Configuring Object Tracking

This module describes the configuration of object tracking on your Cisco IOS XR network. For complete descriptions of the commands listed in this module, see Related Documents, on page 17. To locate documentation for other commands that might appear in the course of performing a configuration task, search online in *Cisco ASR 9000 Series Aggregation Services Router Commands Master List*.

Table 1: Feature History Table

Feature Name	Release Information	Description
Enhanced Object Tracking for Loopback Interfaces	Release 7.10.1	You can now track an object (for example, an interface) and configure to error-disable a loopback interface when the tracked object state changes. This is beneficial when the loopback interface is the source of multiple tunnels and all tunnels need to be brought down simultaneously. Earlier, you couldn't error-disable loopback interface.
Enhanced Object Tracking	Release 6.4.2	The Enhanced Object Tracking feature is introduced. The ability to error-disable interfaces is added based on the state of objects that are tracked.
Enhanced Object Tracking	Release 4.2.1	The ability to create a tracked list based on a threshold percentage or weight was added.
Enhanced Object Tracking	Release 4.0.0	This feature was introduced.

This module contains the following topics:

Prerequisites for Implementing Object Tracking

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Object Tracking

Object tracking is a mechanism for tracking an object to take any client action on another object as configured by the client. The object on which the client action is performed may not have any relationship to the tracked objects. The client actions are performed based on changes to the properties of the object being tracked.

You can identify each tracked object by a unique name that is specified by the track command in the configuration mode.

The tracking process periodically polls the tracked object and reports any changes to its state. The state of the tracked objects can be up or down. The polling occurs either immediately or after a delay of a configured period.

You can also track multiple objects by a list. You can use a flexible method for combining objects with Boolean logic. This functionality includes:

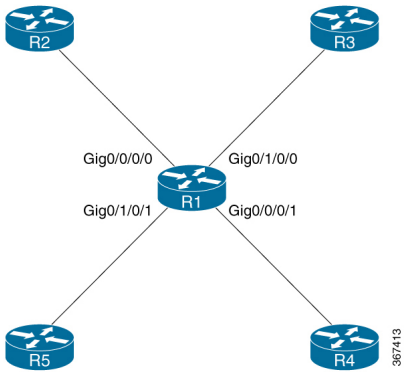
- **Boolean AND function—**When a tracked list has been assigned a Boolean AND function, each object that is defined within a subset must be in an "up" state. This condition enables the tracked object to be in the "up" state.
- **Boolean OR function—**When the tracked list has been assigned a Boolean OR function, at least one object that is defined within a subset must also be in an "up" state. This condition enables the tracked object to be in the "up" state.

Enhanced Object Tracking allows you to extend the track function to implement actions. These actions are triggered when the state of the object that is being tracked changes to "up" or "down". Based on the track state, you can error-disable one or more specified interfaces. Unless you configure the **auto-recover** keyword, the interfaces remain disabled even after the track state changes to the original state. You can configure **auto-recover** for each **action** configuration on a track.

In Figure 1, tracks named track1 and track2 are configured on router R1 to track the line protocol state of interfaces, GigabitEthernet0/0/0/1 and GigabitEthernet0/1/0/1 respectively. A track that is named track3 is configured to track track1 and track2 tracks with the Boolean logic AND. Therefore, track3 goes down if one or both the tracks, track1 and track2, go down. Track3 is also configured with the **action** command to put the interfaces GigabitEthernet0/0/0/0 and GigabitEthernet0/1/0/0 in a disabled state when track3 goes down.

Once the interfaces are error-disabled, they remain in the error-disabled state even if the track state changes to the "up" state. This is the default behaviour. To change this default behaviour, you can optionally configure the **auto-recover** keyword in the **action** command. If you configure the optional **auto-recover** keyword, the error-disabled state on the interfaces is cleared when the track state changes to the "up" state.

Figure 1: Enhanced Object Tracking



The same capability is available to progammatically manage the loopback interface as well. You can error-disable the loopback interface when the tracked-object state goes down.

For example, assume the loopback interface is the source of multiple tunnels. You can configure a track to monitor the line-protocol state of another interface with an action to error-disable the loopback interface when the tracked interface state goes down. When the line-protocol state of the tracked interface goes down, the loopback interface gets error-disabled, thereby bringing down all the associated tunnels. This avoids shutting down each tunnel interface individually.

As with the other interfaces, you can optionally configure the **auto-recover** keyword in the **action** command to clear the error-disabled state of the loopback interface when the track state changes back up. If this keyword is not configured, the loopback interface remains down even when the tracked interface state changes back up.

Restrictions for Enhanced Object Tracking

- You can perform Enhanced Object Tracking on physical interfaces, bundled interfaces, and loopback interface. Other virtual interfaces do not support this.
- The only action you can perform is error-disabling interfaces based on the state of a track (up/down).
- The maximum number of action interfaces that can be added under a single track is 1024.

How to Implement Object Tracking

This section describes the various object tracking procedures.

Tracking the Line Protocol State of an Interface

Perform this task in global configuration mode to track the line protocol state of an interface.

A tracked object is considered up when a line protocol of the interface is up.

After configuring the tracked object, you may associate the interface whose state should be tracked and specify the number of seconds to wait before the tracking object polls the interface for its state.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type line-protocol state**
4. **interface** *type interface-path-id*
5. **exit**
6. (Optional) **delay** {**up** *seconds* | **down** *seconds*}
7. Use one of the following commands:
   - **end**
   - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked. **Note** Special characters are not allowed in a *track-name*.
Step 3	**type line-protocol state** **Example:** `RP/0/RSP0/CPU0:router(config-track)# type line-protocol state`	Creates a track based on the line protocol of an interface.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-track-line-prot)# interface atm 0/2/0/0.1`	Specifies the interface to track the protocol state. • *type*—Specifies the interface type. For more information, use the question mark (?) online help function. • *interface-path-id*—Identifies a physical interface or a virtual interface. **Note** Use the **show interfaces** command to see a list of all possible interfaces currently configured on the router. **Note** The loopback and null interfaces are always in the up state and, therefore, cannot be tracked.

	Command or Action	Purpose
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-track-line-prot)# exit`	Exits the track line protocol configuration mode.
Step 6	(Optional) **delay** {**up** *seconds* | **down** *seconds*} **Example:** `RP/0/RSP0/CPU0:router(config-track)# delay up 10`	Schedules the delay that can occur between tracking whether the object is up or down.

Step 7	Use one of the following commands:
	• **end**
	• **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-track)# end`
	or
	`RP/0/RSP0/CPU0:router(config-track)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		• Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Tracking IP Route Reachability

When a host or a network goes down on a remote site, routing protocols notify the router and the routing table is updated accordingly. The routing process is configured to notify the tracking process when the route state changes due to a routing update.

A tracked object is considered up when a routing table entry exists for the route and the route is accessible.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type route reachability**
4. Use one of the following commands:
   - **vrf** *vrf-table-name*

- **route ipv4** *IP-prefix/mask*

5. **exit**
6. (Optional) **delay** {**up** *seconds* | **down** *seconds*}
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked. **Note** Special characters are not allowed in a *track-name*.
Step 3	**type route reachability** **Example:** `RP/0/RSP0/CPU0:router(config-track)# type route reachability vrf internet`	Configures the routing process to notify the tracking process when the state of the route changes due to a routing update.

Step 4	Use one of the following commands:
	• **vrf** *vrf-table-name*
	• **route ipv4** *IP-prefix/mask*
	Example:
	`RP/0/RSP0/CPU0:router(config-track-route)# vrf vrf-table-4`
	or
	`RP/0/RSP0/CPU0:router(config-track-route)# route ipv4 10.56.8.10/16`
		Configures the type of IP route to be tracked, which can consist of either of the following, depending on your router type:
		• *vrf-table-name*—A VRF table name.
		• *IP-prefix/mask*—An IP prefix consisting of the network and subnet mask (for example, 10.56.8.10/16).

Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-track-line-prot)# exit`	Exits the track line protocol configuration mode.
Step 6	(Optional) **delay** {**up** *seconds*	**down** *seconds*} **Example:** `RP/0/RSP0/CPU0:router(config-track)# delay up 10`	Schedules the delay that can occur between tracking whether the object is up or down.

	Command or Action	Purpose
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Building a Track Based on a List of Objects

Perform this task in the global configuration mode to create a tracked list of objects (which, in this case, are lists of interfaces or prefixes) using a Boolean expression to determine the state of the list.

A tracked list contains one or more objects. The Boolean expression enables two types of calculations by using either AND or OR operators. For example, when tracking two interfaces, using the AND operator, up means that *both* interfaces are up, and down means that *either* interface is down.

Note An object must exist before it can be added to a tracked list.

The NOT operator is specified for one or more objects and negates the state of the object.

After configuring the tracked object, you must associate the interface whose state should be tracked and you may optionally specify the number of seconds to wait before the tracking object polls the interface for its state.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type list boolean** { **and** | **or** }
4. **object** *object-name* [**not**]
5. **exit**
6. (Optional) **delay** {**up** *seconds* | **down** *seconds*}
7. Use one of the following commands:
   - **end**
   - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked. **Note** Special characters are not allowed in a *track-name*.
Step 3	**type list boolean** { **and** | **or** } **Example:** `RP/0/RSP0/CPU0:router(config-track-list)# type list boolean and`	Configures a Boolean list object and enters track list configuration mode. • **boolean**—Specifies that the state of the tracked list is based on a Boolean calculation. • **and**—Specifies that the list is up if all objects are up, or down if one or more objects are down. For example when tracking two interfaces, up means that both interfaces are up, and down means that either interface is down. • **or**—Specifies that the list is up if at least one object is up. For example, when tracking two interfaces, up means that either interface is up, and down means that both interfaces are down.
Step 4	**object** *object-name* [**not**] **Example:** `RP/0/RSP0/CPU0:router(config-track-list)# object 3 not`	Specifies the object to be tracked by the list • *obect-name*—Name of the object to track. • **not**—Negates the state of the object.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-track-line-prot)# exit`	Exits the track line protocol configuration mode.
Step 6	(Optional) **delay** {**up** *seconds* | **down** *seconds*} **Example:** `RP/0/RSP0/CPU0:router(config-track)# delay up 10`	Schedules the delay that can occur between tracking whether the object is up or down.
Step 7	Use one of the following commands: • **end** • **commit**	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes:

	Command or Action	Purpose

	Example:
	`RP/0/RSP0/CPU0:router(config-track)# end`
	or
	`RP/0/RSP0/CPU0:router(config-track)# commit`
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		• Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Building a Track Based on a List of Objects - Threshold Percentage

Perform this task in the global configuration mode to create a tracked list of objects (which, in this case, are lists of interfaces or prefixes) using a threshold percentage to determine the state of the list.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type list threshold percentage**
4. **object** *object-name*
5. **threshold percentage up** *percentage* **down** *percentage*
6. Use one of the following commands:
   - **end**
   - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:**	Enters track configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# track track1`	• *track-name*—Specifies a name for the object to be tracked. **Note** Special characters are not allowed in a *track-name*.
Step 3	**type list threshold percentage** **Example:** `RP/0/RSP0/CPU0:router(config-track-list)# type list threshold percentage`	Configures a track of type threshold percentage list.

Step 4	**object** *object-name*
	Example:
	`RP/0/RSP0/CPU0:router(config-track-list-threshold)# object 1`
	`RP/0/RSP0/CPU0:router(config-track-list-threshold)# object 2`
	`RP/0/RSP0/CPU0:router(config-track-list-threshold)# object 3`
	`RP/0/RSP0/CPU0:router(config-track-list-threshold)# object 4`
		Configures object 1, object 2, object 3 and object 4 as members of track type track1.

Step 5	**threshold percentage up** *percentage* **down** *percentage* **Example:** `RP/0/RSP0/CPU0:router(config-track-list-threshold)# threshold percentage up 50 down 33`	Configures the percentage of objects that need to be UP or DOWN for the list to be considered UP or Down respectively. For example, if object 1, object 2, and object 3 are in the UP state and object 4 is in the DOWN state, the list is considered to be in the UP state.

Step 6	Use one of the following commands:
	• **end**
	• **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-track)# end`
	or
	`RP/0/RSP0/CPU0:router(config-track)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		• Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Building a Track Based on a List of Objects - Threshold Weight

Perform this task in the global configuration mode to create a tracked list of objects (which, in this case, are lists of interfaces or prefixes) using a threshold weight to determine the state of the list.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type list threshold weight**
4. **object** *object-name* **weight** *weight*
5. **threshold weight up** *weight* **down** *weight*
6. Use one of the following commands:
   - **end**
   - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked. **Note** Special characters are not allowed in a *track-name*.
Step 3	**type list threshold weight** **Example:** `RP/0/RSP0/CPU0:router(config-track-list)# type list threshold weight`	Configures a a track of type, threshold weighted list.
Step 4	**object** *object-name* **weight** *weight* **Example:**	Configures object 1, object 2 and object 3 as members of track t1 and with weights 10, 5 and 3 respectively.

	Command or Action	Purpose

	`RP/0/RSP0/CPU0:router(config-track-list-threshold)# object 1 weight 10`
	`RP/0/RSP0/CPU0:router(config-track-list-threshold)# object 2 weight 5`
	`RP/0/RSP0/CPU0:router(config-track-list-threshold)# object 3 weight 3`	

Step 5	**threshold weight up** *weight* **down** *weight* **Example:** `RP/0/RSP0/CPU0:router(config-track-list-threshold)# threshold weight up 10 down 5`	Configures the range of weights for the objects that need to be UP or DOWN for the list to be considered UP or DOWN respectively. In this example, the list is considered to be in the DOWN state because objects 1 and 2 are in the UP state and the cumulative weight is 15 (not in the 10-5 range).

Step 6	Use one of the following commands:
	• **end**
	• **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-track)# end`
	or
	`RP/0/RSP0/CPU0:router(config-track)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		• Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Tracking IPSLA Reachability

Use this task to enable the tracking of the return code of IP service level agreement (SLA) operations.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type rtr** *ipsla-no* **reachability**
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track t1`	Enters track configuration mode. **Note** Special characters are not allowed in a *track-name*.
Step 3	**type rtr** *ipsla-no* **reachability** **Example:** `RP/0/RSP0/CPU0:router(config-track)# type rtr 100 reachability`	Specifies the IP SLA operation ID to be tracked for reachability. Values for the *ipsla-no* can range from1 to 2048.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring IPSLA Tracking: Example

This example shows the configuration of IPSLA tracking:

```
RP/0/RSP0/CPU0:router(config)# track track1
RP/0/RSP0/CPU0:router(config-track)# type rtr 1 reachability
RP/0/RSP0/CPU0:router(config-track)# delay up 5
RP/0/RSP0/CPU0:router(config-track)# delay down 10
```

Configure Enhanced Object Tracking

You can configure tracks with the **action** command to enable Enhanced Object Tracking. As a prerequisite, configure the track type that is to be tracked.

The following example shows how to configure the **action** command on a track based on the change in state of the track:

```
/* Configure track1 to track line-protocol state of the interface FourHundredGigE0/0/0/1
```

```
*/
Router#configure
Router(config)#track track1
Router(config-track)#type line-protocol state
Router(config-track-line-prot)#interface FourHundredGigE0/0/0/1
Router(config-track-line-prot)#exit
Router(config-track)#exit

/* Configure track2 to track line-protocol state of the interface FourHundredGigE0/1/0/1
*/
Router(config)#track track2
Router(config-track)#type line-protocol state
Router(config-track-line-prot)#interface FourHundredGigE0/1/0/1
Router(config-track-line-prot)#exit
Router(config-track)#exit

/* Configure track3 with boolean AND of track1 state and track2 state. Specify actions to
take when track3 state changes. */
Router(config)#track track3
Router(config-track)#type list boolean and
Router(config-track-list-boolean)#object track1
Router(config-track-list-boolean)#object track2
Router(config-track-line-boolean)#exit
Router(config-track)#action
Router(config-track-action)#track-down error-disable interface FourHundredGigE0/0/0/0
auto-recover
Router(config-track-action)#track-down error-disable interface FourHundredGigE0/1/0/0

/* Configure track4 to track line-protocol state of the HundredGigE0/0/0/35 interface and
take action on loopback interface. */
Router(config)#track track4
Router(config-track)#type line-protocol state
Router(config-track-line-prot)#interface HundredGigE0/0/0/35
Router(config-track-line-prot)#exit

/* Specify action to take when track4 state changes. In this example, the action happens
when the track state changes to up. */
Router(config-track)#action
Router(config-track-action)#track-up error-disable interface Loopback100 auto-recover
Router(config-track)#exit
Router(config)#end
```

The following running configuration example shows you how to configure the **action** command for the scenario described in Figure 1.

```
track track1
 type line-protocol state
  interface FourHundredGigE0/0/0/1
 !
!
track track2
 type line-protocol state
  interface FourHundredGigE0/1/0/1
 !
!
track track3
 type list boolean and
  object track1
  object track2
 !
 action
  track-down error-disable interface FourHundredGigE0/0/0/0 auto-recover
  track-down error-disable interface FourHundredGigE0/1/0/0
```

The following example shows you how to configure the **action** command for the loopback interface.

```
track track4
 type line-protocol state
  interface HundredGigE0/0/0/35
 !
 action
  track-up error-disable interface Loopback100 auto-recover
 !
!
```

Verification

To view the state of the track, use the **show track** command.

Initially, let us assume the line-protocol state of FourHundredGigE0/0/0/1 (track1 interface) and FourHundredGigE0/1/0/1 (track2 interface) are up and HundredGigE0/0/0/35 (track4 interface) is down.

```
Router#show track
Track track3
        List boolean and is UP
        7 changes, last change 16:04:28 IST Mon Jul 02 2018
                object track2 UP
                object track1 UP
Track track1
        Interface FourHundredGigE0/0/0/1 line-protocol
        Line protocol is UP
        7 changes, last change 16:04:28 IST Mon Jul 02 2018
Track track2
        Interface FourHundredGigE0/1/0/1 line-protocol
        Line protocol is UP
        7 changes, last change 16:02:41 IST Mon Jul 02 2018

Track track4
        Interface HundredGigE0/0/0/35 line-protocol
        Line protocol is DOWN
        2 changes, last change 06:28:06 UTC Tue Jun 27 2023
        Delay up 0 secs(default), down 0 secs(default)
```

To verify if the interface configured for tracking is error-disabled, use the **show error-disable** command. As none of the track states match the track-action state, there are no error-disabled interfaces.

```
Router#show error-disable
Interface            Error-Disable reason            Retry (s)  Time disabled
-------------------  ------------------------------  ---------  ---------------
There are no interfaces error-disabled matching the given criteria
```

To view the status of all the interfaces of the tracked object, use the **show interface brief** command.

```
Router#show interface brief
Intf Name                   Intf State LineP State    Encap Type   MTU (byte) BW (Kbps)
Lo100                       up         up               Loopback   1500       0
FourHundredGigE0/0/0/0      up         up               ARPA       1514       100000000
FourHundredGigE0/0/0/1      up         up               ARPA       1514       100000000
FourHundredGigE0/1/0/0      up         up               ARPA       1514       100000000
FourHundredGigE0/1/0/1      up         up               ARPA       1514       100000000
HundredGigE0/0/0/35         admin-down admin-down       ARPA       1514       100000000
```

When a track state changes, the corresponding track action happens and the status of the interfaces configured in the action changes. The state of track3 becomes "down" when either track1 state or track2 state becomes "down". The state of track4 becomes "up" when the HundredGigE0/0/0/35 interface comes up. The **show error-disable** command displays the following output when track3 state is down and track4 state is up.

```
Router#show error-disable
Interface            Error-Disable reason            Retry (s)  Time disabled
-------------------  ------------------------------  ---------  ---------------
Loopback100          ot-track-state-change             ---      08:44:07
FH0/0/0/0            ot-track-state-change             ---      08:42:08
FH0/1/0/0            ot-track-state-change             ---      08:42:01
```

When track3 state is down and track4 state is up, the **show interface brief** command displays the following output.

```
Router#show interface brief
Intf Name                Intf State      LineP State    Encap Type   MTU (byte)   BW (Kbps)
Lo100                    err-disable     admin-down     Loopback     1500          0
FourHundredGigE0/0/0/0   err-disable     admin-down     ARPA         1514          100000000
FourHundredGigE0/0/0/1   err-disable     admin-down     ARPA         1514          100000000
FourHundredGigE0/1/0/0   err-disable     admin-down     ARPA         1514          100000000
FourHundredGigE0/1/0/1   up              up             ARPA         1514          100000000
HundredGigE0/0/0/35      up              up             ARPA         1514          100000000
```

When track3 state comes back up, the error-disable status on the interface FourHundredGigE0/0/0/0 clears. This is because of the **auto-recover** configuration for FourHundredGigE0/0/0/0. However, interface FourHundredGigE0/1/0/0 remains in the error-disable status because **auto-recover** isn't configured on this interface.

Similarly, when track4 state goes down, the error-disable status on the interface Loopback100 clears because of the **auto-recover** configuration for track4.

The change reflects in the output of the **show interface brief** command.

```
RP/0/0/CPU0:ios#show interface brief
Intf Name                Intf State  LineP State  Encap Type  MTU (byte) BW (Kbps)
Lo100                    up          up           Loopback    1500       0
FourHundredGigE0/0/0/0   up          up           ARPA        1514       100000000
FourHundredGigE0/0/0/1   up          up           ARPA        1514       100000000
FourHundredGigE0/1/0/0   err-disable admin-down   ARPA        1514       100000000
FourHundredGigE0/1/0/1   up          up           ARPA        1514       100000000
HundredGigE0/0/0/35      admin-down  admin-down   ARPA        1514       100000000
```

Configuration Examples for Configuring Object Tracking

Configuring IPSLA Tracking: Example

This example shows the configuration of IPSLA tracking, including the ACL and IPSLA configuration:

ACL configuration:

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list abf-track
RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit any any nexthop track track1 1.2.3.4
```

Object tracking configuration:

```
RP/0/RSP0/CPU0:router(config)# track track1
RP/0/RSP0/CPU0:router(config-track)# type rtr 1 reachability
RP/0/RSP0/CPU0:router(config-track)# delay up 5
RP/0/RSP0/CPU0:router(config-track)# delay down 10
```

IPSLA configuration:

```
RP/0/RSP0/CPU0:router(config)# ipsla
RP/0/RSP0/CPU0:router(config-ipsla)# operation 1
RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp echo
RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# source address 2.3.4.5
RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# destination address 1.2.3.4
RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# frequency 60
RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# exit
RP/0/RSP0/CPU0:router(config-ipsla-op)# exit
RP/0/RSP0/CPU0:router(config-ipsla)# schedule operation 1
RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time now
RP/0/RSP0/CPU0:router(config-ipsla-sched)# life forever
```

Additional References

The following sections provide references related to implementing object tracking for IPSec network security.

Related Documents

Related Topic	Document Title
IP SLA configuration information	*Implementing IP Service Level Agreements on the Cisco ASR 9000 Series Router* module in *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*
IP SLA commands	*IP Service Level Agreement Commands on the Cisco ASR 9000 Series Router* module in *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Object tracking commands	*Object Tracking Commands on the Cisco ASR 9000 Series Router* module in *System Management Command Reference for Cisco ASR 9000 Series Routers*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 2401	*Security Architecture for the Internet Protocol*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html