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 9. 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 for Implementing Object Tracking

Release	Modification
Release 4.0.0	This feature was introduced.
Release 4.2.1	The ability to create a tracked list based on a threshold percentage or weight was added.

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.

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**	
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.

	Command or Action	**Purpose**
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.
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.

Command or Action	Purpose
	• 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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.
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:**	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).

	Command or Action	Purpose

	`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`	

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	**commit**	

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**	
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.
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** **Example:** RP/0/RSP0/CPU0:router(config-track)# end or	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]:

Command or Action	Purpose
`RP/0/RSP0/CPU0:router(config-track)# commit`	• 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. **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 t1**	Enters track configuration mode.
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 from 1 to 2048.
Step 4	**commit**	

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
```

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