configuration group and apply-group:

**Example:**

```
Router# show running group acref

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

```
Router# show running | inc apply-group
Building configuration...
apply-group isis l2tr isis2 mpp bundle1 acref
```

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

**Example:**

```
Router# show running interface gigabitEthernet 0/0/0/30
interface GigabitEthernet0/0/0/30
!
```

```
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
!
```

```
Router# show running interface gigabitEthernet 0/0/0/31
interface GigabitEthernet0/0/0/31
!
```

```
 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
```

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

# ISIS Hierarchical Configuration

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

### Procedure

**Step 1** Configure the local ISIS configuration:

**Example:**

```
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
  !
 !
!
```

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

**Example:**

```
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
```

```
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
```

```
Router# show running | inc apply-group
Building configuration...
apply-group isis l2tr isis2 mpp bundle1 acref
```

**Step 3** Check the inheritance view of the ISIS configuration:

**Example:**

```
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
  !
 !
!
```

**Step 4** Verify the actual functionality.

**Example:**

```
Router# show isis interface TenGigE0/2/0/0.3528 | inc Metric
Metric (L1/L2): 50/50
```

# OSPF Hierarchy

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

**Procedure**

**Step 1** Configure a local OSPF configuration:

**Example:**

```
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
!
```

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

**Example:**

```
Router# show running group go-a
group go-a router ospf '.*'
area '.*' interface 'Gig.*'
cost 200
  !
 !
!
end-group
```

```
Router# show running group go-b
group go-b router ospf '.*'
area '.*' interface 'Gig.*'
cost 250
  !
 !
!
end-group
```

```
Router# show running group go-c
group go-c router ospf '.*'
area '.*' interface 'Gig.*'
cost 300
  !
 !
!
end-group
```

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

**Example:**

```
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
  !
 !
!
```

**Step 4** Check the functionality of the cost inheritance through the group.

**Example:**

```
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

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

**Procedure**

**Step 1** Configure the configuration groups:

**Example:**

```
Router# show running group bundle1
group bundle1
interface 'GigabitEthernet0/1/0/1[1-6]'
 bundle id 1 mode active
!
end-group
```

```
Router# show running | inc apply-group
Building configuration...
apply-group isis l2tr isis2 mpp bundle1
```

**Step 2** Check the local configuration:

**Example:**

```
RP/0/RSP0/CPU0:router# show running interface gigabitEthernet 0/1/0/11
interface GigabitEthernet0/1/0/11
 !
```

```
Router# show running interface Bundle-Ether1
interface Bundle-Ether1
ipv4 address 198.51.100.1
bundle maximum-active links 10
bunle minimum-active links 5
!
```

**Step 3** Check the inheritance configuration view:

**Example:**

```
Router# show running inheritance interface GigabitEthernet 0/1/0/11
 interface GigabitEthernet0/1/0/11 ## Inherited from group bundle1 bundle id 1 mode active
!
```

**Step 4** Check that the inheritance configuration took effect:

**Example:**

```
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
```

# Replacing Configuration Elements

You can replace interface and IP address configurations, or any pattern in an existing configuration, using the **replace** {**interface**} or **replace** {**pattern**}commands in Global Configuration mode. These commands can be executed not only on individual interfaces or IP addresses, but also on regular expressions to replace a range of interfaces or addresses.

Use these commands to simplify configuration changes where you would normally need to copy the configuration and edit it manually. For example, when you're moving a physical connection from one interface to another, you can use the **replace interface** command to update your configuration to use the new interface address.

**Note** These commands replace every occurrence of the specified interfaces or patterns in the running configuration.

We recommend that you use this command after disconnecting the old interface and before connecting to the new interface.

Similarly, if your IP addressing scheme has changed (for example, a BGP neighbor address), use the **replace pattern** command to update your configuration to use the new IP address.

# Configuration Example for Replacing an Interface Configuration

### Procedure

The example in this section uses the following interface configurations:

**Example:**

```
Router# show configuration running-config

interface MgmtEth0/RSP0/CPU0/0 shutdown
!
interface HundredGigE0/0/0/0
 description first
 ipv4 address 10.20.30.40 255.255.0.0
 shutdown
!
interface HundredGigE0/0/0/1
 shutdown
!
interface HundredGigE0/0/0/2description 10.20.30.40
 shutdown
!
interface HundredGigE0/0/0/3
 description 1020304050607080
 shutdown
!
interface HundredGigE0/0/0/4
```

# Configuration Example for Replacing IP Addresses in a Configuration

### Procedure

**Step 1** The example in this section uses the following configuration:

**Example:**

```
Router# show configuration running-config

. . .

ipv4 access-list mylist

10 permit tcp 209.165.200.224/27 host 209.165.200.225

20 deny ipv4 any 209.165.201.0/27

!

interface MgmtEth0/RSP0/CPU0/0 shutdown

!

interface HundredGigE0/1/0/1 description first

ipv4 address 209.165.200.224/27 255.255.0.0 shutdown

!

interface HundredGigE0/0/0/2 description 209.165.200.224/27 shutdown

!

route-policy temp

if ospf-area is 209.165.200.224/27 or source in (2.3.4.5/20) then pass

endif end-policy

!
```

**Step 2** This example shows how to replace IP address 209.165.200.224/27 with 209.165.201.0/27 using the **replace pattern** *'pattern'* **with** *'pattern'* command:

**Example:**

```
Router(config)# replace pattern '209.165.200.224/27' with '209.165.201.0/27'

Loading.

443 bytes parsed in 1 sec (442)bytes/sec
```

**Note**
Use single quotes around the pattern.

**Step 3** Enter the **show configuration** command to display and verify the configuration changes. Then commit the changes.

In the example below, you can see that every occurrence of IP address 209.165.200.224/27 is replaced with 209.165.201.0/27.

**Example:**

```
Router(config)# show configuration

Building configuration...

!! IOS XR Configuration 0.0.0 ipv4 access-list mylist

no 10

10 permit tcp 209.165.200.224/27 host 1.2.4.5

!

interface HundredGigE0/0/0/2 no description

description 209.165.201.0/27

!

interface HundredGigE0/1/0/1

no ipv4 address 209.165.200.224/27 255.255.0.0

ipv4 address 209.165.201.0/27 255.255.0.0

!

!

route-policy temp

if ospf-area is 209.165.201.0/27 or source in (2.3.4.5/20) then pass

endif end-policy

!

end
Router(config)# commit
```

In the example above, you can see that every occurrence of IP address 209.165.200.224/27 has been replaced with 209.165.201.0/27.

# Configuration Example for Replacing Patterns Using Regular Expressions

You can replace a range of interfaces or addresses using POSIX-compliant regular expressions.

**Note** For information about using regular expressions, refer to the "Understanding Regular Expressions, SpecialCharacters, and Patterns" chapter in the *Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*.

### Procedure

**Step 1** The example in this section uses the following configuration:

**Example:**

```
Router# show configuration running-config
. . .
interface HundredGigE0/2/0/0
ipv4 address 209.165.201.0/27 209.165.200.224/27
!
interface HundredGigE0/2/0/1
ipv4 address 209.165.202.128/27 209.165.200.224/27
!
interface HundredGigE0/2/0/2
ipv4 address 12.0.0.12 209.165.200.224/27
!
interface HundredGigE0/2/0/3
ipv4 address 13.0.0.13 209.165.200.224/27
!
interface HundredGigE0/2/0/4
ipv4 address 14.0.0.14 209.165.200.224/27
!
interface HundredGigE0/3/0/0 shutdown
!
interface HundredGigE0/3/0/1 shutdown
!
interface HundredGigE0/3/0/2 shutdown
!
interface HundredGigE0/3/0/3 shutdown
!
interface HundredGigE0/3/0/4 shutdown
!
interface HundredGigE0/3/0/5 shutdown
```

```
!

interface HundredGigE0/3/0/6 shutdown

!

end
```

**Step 2** This example shows how to replace interfaces HundredGigE0/2/0/0 through HundredGigE0/2/0/4 with interfaces HundredGigE0/3/0/0 through HundredGigE0/3/0/4 using regular expressions:

**Example:**

```
Router(config)# replace pattern 'HundredGigE0/2/0/([0-4]*)' with 'HundredGigE0/3/0/1'

Loading.

619 bytes parsed in 1 sec (617)bytes/sec
```

**Step 3** Enter the **show configuration** command to display and verify the configuration changes. Then commit the changes.

In this example, you can see that the HundredGigE0/2/0/*x* interfaces are removed from the configuration (no interface HundredGigE0/2/0/*x*) and is replaced with HundredGigE0/3/0/*x*.

**Example:**

```
Router(config)# show configuration

Building configuration...

!! IOS XR Configuration 0.0.0 no interface HundredGigE0/2/0/0

 no interface HundredGigE0/2/0/1 no interface HundredGigE0/2/0/2 no interface HundredGigE0/2/0/3 no
 interface HundredGigE0/2/0/4 interface HundredGigE0/3/0/0

ipv4 address 209.165.201.0/27 209.165.200.224/27

!

interface HundredGigE0/3/0/1

ipv4 address 09.165.202.128/27 209.165.200.224/27

!

interface HundredGigE0/3/0/2

ipv4 address 12.0.0.12 209.165.200.224/27

!

interface HundredGigE0/3/0/3

ipv4 address 13.0.0.13 209.165.200.224/27

!

interface HundredGigE0/3/0/4

ipv4 address 14.0.0.14 209.165.200.224/27

!

End
```

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