



## 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 **Additional References** section. To locate documentation for other commands that might appear in the course of performing a configuration task, see **Technical Documentation** section in the Additional References topic.

- [Configuring Object Tracking, on page 1](#)
- [Prerequisites for Implementing Object Tracking, on page 1](#)
- [Information about Object Tracking, on page 2](#)
- [How to Implement Object Tracking, on page 2](#)
- [Configuration Examples for Configuring Object Tracking, on page 11](#)
- [Additional References, on page 14](#)

## 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 **Additional References** section. To locate documentation for other commands that might appear in the course of performing a configuration task, see **Technical Documentation** section in the Additional References topic.

## 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.



---

**Note** Object Tracking is an optional package. You must check if this package is installed on your system by running the command **show install active summary**.

---

## Information about Object Tracking

*Object tracking* is a mechanism to track an object and to take an action on another object with no relationship to the tracked objects, based on changes to the properties of the object being tracked.

Each tracked object is identified by a unique name specified on the tracking command-line interface (CLI). Cisco IOS XR processes then use this name to track a specific object.

The tracking process periodically polls the tracked object and reports any changes to its state in terms of its being up or down, either immediately or after a delay, as configured by the user.

Multiple objects can also be tracked by means of a list, using 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 defined within a subset must be in an up state, so that the tracked object can also be in the up state.
- **Boolean OR function**—When the tracked list has been assigned a Boolean OR function, it means that at least one object defined within a subset must also be in an up state, so that the tracked object can also 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.

#### Procedure

|               | Command or Action  | Purpose   |
|---------------|--|---|
| <b>Step 1</b> | <b>configure</b><br><b>Example:</b><br>RP/0/RP0/CPU0:router# configure                   | Enters mode.  |
| <b>Step 2</b> | <b>track track-name</b><br><b>Example:</b><br>RP/0/RP0/CPU0:router(config)# track track1 | Enters track configuration mode. <ul style="list-style-type: none"> <li>• <i>track-name</i>—Specifies a name for the object to be tracked.</li> </ul> |
| <b>Step 3</b> | <b>type line-protocol state</b><br><b>Example:</b>                                       | Creates a track based on the line protocol of an interface.   |

|               | Command or Action   | Purpose  |
|---------------|---|--|
|               | RP/0/RP0/CPU0:router(config-track)# type<br>line-protocol state   |  |
| <b>Step 4</b> | <p><b>interface</b> <i>type</i> <i>interface-path-id</i></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track-line-prot)# interface atm 0/2/0/0.1</pre>  | <p>Specifies the interface to track the protocol state.</p> <ul style="list-style-type: none"> <li>• <i>type</i>—Specifies the interface type. For more information, use the question mark (?) online help function.</li> <li>• <i>interface-path-id</i>—Identifies a physical interface or a virtual interface.</li> </ul> <p><b>Note</b> Use the <b>show interfaces</b> command to see a list of all possible interfaces currently configured on the router.</p> <p><b>Note</b> The loopback and null interfaces are always in the up state and, therefore, cannot be tracked.</p> |
| <b>Step 5</b> | <p><b>exit</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track-line-prot)# exit</pre>   | Exits the track line protocol configuration mode.  |
| <b>Step 6</b> | <p>(Optional) <b>delay</b> {<b>up</b> <i>seconds</i>   <b>down</b> <i>seconds</i>}</p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# delay up 10</pre>  | Schedules the delay that can occur between tracking whether the object is up or down.  |
| <b>Step 7</b> | <p>Use one of the following commands:</p> <ul style="list-style-type: none"> <li>• <b>end</b></li> <li>• <b>commit</b></li> </ul> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# end or RP/0/RP0/CPU0:router(config-track)# commit</pre> | <p>Saves configuration changes.</p> <ul style="list-style-type: none"> <li>• When you issue the <b>end</b> command, the system prompts you to commit changes:</li> </ul> <pre>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</pre> <ul style="list-style-type: none"> <li>• Entering <b>yes</b> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</li> <li>• Entering <b>no</b> exits the configuration session and returns the router to</li> </ul>             |

|  | Command or Action | Purpose   |
|--|-------------------|---|
|  |                   | <p>EXEC mode without committing the configuration changes.</p> <ul style="list-style-type: none"> <li>• Entering <b>cancel</b> leaves the router in the current configuration session without exiting or committing the configuration changes.</li> <li>• Use the <b>commit</b> command to save the configuration changes to the running configuration file and remain within the configuration session.</li> </ul> |

## 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.

### Procedure

|               | Command or Action  | Purpose  |
|---------------|--|--|
| <b>Step 1</b> | <p><b>configure</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router# configure</pre>  | Enters mode.   |
| <b>Step 2</b> | <p><b>track</b> <i>track-name</i></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config)# track track1</pre>   | <p>Enters track configuration mode.</p> <ul style="list-style-type: none"> <li>• <i>track-name</i>—Specifies a name for the object to be tracked.</li> </ul>   |
| <b>Step 3</b> | <p><b>type route reachability</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# type route reachability vrf internet</pre>   | Configures the routing process to notify the tracking process when the state of the route changes due to a routing update.   |
| <b>Step 4</b> | <p>Use one of the following commands:</p> <ul style="list-style-type: none"> <li>• <b>vrf</b> <i>vrf-table-name</i></li> <li>• <b>route ipv4</b> <i>IP-prefix/mask</i></li> </ul> <p><b>Example:</b></p> | <p>Configures the type of IP route to be tracked, which can consist of either of the following, depending on your router type:</p> <ul style="list-style-type: none"> <li>• <i>vrf-table-name</i>—A VRF table name.</li> </ul> |

|               | Command or Action  | Purpose   |
|---------------|--|---|
|               | <pre>RP/0/RP0/CPU0:router(config-track-route)# vrf vrf-table-4  or  RP/0/RP0/CPU0:router(config-track-route)# route ipv4 10.56.8.10/16</pre>                             | <ul style="list-style-type: none"> <li>• <i>IP-prefix/mask</i>—An IP prefix consisting of the network and subnet mask (for example, 10.56.8.10/16).</li> </ul>  |
| <b>Step 5</b> | <p><b>exit</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track-line-prot)# exit</pre>  | Exits the track line protocol configuration mode.   |
| <b>Step 6</b> | <p>(Optional) <b>delay</b> {<b>up</b> <i>seconds</i>   <b>down</b> <i>seconds</i>}</p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# delay up 10</pre> | Schedules the delay that can occur between tracking whether the object is up or down.   |
| <b>Step 7</b> | Use the <b>commit</b> or <b>end</b> command.   | <p><b>commit</b> —Saves the configuration changes, and remains within the configuration session.</p> <p><b>end</b> —Prompts user to take one of these actions:</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> — Saves configuration changes and exits the configuration session.</li> <li>• <b>No</b> —Exits the configuration session without committing the configuration changes.</li> <li>• <b>Cancel</b> —Remains in the configuration mode, without committing the configuration changes.</li> </ul> |

## 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.

### Procedure

|               | Command or Action  | Purpose   |
|---------------|--|---|
| <b>Step 1</b> | <b>configure</b><br><b>Example:</b><br><pre>RP/0/RP0/CPU0:router# configure</pre>  | Enters mode.  |
| <b>Step 2</b> | <b>track track-name</b><br><b>Example:</b><br><pre>RP/0/RP0/CPU0:router(config)# track track1</pre>                              | Enters track configuration mode. <ul style="list-style-type: none"> <li>• <i>track-name</i>—Specifies a name for the object to be tracked.</li> </ul>   |
| <b>Step 3</b> | <b>type list boolean { and   or }</b><br><b>Example:</b><br><pre>RP/0/RP0/CPU0:router(config-track)# type list boolean and</pre> | Configures a Boolean list object and enters track list configuration mode. <ul style="list-style-type: none"> <li>• <b>boolean</b>—Specifies that the state of the tracked list is based on a Boolean calculation.</li> <li>• <b>and</b>—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.</li> <li>• <b>or</b>—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.</li> </ul> |
| <b>Step 4</b> | <b>object object-name [ not ]</b><br><b>Example:</b><br><pre>RP/0/RP0/CPU0:router(config-track-list)# object 3 not</pre>         | Specifies the object to be tracked by the list <ul style="list-style-type: none"> <li>• <i>object-name</i>—Name of the object to track.</li> <li>• <b>not</b>—Negates the state of the object.</li> </ul>   |
| <b>Step 5</b> | <b>exit</b><br><b>Example:</b><br><pre>RP/0/RP0/CPU0:router(config-track-line-prot)# exit</pre>                                  | Exits the track line protocol configuration mode.   |

|               | Command or Action   | Purpose   |
|---------------|---|---|
| <b>Step 6</b> | <p>(Optional) <b>delay</b> { <b>up</b> <i>seconds</i>   <b>down</b> <i>seconds</i> }</p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# delay up 10</pre>  | Schedules the delay that can occur between tracking whether the object is up or down.   |
| <b>Step 7</b> | <p>Use one of the following commands:</p> <ul style="list-style-type: none"> <li>• <b>end</b></li> <li>• <b>commit</b></li> </ul> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# end or RP/0/RP0/CPU0:router(config-track)# commit</pre> | <p>Saves configuration changes.</p> <ul style="list-style-type: none"> <li>• When you issue the <b>end</b> command, the system prompts you to commit changes: <pre>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</pre> <ul style="list-style-type: none"> <li>• Entering <b>yes</b> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</li> <li>• Entering <b>no</b> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</li> <li>• Entering <b>cancel</b> leaves the router in the current configuration session without exiting or committing the configuration changes.</li> </ul> </li> <li>• Use the <b>commit</b> command to save the configuration changes to the running configuration file and remain within the configuration session.</li> </ul> |

## 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.

### Procedure

|               | Command or Action   | Purpose      |
|---------------|---|--------------|
| <b>Step 1</b> | <p><b>configure</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router# configure</pre> | Enters mode. |

|               | Command or Action   | Purpose   |
|---------------|---|---|
| <b>Step 2</b> | <p><b>track</b> <i>track-name</i></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config)# track track1</pre>  | <p>Enters track configuration mode.</p> <ul style="list-style-type: none"> <li>• <i>track-name</i>—Specifies a name for the object to be tracked.</li> </ul>  |
| <b>Step 3</b> | <p><b>type list threshold percentage</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# type list threshold percentage</pre>   | <p>Configures a track of type threshold percentage list.</p>  |
| <b>Step 4</b> | <p><b>object</b> <i>object-name</i></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track-list-threshold)# object 1 RP/0/RP0/CPU0:router(config-track-list-threshold)# object 2 RP/0/RP0/CPU0:router(config-track-list-threshold)# object 3 RP/0/RP0/CPU0:router(config-track-list-threshold)# object 4</pre> | <p>Configures object 1, object 2, object 3 and object 4 as members of track type track1.</p>  |
| <b>Step 5</b> | <p><b>threshold percentage up percentage down</b><br/><i>percentage</i></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track-list-threshold)# threshold percentage up 50 down 33</pre>   | <p>Configures the percentage of objects that need to be UP or DOWN for the list to be considered UP or Down respectively.</p> <p>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.</p>  |
| <b>Step 6</b> | <p>Use one of the following commands:</p> <ul style="list-style-type: none"> <li>• <b>end</b></li> <li>• <b>commit</b></li> </ul> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# end or RP/0/RP0/CPU0:router(config-track)# commit</pre>   | <p>Saves configuration changes.</p> <ul style="list-style-type: none"> <li>• When you issue the <b>end</b> command, the system prompts you to commit changes:</li> </ul> <pre>Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:</pre> <ul style="list-style-type: none"> <li>• Entering <b>yes</b> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</li> <li>• Entering <b>no</b> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</li> </ul> |



|  | Command or Action | Purpose  |
|--|-------------------|--|
|  |                   | <ul style="list-style-type: none"> <li>• Entering <b>cancel</b> leaves the router in the current configuration session without exiting or committing the configuration changes.</li> <li>• Use the <b>commit</b> command to save the configuration changes to the running configuration file and remain within the configuration session.</li> </ul> |

## 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.

### Procedure

|               | Command or Action   | Purpose   |
|---------------|---|---|
| <b>Step 1</b> | <b>configure</b><br><b>Example:</b><br>RP/0/RP0/CPU0:router# configure  | Enters mode.  |
| <b>Step 2</b> | <b>track track-name</b><br><b>Example:</b><br>RP/0/RP0/CPU0:router(config)# track track1  | Enters track configuration mode. <ul style="list-style-type: none"> <li>• <i>track-name</i>—Specifies a name for the object to be tracked.</li> </ul> |
| <b>Step 3</b> | <b>type list threshold weight</b><br><b>Example:</b><br>RP/0/RP0/CPU0:router(config-track)# type list threshold weight  | Configures a track of type, threshold weighted list.  |
| <b>Step 4</b> | <b>object object-name weight weight</b><br><b>Example:</b><br>RP/0/RP0/CPU0:router(config-track-list-threshold)# object 1 weight 10<br>RP/0/RP0/CPU0:router(config-track-list-threshold)# object 2 weight 5<br>RP/0/RP0/CPU0:router(config-track-list-threshold)# object 3 weight 3 | Configures object 1, object 2 and object 3 as members of track t1 and with weights 10, 5 and 3 respectively.  |
| <b>Step 5</b> | <b>threshold weight up weight down weight</b><br><b>Example:</b>  | 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             |

|               | Command or Action   | Purpose   |
|---------------|---|---|
|               | <pre>RP/0/RP0/CPU0:router(config-track-list-threshold)# threshold weight up 10 down 5</pre>   | 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).  |
| <b>Step 6</b> | <p>Use one of the following commands:</p> <ul style="list-style-type: none"> <li>• <b>end</b></li> <li>• <b>commit</b></li> </ul> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config-track)# end or RP/0/RP0/CPU0:router(config-track)# commit</pre> | <p>Saves configuration changes.</p> <ul style="list-style-type: none"> <li>• When you issue the <b>end</b> command, the system prompts you to commit changes: <pre>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</pre> <ul style="list-style-type: none"> <li>• Entering <b>yes</b> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</li> <li>• Entering <b>no</b> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</li> <li>• Entering <b>cancel</b> leaves the router in the current configuration session without exiting or committing the configuration changes.</li> </ul> </li> <li>• Use the <b>commit</b> command to save the configuration changes to the running configuration file and remain within the configuration session.</li> </ul> |

## Tracking IPSLA Reachability

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

### Procedure

|               | Command or Action   | Purpose   |
|---------------|---|---|
| <b>Step 1</b> | <p><b>configure</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router# configure</pre>               | Enters global configuration mode.   |
| <b>Step 2</b> | <p><b>track track-name</b></p> <p><b>Example:</b></p> <pre>RP/0/RP0/CPU0:router(config)# track t1</pre> | <p>Enters track configuration mode.</p> <p><b>Note</b> Special characters are not allowed in a <i>track-name</i>.</p> |

|               | Command or Action   | Purpose  |
|---------------|---|--|
| <b>Step 3</b> | <b>type rtr <i>ipsla-no</i> reachability</b><br><b>Example:</b><br>RP/0/RP0/CPU0:router(config-track)# <b>type rtr 100 reachability</b> | Specifies the IP SLA operation ID to be tracked for reachability. Values for the <i>ipsla-no</i> can range from 1 to 2048.   |
| <b>Step 4</b> | Use the <b>commit</b> or <b>end</b> command.  | <b>commit</b> —Saves the configuration changes and remains within the configuration session.<br><b>end</b> —Prompts user to take one of these actions: <ul style="list-style-type: none"> <li>• <b>Yes</b> — Saves configuration changes and exits the configuration session.</li> <li>• <b>No</b> —Exits the configuration session without committing the configuration changes.</li> <li>• <b>Cancel</b> —Remains in the configuration session, without committing the configuration changes.</li> </ul> |

### Configuring IPSLA Tracking: Example

This example shows the configuration of IPSLA tracking:

```
RP/0/RP0/CPU0:router(config)# track track1
RP/0/RP0/CPU0:router(config-track)# type rtr 1 reachability
RP/0/RP0/CPU0:router(config-track)# delay up 5
RP/0/RP0/CPU0:router(config-track)# delay down 10
```

## Configuration Examples for Configuring Object Tracking

### Tracking Whether the Interface Is Up or Down: Running Configuration Example

```
track connection100
  type list boolean and
  object object3 not
  delay up 10
!
interface service-ipsec 23
  line-protocol track connection100
!
```

### Tracking the Line Protocol State of an Interface: Running Configuration Example

In this example, traffic arrives from interface service-ipsec1 and exits through interface gigabitethernet0/0/0/3:

```
track IPsec1
  type line-protocol state
  interface gigabitethernet0/0/0/3
  !
interface service-ipsec 1
  ipv4 address 70.0.0.1 255.255.255.0
  profile vrf1_profile_ipsec
  line-protocol track IPsec1
  tunnel source 80.0.0.1
  tunnel destination 80.0.0.2
  service-location preferred-active 0/0/1
  !
```

This example displays the output from the **show track** command after performing the previous example:

```
RP/0/RP0/CPU0:router# show run track

Track IPsec1
Interface GigabitEthernet0_0_0_3 line-protocol
!
  Line protocol is UP
  1 change, last change 10:37:32 UTC Thu Sep 20 2007
  Tracked by:
  service-ipsec1
  !
```

### Tracking IP Route Reachability: Running Configuration Example

In this example, traffic arriving from interface service-ipsec1 has its destination in network 7.0.0.0/24. This tracking procedure follows the state of the routing protocol prefix to signal when there are changes in the routing table.

```
track PREFIX1
  type route reachability
  route ipv4 7.0.0.0/24
  !
interface service-ipsec 1
  vrf 1
  ipv4 address 70.0.0.2 255.255.255.0
  profile vrf_1_ipsec
  line-protocol track PREFIX1
  tunnel source 80.0.0.2
  tunnel destination 80.0.0.1
  service-location preferred-active 0/2/0
```

### Building a Track Based on a List of Objects: Running Configuration Example

In this example, traffic arriving from interface service-ipsec1 exits through interface gigabitethernet0/0/0/3 and interface ATM 0/2/0/0.1. The destination of the traffic is at network 7.0.0.0/24.

If either one of the interfaces or the remote network goes down, the flow of traffic must stop. To do this, we use a Boolean AND expression.

```
track C1
  type route reachability
  route ipv4 3.3.3.3/32
  !
!
track C2
  type route reachability
  route ipv4 1.2.3.4/32
  !
!
track C3
  type route reachability
  route ipv4 10.0.20.2/32
  !
!
track C4
  type route reachability
  route ipv4 10.0.20.0/24
  !
!
track OBJ
  type list boolean and
  object C1
  object C2
  !
!
track OBJ2
  type list boolean or
  object C1
  object C2
  !
```

### Configuring IPSLA based Object Tracking: Configuration Example

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

ACL configuration:

```
RP/0/RP0/CPU0:router(config)# ipv4 access-list abf-track
RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit any nexthop track track1 1.2.3.4
```

Object tracking configuration:

```
RP/0/RP0/CPU0:router(config)# track track1
RP/0/RP0/CPU0:router(config-track)# type rtr 1 reachability
RP/0/RP0/CPU0:router(config-track)# delay up 5
```

```
RP/0/RP0/CPU0:router(config-track)# delay down 10
```

IPSLA configuration:

```
RP/0/RP0/CPU0:router(config)# ipsla
RP/0/RP0/CPU0:router(config-ipsla)# operation 1
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp echo
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# source address 2.3.4.5
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# destination address 1.2.3.4
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# frequency 60
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# schedule operation 1
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time now
RP/0/RP0/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 | <i>Implementing IP Service Level Agreements on System Monitoring Configuration Guide for Cisco NCS 560 Series Routers</i>  |
| IP SLA commands                  | <i>IP Service Level Agreement Commands on System Monitoring Command Reference for Cisco NCS 5500 Series Routers and Cisco NCS 540 and NCS 560 Series Routers</i> |

### 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:<br><a href="https://cfnng-stg.cisco.com/mibs">https://cfnng-stg.cisco.com/mibs</a> . |

**RFCs**

| <b>RFCs</b> | <b>Title</b>   |
|-------------|--|
| RFC 2401    | <i>Security Architecture for the Internet Protocol</i> |

**Technical Assistance**

| <b>Description</b>  | <b>Link</b>   |
|---|---|
| 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. | <a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a> |

