# Best Practices for Route-Map Configuration in Cisco IOS XE: Action and Sequence Number Importance

## Contents

## Introduction

This document describes the correct method to configure and display a route-map within configuration mode, particularly emphasizing the importance of specifying the action (permit or deny) and sequence number.

## Background Information

It addresses an observed behavior where a route-map entry originally configured with a deny action appears to have changed to permit unexpectedly.

## Problem

In Cisco IOS® XE, if a route-map is entered without explicitly specifying an action (permit/deny) and sequence number, the CLI defaults the action to permit. This can inadvertently alter the behavior of the route-map if administrators are not careful with syntax.

## Observed Behavior

When this configuration is applied:

```
router(config)#route-map TEST1 deny 1
router(config-route-map)#match ip address prefix-list PREFIXES
router(config-route-map)#exit
```

```
router(config)#route-map TEST1 permit 10
router(config-route-map)#match ip address prefix-list FIX
router(config-route-map)#exit
```

The expected output is:

```
router(config)#do sh run | sec TEST1
route-map TEST1 deny 1
 match ip address prefix-list PREFIXES
route-map TEST1 permit 10
 match ip address prefix-list FIX
```

However, upon re-entering route-map TEST1 without specifying sequence or action as shown:

```
router(config)#route-map TEST1
router(config-route-map)#do sh run | sec TEST1
```

The CLI output unexpectedly shows:

```
route-map TEST1 permit 1
 match ip address prefix-list PREFIXES
route-map TEST1 permit 10
 match ip address prefix-list FIX
```

This indicates the originally configured deny 1 entry has been modified to permit 1.

# Expected CLI Behavior

• When you enter the command **router(config)# route-map TEST1** without specifying an action (permit or deny) and a sequence number, the parser assumes you meant 'permit' by default.

• As soon as you exit that sub-mode, Cisco IOS XE rewrites the first sequence and changes the action from deny to permit because no explicit action was provided. Simply entering route-map <name> without a sequence number enters sequence 10 as permit by default. This behavior can sometimes override prior entries if not specified carefully.

In other words, the change is not triggered by an unseen process — it is simply the CLIs default handling of an incomplete route-map command.

In order to preserve the deny in the first statement, always include both the action and the sequence number, for example:

**router(config)# route-map TEST1 deny 1**.

# Why This Happens

This is not a bug but an intended behavior of the Cisco IOS XE CLI parser. It adheres to these rules:

- If a route-map is called without a sequence number, **sequence 10** is assumed.
- If no action is specified, **permit** is the default.
- If re-entering an existing route-map without details, IOS can **reassign default actions**, unintentionally modifying prior entries.

# Solution

In order to preserve the intended behavior of your route-map configuration, always include both:

- the **action** (permit or deny)
- and the **sequence number**

<#root>

```
router(config)#route-map TEST1 deny 1 >>>>>>>>>>>>>>>>
```

**This ensures IOS-XE does not override the entry with default values.**

```
router(config-route-map)#do sh run | sec route-map TES
route-map TEST1 deny 1
 match ip address prefix-list ALL-PREFIXES
route-map TEST1 permit 10
 match ip address prefix-list ALL-VPN-FIX
```

# Recommended Method To Implement

The configuration outputs for reference are mentioned here. The command behavior reflects the importance of properly defined sequence numbers and how the order of insertion of permit/deny does not affect ascending arrangement of sequence number in the running configuration.

### Example 1. Incorrect CLI Use Leading to Default Permit

<#root>

```
ASR1002-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
ASR1002-1(config)#route-map TEST deny 1
ASR1002-1(config-route-map)#$dress prefix-list ALL-PREFIXES
ASR1002-1(config-route-map)#exit
ASR1002-1(config)#route-map TEST permit 10
ASR1002-1(config-route-map)#$dress prefix-list ALL-VPN-FIX
ASR1002-1(config-route-map)#exit
ASR1002-1(config)#do sh run | sec TEST
route-map TEST deny 1
 match ip address prefix-list ALL-PREFIXES
route-map TEST permit 10
 match ip address prefix-list ALL-VPN-FIX
ASR1002-1(config)#exit
```

```
ASR1002-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
ASR1002-1(config)#route-map TEST

ASR1002-1(config-route-map)#do sh run | sec route-map TES


route-map TEST permit 1


 match ip address prefix-list ALL-PREFIXES


route-map TEST permit 10


 match ip address prefix-list ALL-VPN-FIX


ASR1002-1(config-route-map)#exit
```

Note how deny 1 becomes permit 1. Instead, include the action along with the sequence number in order to maintain the correct configuration of route-map:

```
<#root>

ASR1002-1(config)#route-map TEST deny 1


ASR1002-1(config-route-map)#do sh run | sec route-map TES


route-map TEST deny 1


 match ip address prefix-list ALL-PREFIXES


route-map TEST permit 10


 match ip address prefix-list ALL-VPN-FIX
```

## Example 2. Sequence Order Maintained Regardless of Entry Order

```
ASR1001-1(config)#route-map test deny 50
ASR1001-1(config-route-map)#^Z
ASR1001-1#show running-config | sec route-map
route-map test deny 50
ASR1001-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
ASR1001-1(config)#route-map test permit 40
ASR1001-1(config-route-map)#^Z
ASR1001-1#show running-config | sec route-map
route-map test permit 40
route-map test deny 50
ASR1001-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
ASR1001-1#show running-config | sec route-map
route-map test permit 40
route-map test deny 50
```

Although the deny statement was entered first, the CLI sorts by ascending sequence number, not by configuration order.

# Conclusion

When configuring route-maps in Cisco IOS-XE, always define:

- the **sequence number**
- and the **action**

Failure to do so can result in unexpected behavior due to CLI defaults. Understanding these nuances is essential for maintaining accurate routing policy configurations.