

Chapter 31

Filtering IP Routes

Introduction	31-3
Types of Filters	31-4
About Prefix Lists	31-4
About AS Path Lists	31-5
About Route Maps	31-5
About IP Route Filters	31-7
About IP Filters	31-8
Creating Filters	31-9
Creating Prefix Lists	31-9
Creating AS Path Lists for BGP Routes	31-9
Creating Route Maps for BGP Routes	31-10
Creating Route Maps for OSPF Routes	31-16
Creating Route Maps for Redistributing BGP Routes into RIP	31-19
Creating IP Route Filters	31-21
Creating IP Filters	31-21
Applying Filters	31-22
Applying Filters When Writing to the RIB	31-22
Applying Filters When Redistributing from the RIB	31-24
Applying Filters Before Advertising Routes	31-27
Overview of Filters for each Route Source	31-29
Border Gateway Protocol (BGP-4)	31-29
Open Shortest Path First (OSPF)	31-30
Routing Information Protocol (RIP)	31-32
Interface Routes	31-33
Statically-Configured Routes	31-33
Configuration Examples	31-34
Filtering When Writing BGP Routes to the RIB: Using an AS Path Filter ..	31-34
Filtering When Writing BGP Routes to the RIB: Using a Route Map	31-35
Filtering Before Advertising Routes with BGP: Using an AS Path Filter ..	31-36
Filtering Before Advertising Routes with BGP: Using a Route Map	31-37
Filtering Inbound and Outbound BGP Routes: Using Communities	31-38
Filtering When Importing Routes from BGP to OSPF	31-39
Command Reference	31-40
add ip aspathlist	31-40
add ip communitylist	31-42
add ip prefixlist	31-44
add ip route filter	31-46
add ip routemap	31-49
delete ip aspathlist	31-56
delete ip communitylist	31-57
delete ip prefixlist	31-57

delete ip route filter	31-58
delete ip routemap	31-59
set ip prefixlist	31-61
set ip route filter	31-63
set ip routemap	31-66
show ip aspathlist	31-73
show ip communitylist	31-74
show ip prefixlist	31-75
show ip route filter	31-76
show ip routemap	31-77

Introduction

This chapter describes the switch's functions for filtering IP routes. IP route filtering enables you to control your routing tables, for example, to meet the terms of business relationships you have with the networks to which you are connected.

If you are a network provider, you can filter the routing information that your switches receive from the networks they connect to, and that they advertise to those networks. This gives you control over the path of any traffic originating from or traversing your network. Usually, one or more of your switches form peer relationships with switches at other ISPs with which you have entered into data transporting agreements. The process of filtering is, in effect, the process of specifying the routes that your switches send or receive from each of their peers.

The switch provides several different mechanisms for filtering routes. Some of the functionality of these mechanisms overlaps, so sometimes you can achieve a given filtering effect in several ways. This chapter discusses all the different mechanisms and places them in context within the overall picture of how you can filter routes.

In general terms, configuring any filter involves the following three steps, which this chapter describes:

1. Select the required filter type, as described in [Types of Filters](#).
2. Create the filter, as described in [Creating Filters](#).
3. Apply it, as described in [Applying Filters](#).

When to use filters

You can use route filtering to select which routes:

- the switch copies from a routing protocol into its Routing Information Base (RIB). This determines which routes the switch uses to send traffic ([Applying Filters When Writing to the RIB](#)).
- the switch copies from its RIB into a routing protocol. This determines which routes the protocol has available for advertising to neighbouring devices ([Applying Filters When Redistributing from the RIB](#)).
- routing protocols actually advertise to neighbouring devices ([Applying Filters Before Advertising Routes](#)).

The RIB is another term for the switch's main IP route table, which is described in "The Routing Table" on page 23-23 of [Chapter 23, Internet Protocol \(IP\)](#).

Types of routes you can filter

As explained above, this chapter first divides the information about filtering into sections about each type of filter, rather than each type of routing protocol. Then it summarises the available filters for each routing protocol, in the following sections:

- [Border Gateway Protocol \(BGP-4\)](#)
- [Open Shortest Path First \(OSPF\)](#)
- [Routing Information Protocol \(RIP\)](#)
- [Interface Routes](#)
- [Statically-Configured Routes](#)

Types of Filters

The type of filter to use depends on the route source and the point at which you want to filter. This section describes the available filters, in the following subsections:

- [About Prefix Lists](#)
- [About AS Path Lists](#)
- [About Route Maps](#)
- [About IP Route Filters](#)
- [About IP Filters](#) for filtering routes (now superseded by other filter types)

This section describes each of these types of filters and summarises the circumstances in which you use them.

About Prefix Lists

- Description** A prefix list is a list of entries, each of which specifies:
- an IPv4 prefix, and a mask length or range of mask lengths
 - whether those prefixes explicitly match or explicitly do not match the prefix list

- When to use prefix lists** Prefix lists offer detailed control over which routes you import, export or advertise.

[“Applying Filters” on page 31-22](#) describes in detail how to use prefix lists, but this section summarises the uses.

For BGP, you can use prefix lists when:

- copying routes from an update message to the RIB, by using the prefix list:
 - directly as a filter, by making it the **infilter** on a BGP peer.
 - in a route map and applying the route map as the **inroutemap** on a BGP peer
- determining which routes to import from other route sources, by using the prefix list in a route map and applying the route map to the import entry
- determining which routes to advertise, by using the prefix list:
 - directly as a filter, by making it the **outfilter** on a BGP peer
 - in a route map and applying the route map as the **outroutemap** on a BGP peer

For BGP, prefix filtering can reject some of the routes from an update message, without rejecting the whole update. This enables you to configure the switch to accept only routes for particular networks from a particular peer, and to send only routes for particular networks to a particular peer.

When you apply a prefix list as an **infilter** or **outfilter** on a BGP peer, BGP looks at the individual prefixes within each update message, and compares them against the list. If a prefix in the update matches a prefix in the prefix list, BGP rejects that route. Otherwise, it accepts the route.

For OSPF, you can use prefix lists in a route map, and then use the route map:

- to filter OSPF routes before adding them to the RIB
- when importing static routes into the OSPF LSA database

About AS Path Lists

Description In BGP, the *AS_path* attribute lists the AS numbers of every Autonomous System that the routing information in an update message has passed through. It shows the path the update message has taken, and how “close” the routes are to the switch.

AS path lists let you filter to accept or reject update messages on the basis of all or part of their AS path. They look at the *AS_path* attribute in BGP update messages. If the attribute in the update message matches the filter criteria then the whole update message is filtered out (or accepted, depending on what action the filter entry has been configured to carry out).

When to use AS path lists You can only use AS path lists with BGP. [“Applying Filters” on page 31-22](#) describes in detail how to use AS path lists, but this section summarises the uses.

For BGP, you can use AS path lists when:

- copying routes from an update message to the RIB, by using the AS path list:
 - as the **inpathfilter** on a BGP peer.
 - in a route map and applying the route map as the **inroutemap** on a BGP peer
- determining which routes to advertise, by using the AS path list:
 - as the **outpathfilter** on a BGP peer
 - in a route map and applying the route map as the **outroutemap** on a BGP peer

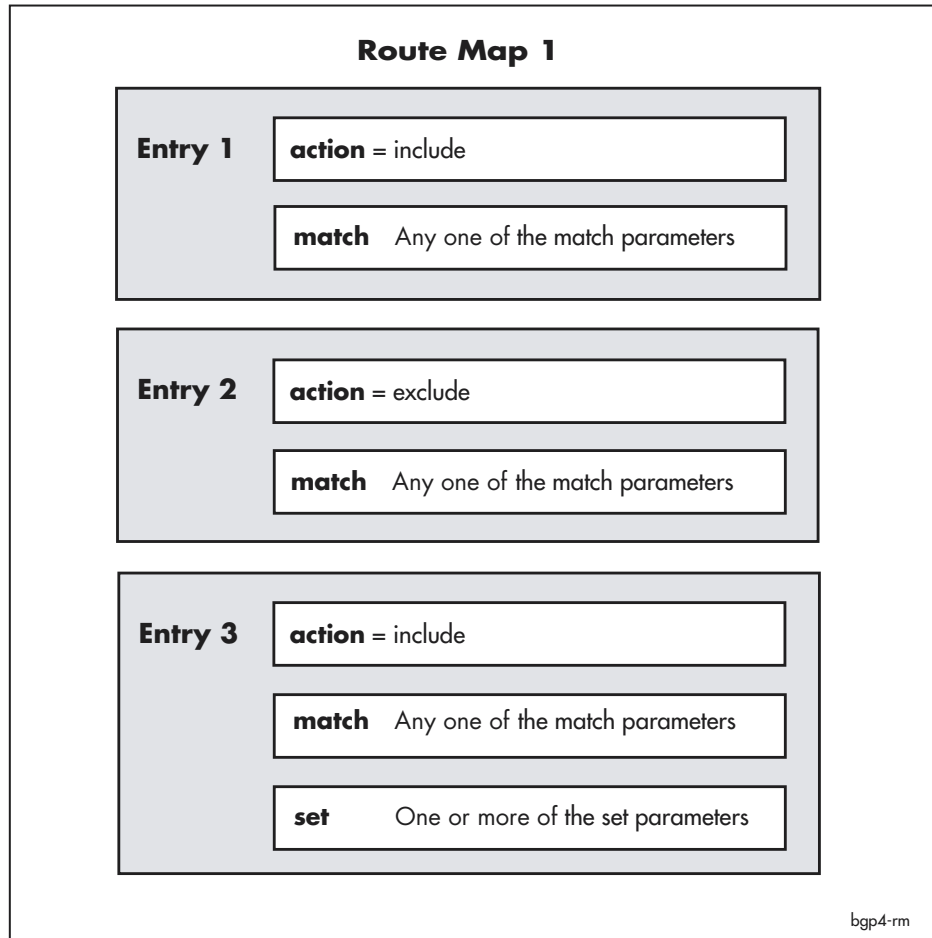
About Route Maps

Description Route maps are the most powerful route filtering option, and allow you to configure complex flexible filters. They achieve this by having several levels of structure:

- each route map consists of multiple entries
- each entry consists of an *action* (include or exclude) and at least one clause:
 - zero or one *match* clause, which determines which routes or BGP update messages match the entry. If you do not specify a match clause, every route or update message matches.
 - zero or more *set* clauses, which change certain features of matching routes or the attributes of matching BGP updates.

The following figure shows valid combinations of action and clause inside a route map.

Figure 31-1: Example structure of a route map



When to use route maps “Applying Filters” on page 31-22 describes in detail how to use route maps, but this section summarises the uses.

For BGP, you can use route maps when:

- copying routes from an update message to the RIB, by applying the route map as the **inroutemap** on a BGP peer
- determining which routes to import from other route sources, by applying the route map to the import entry
- when redistributing BGP routes as RIP routes, by applying the route map in the RIP redistribution configuration
- determining which routes to advertise, by applying the route map as the **outroutemap** on a BGP peer

When applied to a BGP peer, route maps can:

- accept or reject update messages on the basis of origin, community, AS path, next hop or Multi Exit Discriminator (MED)
- accept or reject particular routes, by comparing the update message’s routes with a prefix list, or by checking the route’s interface, metric, or nexthop.
- alter the attribute values in matching update messages.

For OSPF, you can use route maps:

- to filter routes from OSPF before adding them to the RIB
- when importing static routes into the OSPF LSA database

When applied to OSPF routes, route maps can:

- accept or reject particular routes on the basis of their interface, metric, route type, source, next hop or tag
- accept or reject particular routes, by comparing the update message's routes with a prefix list
- alter matching routes' metric, type and tag.

For RIP, you can use route maps when redistributing BGP routes as RIP routes, by applying the route map in the RIP redistribution configuration.

When applied to BGP routes for import into RIP, route maps can:

- accept or reject update messages on the basis of origin, community, AS path, next hop, interface, metric, or Multi Exit Discriminator (MED)
- accept or reject particular routes, by comparing the update message's routes with a prefix list
- alter matching routes' metric and tag

About IP Route Filters

Description Route filters are simple filters that examine a number of aspects of each route. When you apply filters to routing information that the switch receives, the filter determines whether each route is added to the RIB. When you apply filters to routing information that the switch transmits, the filter determines whether each route is advertised.

When to use IP route filters [“Applying Filters” on page 31-22](#) describes in detail how to use IP route filters, but this section summarises the uses.

The main uses of IP route filters are to select:

- RIP routes when adding routes to the RIB
- RIP routes when determining which routes to advertise
- OSPF, static or interface routes when determining which routes to redistribute from the RIB into RIP
- OSPF summary routes when determining which routes to redistribute from the RIB into the OSPF LSA database

You can also use IP route filters to select:

- OSPF routes to add to the RIB, but we recommend you use route maps instead
- BGP, RIP or static routes to redistribute from the RIB into the OSPF LSA database, but we recommend you use route maps instead

IP route filters affect the interaction between the routing module and the RIB, but IP route filters do not filter receipt of routing protocol messages by the routing module and do not directly filter messages sent from the routing protocol. Messages sent from the routing protocol are affected if and only if

they are derived from the RIB, which is true in most situations, including RIP, OSPF-ext messages, and OSPF summary Link State Advertisements (LSAs). Note that the design of OSPF prevents route filters from filtering some types of OSPF LSAs (see [“Limitations of route filtering on OSPF” on page 31-31](#)).

IP route filters do not filter BGP-derived routes, except when determining whether to add BGP routes to the OSPF LSA database. This means you cannot use IP route filters to select the routes that BGP receives, copies to the RIB, or advertises. For an overview of the filter types to use with BGP, see [“Border Gateway Protocol \(BGP-4\)” on page 31-29](#).

About IP Filters

Description	An IP filter filters routes when the type parameter is set to routing . It matches on the source and mask of the route, and specifies whether matching routes are included or excluded.
When to use IP filters	<p>IP filters of type routing have been superseded by other filtering mechanisms. They are still supported to provide backwards compatibility, and can be used in the following situations:</p> <ul style="list-style-type: none">■ when you want to filter routes that the switch imports from BGP into OSPF, but we recommend you use a route map instead■ as a BGP prefix filter (either an infilter or an outfilter), but we recommend you use a prefix list instead.

Creating Filters

This section describes the commands, options and procedures for creating each of the different types of filter. It contains the following subsections:

- [Creating Prefix Lists](#)
- [Creating AS Path Lists for BGP Routes](#)
- [Creating Route Maps for BGP Routes](#)
- [Creating Route Maps for OSPF Routes](#)
- [Creating Route Maps for Redistributing BGP Routes into RIP](#)
- [Creating IP Route Filters](#)
- [Creating IP Filters](#) for filtering routes (now superseded by other filter types)

Creating Prefix Lists

To create a prefix list and add entries to it, use the command:

```
add ip prefixlist=name entry=1..65535  
[action={match|nomatch}] [masklength=range] [prefix=ipadd]
```

The **masklength** parameter specifies the range of prefix mask lengths matched by this entry in the prefix list. The *range* is either a single CIDR mask from 0 to 32, or two masks separated by a hyphen. These options are valid for setting the mask length:

- As a mask length range (**masklength=*a-b***).
For a route to match against this entry, its prefix mask length must be between *a* and *b* inclusive. *a* must be less than *b*.
- As a single mask length (**masklength=*a***).
For a route to match against this entry, its prefix mask length must be exactly *a*.
- As an implicit mask length, by not specifying **masklength** (for example, **prefix=192.168.0.0**).
For a route to match against this entry, its prefix mask length must correspond exactly to the mask for the class of the given address—in this example, 24.

Creating AS Path Lists for BGP Routes

To create an AS path list and add entries to it, use one of the commands:

```
add ip aspathlist=1..99 [entry=1..4294967295]  
include=aspath-reg-exp  
  
add ip aspathlist=1..99 [entry=1..4294967295]  
exclude=aspath-reg-exp
```

Each entry uses a regular expression, *aspath-reg-exp*, to both specify the AS numbers that the entry matches, and to establish whether matching AS numbers are included or excluded.

The following table shows regular expression syntax and examples:

Table 31-1: Syntax for AS path regular expressions

Token	Description	Examples	Meaning of example
<AS number>	Matches that identical AS number.	123	Matches any AS path attribute that contains AS 123 (but not 1234, 12345, or 5123).
^	Matches the start of the AS path attribute.	^123	Matches AS path attributes that have AS 123 as the first AS.
\$	Matches the end of the AS path attribute.	^\$	Matches an empty AS path attribute.
		^123\$	Matches an AS path attribute with a single AS number, 123.
<space>	Separates AS numbers in a regular expression.	"123 456"	Matches AS path attributes that contain ASs 123 and 456, in that order, with no other AS numbers between them.
" "	Surrounds regular expressions that contain spaces.		
.	Matches any AS number.	.*	Matches all AS path attributes.
*	Matches zero or more repetitions of the preceding token in the AS path list being filtered.	"123 .* 456"	Matches AS path attributes that contain ASs 123 and 456, in that order, with any number of other AS numbers between them.
+	Matches one or more repetitions of the preceding token in the AS path list being filtered.	"123 .+ 456"	Matches AS path attributes that contain ASs 123 and 456, in that order, with at least one other AS number between them.

You can apply AS path lists directly to BGP peers, or use them in route maps (see ["Matching on AS path list" on page 31-11](#)).

Creating Route Maps for BGP Routes

A route map consists of multiple entries, which are in effect individual filters. Each entry specifies both what it matches on, in a *match* clause, and what is done to matching traffic, in the entry's *action* and any *set* clauses it has.

Most set clauses modify the BGP attributes of matching update messages. If you want to change the attributes of all candidate routes, configure an entry with no match clause. Such an entry matches all update messages.

When a BGP process passes an update message through a route map:

1. It checks the entries in order, starting with the lowest numbered entry, until it finds a match.
2. It then takes the action specified by that entry's action parameter. If the action is **exclude**, it filters out that update or prefix. If the action is **include**, it filters in that update or prefix.
3. If the action is **include**, it modifies attributes as specified by the entry's set clauses if there are any.
4. It then stops processing that update message; it does not check the remaining entries in the route map.

Every route map ends with an implicit entry that matches all routes with an action of **include**. This ensures that if no entries in a route map generate a match, the update message or route is included without modification.

The rest of this section describes:

- [How to create a route map](#)
- [How to configure an entry with a match clause](#)
- [How to configure an entry with a set clause](#)

How to create a route map

You do not have to create a route map as a separate step—adding the first entry automatically creates it.

How to configure an entry with a match clause

The match clause for a route map entry determines which update messages or prefixes match the entry. Each entry can only match on one characteristic. Available characteristics you can use with BGP are:

- [AS path list](#)
- [community list](#)
- [interface](#)
- [Multi Exit Discriminator \(MED\)](#)
- [metric](#)
- [next_hop attribute](#)
- [origin attribute](#)
- [prefix list](#)
- [tag](#)

Matching on AS path list

An entry that matches on **aspath** lets you select or discard routes that have taken a particular route or routes through the network.

To do this, first create an AS path list and add entries to it by using one of the commands:

```
add ip aspathlist=1..99 [entry=1..4294967295]
    include=aspath-reg-exp

add ip aspathlist=1..99 [entry=1..4294967295]
    exclude=aspath-reg-exp
```

See [“Creating AS Path Lists for BGP Routes” on page 31-9](#) for more information about creating path lists, valid syntax for the regular expression *aspath-reg-exp*, and syntax examples.

Then use the AS path list in the match clause of a route map by using the command:

```
add ip routemap=routemap entry=1..4294967295
    [action={include|exclude}] match aspath=1..99
```

When the switch uses this route map to examine an update message, the switch goes through the entries in the AS path list. The update matches if an entry in the AS path list matches the AS path in the update message, **and** that AS path list entry is an **include** entry.

If the update message matches, the switch carries out the action of the route map; one of:

- exclude the update message
- include the update message without modification
- include the update message and modify its attributes

Note that the action (include/exclude) of the AS path list and the action of the route map entry are separate. The following table shows the effect of each combination.

AS path list entry	Route map entry	Action when route map applied
include	include	An update message with that AS_path matches, and is processed
include	exclude	An update message with that AS_path matches, and is discarded
exclude	include	An update message with that AS_path does not match. The switch continues checking to see if the update message matches other entries in the route map.
exclude	exclude	An update message with that AS_path does not match. The switch continues checking to see if the update message matches other entries in the route map.

In this context, the parameters **include** and **exclude** in the AS path list do not indicate whether the matching update message is allowed or dropped; they simply indicate whether the update matches or does not match the path list. This is different to the behaviour when you use the AS path list itself as a filter, as described in [“Applying Filters” on page 31-22](#).

Example comparing AS path filter and route map

Compare this configuration, which uses an AS path list in a path filter:

```
add ip aspathlist=2 entry=1 exclude="^$"
add ip aspathlist=2 entry=2 include="15557"
set bgp peer=192.168.200.201 outpathfilter=2
```

with this configuration, which uses a route map and matches on AS path list:

```
add ip aspathlist=2 entry=1 include="^$"
add ip aspathlist=2 entry=2 exclude="15557"
add ip routemap=outdef3 entry=1 action=exclude match
    aspathlist=2
set bgp peer=192.168.200.201 outroutemap=outdef3
```

With both these configurations, the switch drops update messages with empty AS paths, and advertises update messages with an AS path containing 15557. For the route map to achieve this (the second configuration):

- The AS path list has to **include** empty paths, so that the empty path matches the path list, and therefore is included into the route map’s action of dropping packets that match the path list.

- The AS path list has to **exclude** updates whose AS path includes 15557. This excludes those updates from the route map's action of dropping packets that match the path list, so they are not dropped.

Matching on community list

An entry that matches on **communitylist** lets you select or discard routes that belong to a particular community.

To do this, first create a community list and add entries to it by using one of the commands:

```
add ip communitylist=1..99 [entry=1..4294967295]
    include={internet|noexport|noadvertise|
    noexportsubconfed|aa:xx}[,...]

add ip communitylist=1..99 [entry=1..4294967295]
    exclude={internet|noexport|noadvertise|
    noexportsubconfed|aa:xx}[,...]
```

Then use the community list in the match clause of a route map by using the command:

```
add ip routemap=routemap entry=1..4294967295
    [action={include|exclude}] match community=1..99
    [exact={no|yes}]
```

Note that the action (include/exclude) of the community list and of the route map entry are separate. This leads to the same behaviour as the distinction between the AS path list include/exclude parameters and the route map entry action. For a discussion of the distinction between these two include/exclude actions, see the table in [“Matching on AS path list” on page 31-11](#).

If you specify **exact=yes**, an update message only matches the route map entry if its community attribute contains all the communities specified in the community list, and no other communities. If you specify **exact=no**, which is the default, then the set of communities in the attribute list of the update message must contain all the communities in the specified community list, but can also contain other communities.

Matching on interface

An entry that matches on interface lets you select or discard routes that have a particular interface. To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
    [action={include|exclude}] match interface=interface
```

Matching on MED

An entry that matches on **med** lets you select or discard routes with a particular Multi Exit Discriminator metric. BGP can use the MED to determine the best route to a destination. To match on MED, use the command:

```
add ip routemap=routemap entry=1..4294967295
    [action={include|exclude}] match med=0..4294967295
```

Matching on metric

An entry that matches on **metric** lets you select or discard all routes with that metric or a metric in that range. To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
    [action={include|exclude}] match
    metric=0..4294967295[-0..4294967295]
```

Matching on next hop

An entry that matches on **nexthop** lets you select or discard routes that traverse a particular node. To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
    [action={include|exclude}] match nexthop=ipadd
```

Matching on origin An entry that matches on **origin** lets you select or discard routes depending on how BGP learned them: internally, externally, or from another means (such as statically-configured routes). To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match
origin={egp|igp|incomplete}
```

Matching on prefix list An entry that matches on **prefixlist** lets you select or discard routes to a list of destinations.

To do this, first create the prefix list and add entries to it by using the command:

```
add ip prefixlist=name entry=1..65535
[action={match|nomatch}] [masklength=range] [prefix=ipadd]
```

See [“Creating Prefix Lists” on page 31-9](#) for more information.

Then use the prefix list in the match clause of a route map by using the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match prefixlist=name
```

All the match options described previously—AS path, community, next hop and origin—match on the **attributes** in an update message. Prefix list does not; it matches prefixes.

Note that the action of the prefix list and of the route map entry are separate. The following table shows the effect of each combination.

Prefix list entry	Route map entry	Action when route map applied
match	include	An update message that contains the prefix matches the route map entry. The prefix is processed.
match	exclude	An update message that contains the prefix matches the route map entry. The prefix is removed from the update message. Other prefixes in the update are not removed.
nomatch	include	An update message that contains the prefix does not match the route map entry. The switch continues checking to see if the update message matches other entries in the route map.
nomatch	exclude	An update message that contains the prefix does not match the route map entry. The switch continues checking to see if the update message matches other entries in the route map.

In this context, the parameters **match** and **nomatch** in the prefix list do not indicate whether the prefix is allowed or dropped; they simply indicate whether the prefix matches or does not match the prefix list.

Matching on tag

An entry that matches on **tag** lets you select or discard certain routes for importing into BGP. To do this, first *tag* the routes of interest with an identification number, using a route map or (for static routes) one of the commands:

```
add ip route=ipadd interface=interface nexthop=ipadd
tag=1..65535 [other-options]
```

```
set ip route=ipadd interface=interface mask=mask
nexthop=ipadd tag=1..65535 [other-options]
```

To see which number a route is tagged with, use the command:

```
show ip route
```

Then use the tags in a route map by using the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match tag=1..65535
```

How to configure an entry with a set clause

Once you have determined what update messages or prefixes a route map entry matches, you can configure set clauses to change the attributes of matching items.

To create a set clause for an entry, use one of the commands shown in the following table.

Route map set clauses for BGP

Command	Result
add ip routemap = <i>routemap</i> entry=1..4294967295 set aspath={1..65534[,...]}	Adds up to 10 AS numbers at the beginning of the AS path attribute.
add ip routemap = <i>routemap</i> entry=1..4294967295 set bgpdampid=1..100	Sets the BGP route flap damping ID that is given to matching routes (see “Damping routes on specific peers” on page 30-27 of Chapter 30, Border Gateway Protocol version 4 (BGP-4)).
add ip routemap = <i>routemap</i> entry=1..4294967295 set community={noexport noadvertise noexportsubconfed aa:xx[,...]} [add={no yes}]	Either: replaces the community attribute with a list of up to 10 community values, if add=no (the default), or adds up to 10 community values to the community attribute, if add=yes
add ip routemap = <i>routemap</i> entry=1..4294967295 set localpref=0..4294967295	Replaces the existing local_preference attribute, or sets it if it was not already set.
add ip routemap = <i>routemap</i> entry=1..4294967295 set med={0..4294967295 remove}	Replaces the existing MED attribute, or sets it if it was not already set, or if you specify med=remove , deletes the MED attribute.

**Route map set clauses
for BGP**

Command	Result
add ip routemap = <i>routemap</i> entry=1..4294967295 set origin={igp egp incomplete}	Replaces the existing origin attribute, or sets it if it was not already set.
add ip routemap = <i>routemap</i> entry=1..4294967295 set tag=1..65535	Tags the matching routes with an ID number. If you then redistribute the routes into OSPF or RIP, the tag allows you to identify the redistributed routes.

A prefix list can match a subset of prefixes in an update message. You can use this to change the attributes of some of the prefixes in an outgoing update, without having to change the attributes of all the prefixes. However, an update message contains just one set of attributes, which must apply to all the prefixes in the update. Therefore, the switch splits the original update into two updates:

- one that contains the original attribute values and the prefixes that were not included by the route map entry, and
- one that contains the new attribute values and the prefixes that were included by the route map entry

Creating Route Maps for OSPF Routes

A route map consists of multiple entries, which are in effect individual filters. Each entry specifies both what it matches on, in a *match* clause, and what is done to matching traffic, in the entry's *action* and any *set* clauses it has.

When the switch applies a route map to OSPF routes:

1. It checks the entries in order, starting with the lowest numbered entry, until it finds a match.
2. It then takes the action specified by that entry's action parameter. If the action is **exclude**, it filters out that route. If the action is **include**, it filters in that route.
3. If the action is **include**, it modifies the route characteristics as specified by the entry's set clauses if there are any.
4. It then stops processing that route; it does not check the remaining entries in the route map.

Every route map ends with an implicit entry that matches all routes with an action of **include**. This ensures that if no entries in a route map generate a match, the route is included without modification.

The rest of this section describes:

- [How to create a route map](#)
- [How to configure an entry with a match clause](#)
- [How to configure an entry with a set clause](#)

How to create a route map

You do not have to create a route map as a separate step—adding the first entry automatically creates it.

How to configure an entry with a match clause

The match clause for a route map entry determines which routes match the entry. Each entry can only match on one characteristic. Available characteristics you can use with OSPF routes are:

- [interface](#)
- [metric](#)
- [next hop](#)
- [prefix list](#)
- [route source](#)
- [route type](#)
- [tag](#)

Matching on interface

An entry that matches on interface lets you select or discard routes that have a particular interface. To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match interface=interface
```

Matching on metric

An entry that matches on **metric** lets you select or discard all routes with that metric or a metric in that range. To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match
metric=0..4294967295[-0..4294967295]
```

Matching on next hop

An entry that matches on **nexthop** lets you select or discard routes that traverse a particular node. To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match nexthop=ipadd
```

Matching on prefix list

An entry that matches on **prefixlist** lets you select or discard routes to a list of destinations.

To do this, first create the prefix list and add entries to it by using the command:

```
add ip prefixlist=name entry=1..65535
[action={match|nomatch}] [masklength=range] [prefix=ipadd]
```

See [“Creating Prefix Lists” on page 31-9](#) for more information.

Then use the prefix list in the match clause of a route map by using the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match prefixlist=name
```

Note that the action of the prefix list and of the route map entry are separate. The following table shows the effect of each combination.

Table 31-2: The effect of actions in prefix list and route map entries

Prefix list entry	Route map entry	Action when route map applied
match	include	A route to that prefix matches the route map entry. The switch adds the route to its RIB.
match	exclude	A route to that prefix matches the route map entry. The switch excludes the route from its RIB.
nomatch	include	A route to that prefix does not match the route map entry. The switch continues checking to see if the route matches other entries in the route map.
nomatch	exclude	A route to that prefix does not match the route map entry. The switch continues checking to see if the route matches other entries in the route map.

In this context, the parameters **match** and **nomatch** in the prefix list do not indicate whether a route to that prefix is allowed or dropped; they simply indicate whether the prefix matches or does not match the prefix list.

Matching on route source

An entry that matches on **routesource** lets you select or discard routes depending on the router ID of the router that they were learnt from.

To do this, first create a prefix list for the router IDs, by using the command:

```
add ip prefixlist=name entry=1..65535
[action={match|nomatch}] masklength=32 [prefix=ipadd]
```

See “[Creating Prefix Lists](#)” on page 31-9 for more information. Note that the mask for a router ID must be 255.255.255.255, so the mask length must be 32.

Then use the prefix list in the match clause of a route map by using the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match
routesource=prefixlist-name
```

Note that the action of the prefix list and of the route map entry are separate. [Table 31-2](#) shows the effect of each combination.

Matching on route type

An entry that matches on **routetype** lets you select or discard particular types of routes: intra-area, inter-area, External Type 1, External Type 2, or other routes. To do this, use the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match
routetype={intra|inter|type1|type2|other}
```

See “[Routing with OSPF](#)” on page 29-9 of [Chapter 29, Open Shortest Path First \(OSPF\)](#) for more information about these route types.

Matching on tag An entry that matches on **tag** lets you select or discard certain routes for importing into OSPF.

To do this, first *tag* the routes of interest with an identification number, using a route map or (for static routes) one of the commands:

```
add ip route=ipadd interface=interface nexthop=ipadd
tag=1..65535 [other-options]

set ip route=ipadd interface=interface mask=mask
nexthop=ipadd tag=1..65535 [other-options]
```

To see which number a route is tagged with, use the command:

```
show ip route
```

Then use the tags in a route map by using the command:

```
add ip routemap=routemap entry=1..4294967295
[action={include|exclude}] match tag=1..65535
```

How to configure an entry with a set clause

Once you have determined what routes a route map entry matches, you can configure set clauses to change the characteristics of matching items.

To create a set clause for an entry, use one of the commands shown in the following table.

Route map set clauses for OSPF

Command	Result
<code>add ip routemap=routemap entry=1..4294967295</code> <code>set metric=0..4294967295</code>	Sets the OSPF metric of matching routes. Routes with a lower metric are preferred.
<code>add ip routemap=routemap entry=1..4294967295</code> <code>set type={1 2}</code>	Sets the route type of matching routes to External Type 1 or External Type 2. See “Routing with OSPF” on page 29-9 of Chapter 29, Open Shortest Path First (OSPF) for more information about route types.
<code>add ip routemap=routemap entry=1..4294967295</code> <code>set tag=1..65535</code>	Tags the matching routes with an ID number.

Creating Route Maps for Redistributing BGP Routes into RIP

A route map consists of multiple entries, which are in effect individual filters. Each entry specifies both what it matches on, in a *match* clause, and what is done to matching traffic, in the entry's *action* and any *set* clauses it has.

The set clauses modify the characteristics of matching routes. If you want to change the characteristics of all candidate routes, configure an entry with no match clause. Such an entry matches all routes.

When RIP passes a BGP-sourced route through a route map:

1. It checks the entries in order, starting with the lowest numbered entry, until it finds a match.

2. It then takes the action specified by that entry's action parameter. If the action is **exclude**, it filters out that route. If the action is **include**, it filters in that route.
3. If the action is **include**, it modifies characteristics as specified by the entry's set clauses if there are any.
4. It then stops processing that route; it does not check the remaining entries in the route map.

Every route map ends with an implicit entry that matches all routes, with an action of **include**. This ensures that if no entries in a route map generate a match, the route is included without modification.

The rest of this section describes:

- [How to create a route map](#)
- [How to configure an entry with a match clause](#)
- [How to configure an entry with a set clause](#)

How to create a route map

You do not have to create a route map as a separate step—adding the first entry automatically creates it.

How to configure a match clause

The match clause for a route map entry determines which routes match the entry. A route map for use when importing BGP routes into RIP can match on any of the match clauses that apply to BGP routes (see [“How to configure an entry with a match clause” on page 31-11](#)).

How to configure a set clause

Once you have determined what routes a route map entry matches, you can configure set clauses to change the attributes of matching items.

To create a set clause for an entry, use one of the commands shown in the following table.

Route map set clauses for RIP

Command	Result
add ip routemap = <i>routemap</i> entry=1..4294967295 set metric=0..4294967295	Sets the RIP metric of matching routes. Routes with a lower metric are preferred. Metrics higher than 16 are treated as 16.
add ip routemap = <i>routemap</i> entry=1..4294967295 set tag=1..65535	Tags the matching routes with an ID number. This lets you later identify the routes that came from BGP.

Creating IP Route Filters

To create a route filter, use the command:

```
add ip route filter [=filter-id] ip=ipadd mask=ipadd
  action={include|exclude} [direction={receive|send|both}]
  [interface=interface] [nexthop=ipadd] [policy=0..7]
  [protocol={any|ospf|rip}]
```

The **protocol** parameter specifies the routing protocol to which the filter applies. When **direction** is **receive**, then **protocol** specifies the routing protocol that receives the route information. If **direction** is **send**, **protocol** specifies the routing protocol that advertises the routes.

When the routing protocol receives or transmits a route, it searches the list of route filters for a match to the route. The **ip**, **mask**, **interface**, **nexthop**, and **policy** parameters define a pattern to match against. The **action** parameter determines whether routes matching the pattern are used or discarded.

The switch checks each route against each filter, starting with the lowest-numbered filter, until it finds a match. Then it applies that filter and stops processing the list of filters.

When you create a list of filters—even a list of only one filter—the switch ends the list with an implicit filter to *exclude* all routes. So if you want the switch to include all routes that do not match your filters, end your filter list with a filter that matches all routes and includes them, such as:

```
add ip route filter=100 ip=*. *.*.*.* mask=*. *.*.*.*
  action=include
```

Creating IP Filters

Tip Use of IP filters for route filtering has been superseded by route maps and prefix lists. However, it is still available for backwards compatibility.

To create an IP filter that will filter routes, use the command:

```
add ip filter=0..999 type=routing action={include|exclude}
  source=ipadd [smask=ipadd] [entry=1..255]
```

The **source** parameter is the network IP address of the subnet to be filtered.

The **smask** parameter determines how many bits of the prefix are significant. When the switch checks routes against the filter, it only checks the significant bits.

By default, new entries are added at the end of the filter. If you want the entry to be checked before some of the other entries, give it a lower entry number. This pushes existing entries with the same or higher number further down the list.

Applying Filters

This section describes how to apply the filters you have created, to achieve the following results:

- [Applying Filters When Writing to the RIB](#)
- [Applying Filters When Redistributing from the RIB](#)
- [Applying Filters Before Advertising Routes.](#)

For BGP, you can apply several types of filter to each peer. If you do this, the switch first applies the AS path filter, then the prefix filter, then the route map. Note that the switch stops checking after the first filter entry that excludes the update or prefix, so an update or prefix is only included if all the applied filters result in it being included.

Applying Filters When Writing to the RIB

When the switch receives information about a route, it normally adds that route to its RIB. This makes the route available for the switch to use. You can use route filters to stop the switch from adding certain routes—or routes with certain characteristics—into the RIB. This gives you control over the routes packets take when they leave the switch.

Filtering BGP routes when writing to the RIB

Filters act on the BGP update messages that the switch receives, or on the routes within update messages. You can use the following types of filter:

- [prefix lists](#)
- [AS path lists](#)
- [route maps](#)

Applying prefix lists

Prefix filtering rejects some of the routes from an update message, without rejecting the whole update. This enables you to configure the switch to accept only routes for particular networks from a particular peer.

To use a prefix list as a prefix filter, use one of the commands:

```
add bgp peer=ipadd remoteas=asn [infilter=prefixlist-name]
    [other-options]

set bgp peer=ipadd [infilter=prefixlist-name] [other-options]

add bgp peertemplate=1..30 [infilter=prefixlist-name]
    [other-options]

set bgp peertemplate=1..30 [infilter=prefixlist-name]
    [other-options]
```

The **infilter** parameter uses the prefix list to filter update messages that the switch receives from the peer. If a prefix matches a prefix in the prefix list, BGP rejects that route. Otherwise, it accepts the route.

The switch checks every route in the update message against every entry in the filter, starting with the entry with the lowest entry number, until it finds a match or gets to the end of the filter.

You can also use a prefix list in a route map and apply the route map.

Applying AS path lists

To apply an AS path list directly as a filter on a BGP peer, use the command:

```
add bgp peer=ipadd remoteas=asn [inpathfilter=1..99]
[outpathfilter=1..99] [other-options]
```

The **inpathfilter** parameter applies the AS path list as a filter on update messages that the switch receives from the peer. The switch only accepts update messages if they match an AS path list entry that has the action **include**. If an update message matches an entry with the action **exclude**, the switch rejects the update. If an update message does not match any entry in the AS path list, the switch rejects the update. This is because each non-empty AS path list ends with an implicit entry that matches any AS path list and has the action **exclude**.

You can also use an AS path list in a route map and apply the route map.

Applying route maps

To use a route map to filter or modify update messages that it receives from a peer, use one of the commands:

```
add bgp peer=ipadd remoteas=asn inroutemap=routemap
[other-options]

set bgp peer=ipadd inroutemap=routemap [other-options]
```

The switch checks every route in the update message against every entry in the filter, starting with the entry with the lowest entry number, until it finds a match or gets to the end of the filter.

Filtering OSPF routes when writing to the RIB

To filter OSPF routes before adding them to the RIB, first create a route map that matches on the appropriate route characteristics. Then use the route map in the command:

```
set ospf inroutemap=routemap
```

Plan your filters carefully. If a filter excludes a matching route from the RIB, OSPF does not advertise a summary LSA for that route because summary LSA messages are derived from the filtered RIB. This means that incorrect filters can prevent Area Border Routers from advertising routes to other areas.

Filtering RIP routes when writing to the RIB

To filter RIP routes before adding them to the RIB, simply create a filter or series of filters, using the command:

```
add ip route filter[=filter-id] ip=ipadd mask=ipadd
action={include|exclude} protocol=rip direction=receive
[other-options]
```

The switch automatically applies the filter when importing RIP routes, because **protocol=rip**.

The immediate effect of a route filter with **direction** set to **receive** and **action** set to **exclude** is that route advertisements received matching the filter do not result in a new entry in the local RIB. However, routes already in the RIB are not deleted even when they match the route filter. Therefore, if you dynamically add a route filter at the manager prompt, you may also need to manually delete unwanted routes from the RIB.

Filtering invalid when writing static and interface routes to the RIB

You cannot filter in a way that excludes statically-configured or interface routes from the RIB.

Applying Filters When Redistributing from the RIB

The switch is able to import routes from the RIB into BGP, OSPF or RIP, even if it learnt them from a different routing protocol or source. For example, you can add non-BGP routes to BGP, such as static routes and routes learned by OSPF or RIP. BGP can then advertise these routes.

When you import routes from some route sources, you can also filter, to block certain routes. Most of the filtering options use route maps, so you can also give the imported routes certain characteristics, such as changing their metric.

Note that the route map must match on characteristics that are relevant for the routes you are importing. For example, if you are importing routes into BGP, you cannot match on AS path or community. These attributes are not relevant to non-BGP routes.

Filtering when copying routes to BGP

BGP can import routes from OSPF and RIP, as well as statically-configured and interface routes. You can use route maps to filter routes from any of these sources. The switch uses the route map to filter routes and/or set attributes when it imports the routes into BGP. The following table shows how to filter routes from each source.

From	To filter
OSPF	<ol style="list-style-type: none"> For finest control, tag each OSPF route you want to import into BGP, by creating a route map and applying it to OSPF routes. Use the commands: <pre>add ip routemap=routemap entry=1..4294967295 action={include exclude} match [match-options] add ip routemap=routemap set tag=1..65535 set ospf inroutemap=routemap</pre> Create another route map to use when importing into BGP. Match on nexthop, prefixlist or tag. Apply the route map, using the command: <pre>add bgp import=ospf routemap=routemap</pre>
RIP	<ol style="list-style-type: none"> Create a route map, matching on nexthop or prefixlist Apply the route map, using the command: <pre>add bgp import=rip routemap=routemap</pre>
Interface routes	<ol style="list-style-type: none"> Create a route map, matching on nexthop or prefixlist Apply the route map, using the command: <pre>add bgp import=interface routemap=routemap</pre>

From	To filter
Static routes	<ol style="list-style-type: none"> 1. For finest control, tag each route you want to include, using the command: <code>set ip route=ipadd interface=interface mask=mask nexthop=ipadd tag=1..65535 [other-options]</code> 2. Create a route map, matching on nexthop, prefixlist or tag 3. Apply the route map, using the command: <code>add bgp import=static routemap=routemap</code>

Filtering when copying routes to OSPF

OSPF can import the following:

- BGP routes, with or without filtering
- RIP routes, with or without filtering
- non-interface routes, with or without filtering
- statically-configured routes, with or without filtering

The following table shows how to filter routes.

From	How to filter
BGP	<ol style="list-style-type: none"> 1. Create a route map, matching on metric, interface, nexthop, prefixlist, aspath, community, med, or origin. Use the command: <code>add ip routemap=routemap entry=1..4294967295 action={include exclude} match [match-options]</code> 2. If you want to tell later which routes came from BGP, tag the routes you want to import, by adding a set clause to the route map. Use the command: <code>add ip routemap=routemap set tag=1..65535</code> 3. Apply the route map, using the command: <code>add ospf redistribute protocol=bgp routemap=routemap [other options]</code>
RIP	<ol style="list-style-type: none"> 1. Create a route map, matching on metric, interface, nexthop, or prefixlist. Use the command: <code>add ip routemap=routemap entry=1..4294967295 action={include exclude} match [match-options]</code> 2. If you want to tell later which routes came from RIP, tag the routes you want to import, by adding a set clause to the route map. Use the command: <code>add ip routemap=routemap set tag=1..65535</code> 3. Apply the route map, using the command: <code>add ospf redistribute protocol=rip routemap=routemap [other options]</code>
non-OSPF interface	<ol style="list-style-type: none"> 1. Create a route map, matching on metric, interface, nexthop, or prefixlist. Use the command: <code>add ip routemap=routemap entry=1..4294967295 action={include exclude} match [match-options]</code> 2. If you want to tell later which routes started as non-OSPF interface routes, tag the routes you want to import, by adding a set clause to the route map. Use the command: <code>add ip routemap=routemap set tag=1..65535</code> 3. Apply the route map, using the command: <code>add ospf redistribute protocol=interface routemap=routemap [other options]</code>

From	How to filter
Static routes	<ol style="list-style-type: none"> For finest control, tag each route you want to include (or each route you want exclude), using one of the commands: <code>add ip route=<i>ipadd</i> interface=<i>interface</i> nexthop=<i>ipadd</i> tag=1..65535 [other-options]</code> or <code>set ip route=<i>ipadd</i> interface=<i>interface</i> mask=<i>mask</i> nexthop=<i>ipadd</i> tag=1..65535 [other-options]</code> Create a route map, matching on tag, metric, interface, nexthop, or prefixlist. Apply the route map, using the command: <code>add ospf redistribute protocol=static routemap=<i>routemap</i> [other options]</code>

Filtering when copying routes to RIP

RIP can import static, BGP and OSPF routes. It also automatically imports interface routes. The following table shows how to filter routes.

From	How to filter
BGP	<ol style="list-style-type: none"> Create a route map, matching on prefix or on any BGP route attribute. Apply the route map, using the command: <code>add ip rip redistribute protocol=bgp routemap=<i>routemap</i></code>
OSPF	<ol style="list-style-type: none"> Turn on exporting of OSPF routes into RIP, by using the command: <code>set ospf rip=export [other-options]</code> Create IP route filters to determine which OSPF routes are copied into the LSA database, by using the command: <code>add ip route filter[=<i>filter-id</i>] ip=<i>ipadd</i> mask=<i>ipadd</i> action={include exclude} protocol=rip direction=send [other-options]</code> The switch automatically applies the filter when importing routes into RIP, because protocol=rip.
Static	<ol style="list-style-type: none"> By default, RIP imports and advertises static routes. If this has been turned off, turn it on for the required interfaces by using the command: <code>set ip rip interface=<i>interface</i> staticexport=yes [other-options]</code> Create IP route filters to determine which static routes are imported, by using the command: <code>add ip route filter[=<i>filter-id</i>] ip=<i>ipadd</i> mask=<i>ipadd</i> action={include exclude} protocol=rip direction=send [other-options]</code> The switch automatically applies the filter when importing routes into RIP, because protocol=rip.
Interface	<p>RIP automatically imports interface routes. Create IP route filters to determine which interface routes are imported, by using the command: <code>add ip route filter[=<i>filter-id</i>] ip=<i>ipadd</i> mask=<i>ipadd</i> action={include exclude} protocol=rip direction=send [other-options]</code> The switch automatically applies the filter when importing routes into RIP, because protocol=rip.</p>

Applying Filters Before Advertising Routes

Routing protocols send their neighbours or peers information about the routes in the switch's RIB. You can use route filters to stop the switch from advertising certain routes or routes with certain characteristics. This gives you control over the routes that packets take through your network and when leaving your network.

Filtering when using BGP to advertise routes

Filters act on all routes with a particular BGP attribute, or on particular routes. You can use the following types of filter:

- [prefix lists](#)
- [AS path lists](#)
- [route maps](#)

Applying prefix lists

Prefix filtering rejects some of the routes from an update message, without rejecting the whole update. This enables you to configure the switch to send only routes for particular networks to a particular peer.

To use a prefix list as a prefix filter, use one of the commands:

```
add bgp peer=ipadd remoteas=asn outfilter=prefixlist-name
[other-options]

set bgp peer=ipadd outfilter=prefixlist-name [other-options]

add bgp peertemplate=1..30 outfilter=prefixlist-name
[other-options]

set bgp peertemplate=1..30 outfilter=prefixlist-name
[other-options]
```

The **outfilter** parameter uses the prefix list to filter update messages that the switch sends to the peer. If a prefix matches a prefix in the prefix list, BGP removes that route from the update message. Otherwise, it leaves the route in the update message and therefore advertises it to the peer.

The switch checks every route in the update message against every entry in the filter, starting with the entry with the lowest entry number, until it finds a match or gets to the end of the filter.

For example, to create a peer relationship on the local switch, with a peer that has the IP address 192.168.1.1 and is part of AS 1, and prevent the local switch from advertising routes from the 10.0.0.0/8 network, use the commands:

```
add ip prefixlist=10_network entry=1 action=match
prefix=10.0.0.0/8

add bgp peer=192.168.1.1 remotas=1 outfilter=10_network
```

You can also use a prefix list in a route map and apply the route map, as described below.

Applying AS path lists

To apply an AS path list directly as a filter on a BGP peer, use one of the commands:

```
add bgp peer=ipadd remoteas=asn [inpathfilter=1..99]
[outpathfilter=1..99] [other-options]

set bgp peer=ipadd [inpathfilter=1..99] [outpathfilter=1..99]
[other-options]
```

The **outpathfilter** parameter applies the AS path list as a filter on update messages that the switch sends to the peer. The switch only sends update messages if the update's AS path attribute matches an entry that has the action **include**. If a route matches an entry with the action **exclude**, the switch does not advertise it to that peer. If an update message does not match any entry in the AS path list, the switch does not advertise it to that peer.

You can also use an AS path list in a route map and apply the route map.

Applying route maps

To use the route map to filter or modify update messages that it sends to a peer, use one of the commands:

```
add bgp peer=ipadd remoteas=asn outroutemap=routemap
[other-options]

set bgp peer=ipadd outroutemap=routemap [other-options]
```

The switch checks every route in the update message against every entry in the filter, starting with the entry with the lowest entry number, until it finds a match or gets to the end of the filter.

If your route map is intended to modify the community attribute of outgoing update messages, you also need to enable the switch to set the community attribute in messages to that peer. Use one of the commands:

```
add bgp peer=ipadd remoteas=asn outroutemap=routemap
sendcommunity=yes [other-options]

set bgp peer=ipadd outroutemap=routemap sendcommunity=yes
[other-options]
```

Filtering invalid when using OSPF to advertise routes

The design of the OSPF protocol does not allow you to filter LSAs before advertising them. This is because OSPF shares LSAs between all the routers in an area. The protocol assumes that all the routers in the area have shared all the advertisements among each other, and that all agree on the state of the complete link state database for the area. If some routers in the area are learning, but not advertising, that breaks the OSPF model.

Therefore, once a route is in the LSA database, you have no control over whether it is advertised.

Filtering when using RIP to advertise routes

To filter routes before advertising them with RIP, create a filter or series of filters, using the command:

```
add ip route filter[=filter-id] ip=ipadd mask=ipadd
action={include|exclude} protocol=rip direction=send
[other-options]
```

The switch automatically applies the filter when advertising routes to RIP neighbours, because **protocol=rip**.

No mechanism for advertising static and interface routes

Statically-configured and interface routes do not have mechanisms to advertise routes. Only the routing protocols (OSPF, BGP, and RIP) advertise routes. To advertise static and interface routes, with or without filtering, import the routes into the required routing protocol.

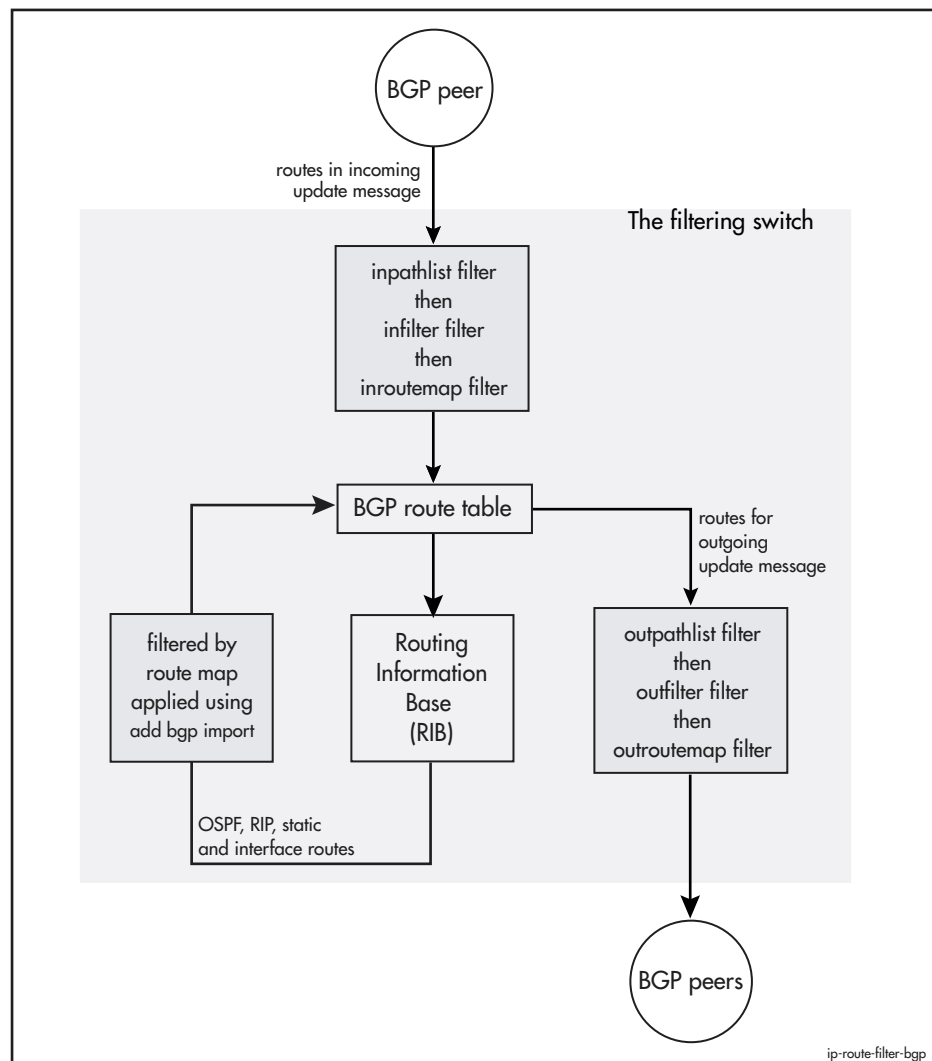
Overview of Filters for each Route Source

The sections above describe each type of filter. This section contains a series of diagrams that summarise the available filters for each route source:

- **Border Gateway Protocol (BGP-4)**
- **Open Shortest Path First (OSPF)**
- **Routing Information Protocol (RIP)**
- **Interface Routes**
- **Statically-Configured Routes**

Border Gateway Protocol (BGP-4)

When the switch runs BGP, it receives routing information from peer routers. It may also advertise routing information from BGP and other route sources to peer routers. You can filter routing information at the processing points shown in the following figure.



Processing points for route filtering when using BGP

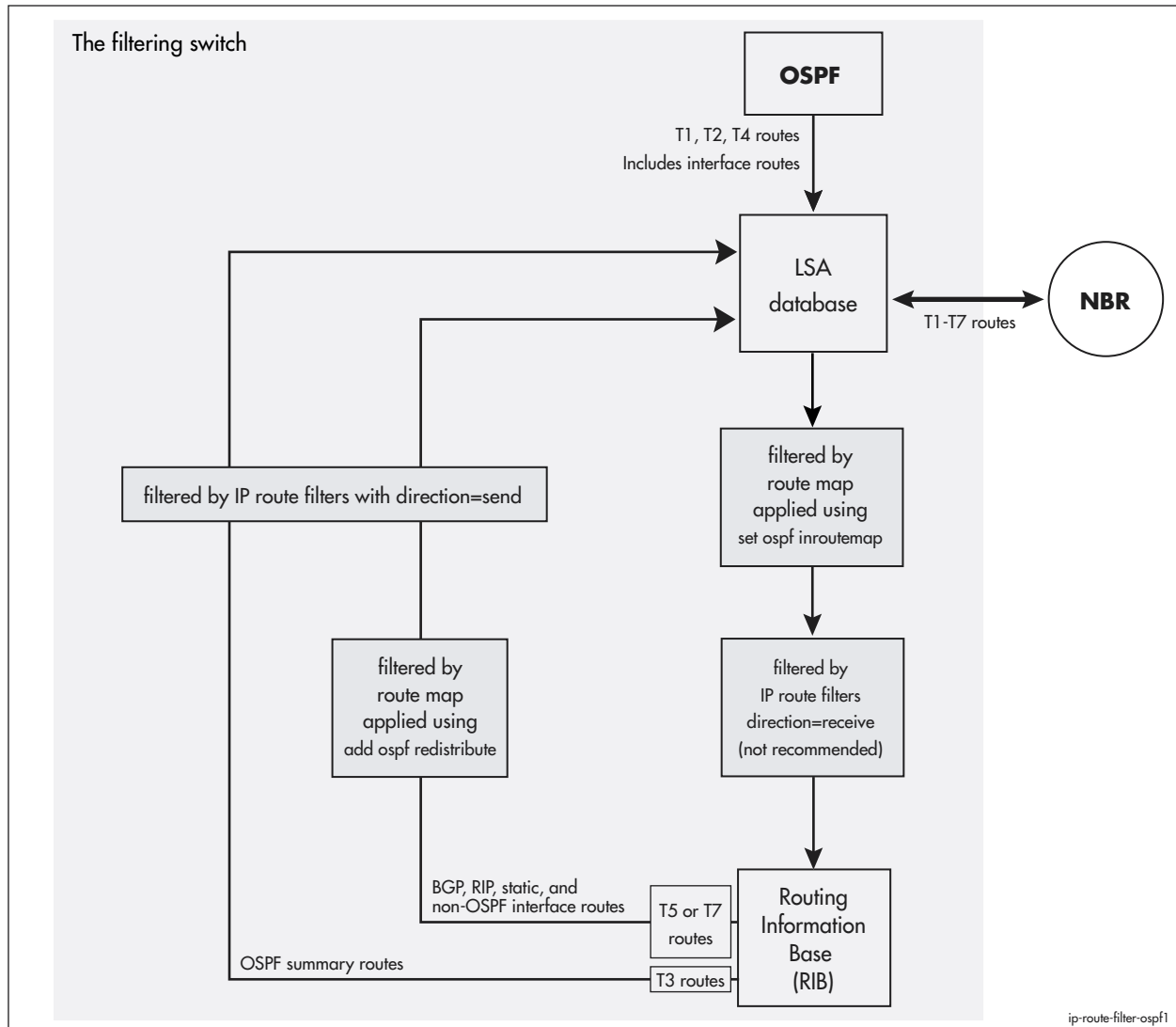
Open Shortest Path First (OSPF)

When the switch runs OSPF, it receives routing information from neighbouring routers and advertises routing information to neighbouring routers. This routing information is contained in Link State Advertisements (LSAs). OSPF also generates LSAs internally.

The following table describes types of LSA.

LSA Name	LSA describes	LSA is created
Type-1 Router-LSA	the state and cost of each of the switch's interfaces to the area	by OSPF, on every router in the area
Type-2 Network-LSA	all routers attached to the network	by OSPF, on the network's Designated Router
Type-3 Summary-LSA	inter-area destinations, when the destination is an IP network	from the RIB, by Area Border Routers
Type-4 Summary-LSA	inter-area destinations, when the destination is an Autonomous System (AS) boundary router	by OSPF, by Area Border Routers
Type-5 AS-external-LSA	a destination outside the AS	from the RIB, by AS boundary routers
Type-7 AS-external-LSA	a destination outside the AS. Used in not-so-stubby areas	from the RIB, by AS boundary routers

You can filter routing information at the processing points shown in the following figure. The figure also indicates the type of LSA at each processing point.



Processing points for route filtering when using OSPF

Limitations of route filtering on OSPF

As the previous diagram shows, the OSPF LSA database is a completely separate entity to the switch's RIB. The OSPF design does not allow you to filter the contents of the database before advertising routes to neighbouring routers. This is because OSPF shares LSAs between all the routers in an area. The protocol assumes that all the routers in the area have shared all the advertisements among each other, and that all agree on the state of the complete link state database for the area. If some routers in the area are learning, but not advertising, that breaks the OSPF model.

These limitations mean you can only filter to control the following:

- which routes learned by OSPF can be imported by the switch from the LSA database into the RIB

We recommend you use route maps to filter this (see [“Filtering OSPF routes when writing to the RIB”](#) on page 31-23)

- which static, BGP or RIP routes can be exported from the RIB into the LSA database

We recommend you use route maps to filter these (see [“Filtering when copying routes to OSPF” on page 31-25](#))

- which summary routes can be exported from the RIB into the LSA database for advertising as summary LSAs

You can use IP route filters to filter this (see [“Filtering when copying routes to OSPF” on page 31-25](#))

Another way to filter summary LSAs is to define a “do not advertise” OSPF range on an Area Border Router. This stops OSPF from advertising inter-area routes into another area. To do this, use the command:

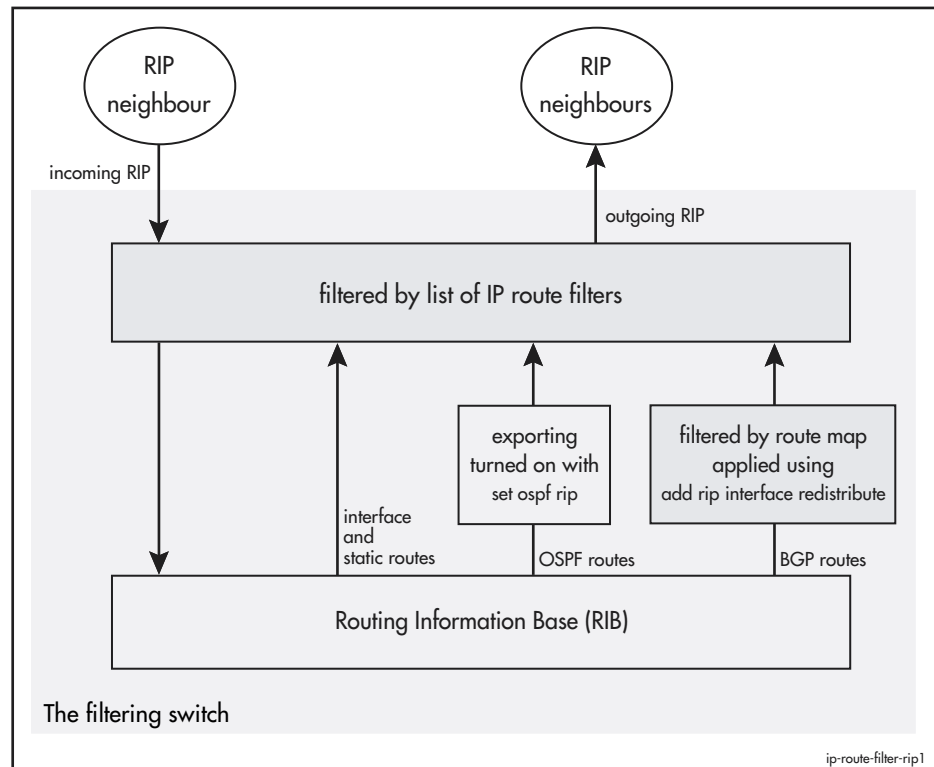
```
set ospf range=ipadd effect=donotadvertise [other-options]
```

Note that filtering **cannot** do the following:

- remove an entry from the LSA database once the entry has been added
- prevent the switch from advertising an entry to interfaces in the same area that the entry is relevant to
- prevent updates that OSPF learns from being put into the database
- change the properties of an entry in the database

Routing Information Protocol (RIP)

When the switch runs RIP, it receives routing information from neighbouring routers, and can advertise RIP, BGP, statically-configured and interface routes to neighbouring routers. You can filter routing information at the processing points shown in the following figure.



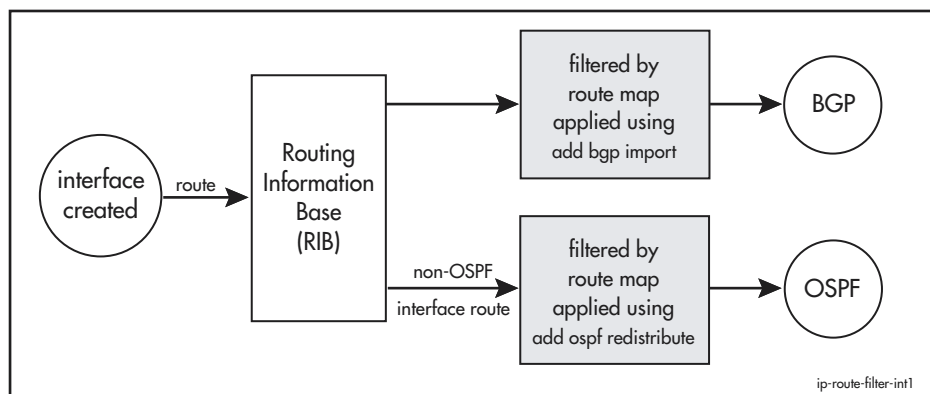
Interface Routes

When you create an interface on the switch, it automatically creates an interface route. This route tells the switch to send packets over that interface when the packets are addressed to the interface's subnet.

Filtering these routes before placing them in the RIB would be meaningless, so there is no mechanism to do so.

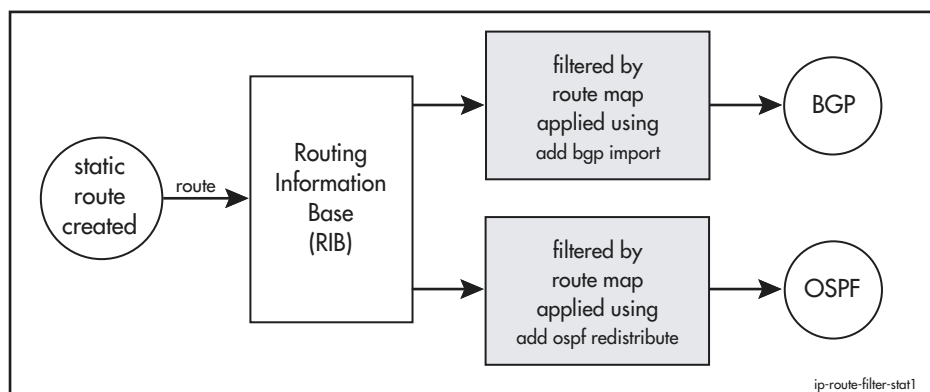
However, BGP, RIP and OSPF also use the interface routes. BGP allows you to filter the interface routes before adding them to its database. RIP uses all interface routes, and does not allow you to filter them. OSPF uses all routes that belong to OSPF interfaces, and does not allow you to filter them. OSPF does not automatically import non-OSPF interface routes, but you can choose to redistribute these into OSPF, with or without filtering.

The following diagram summarises how to filter interface routes.



Statically-Configured Routes

You can manually enter routing information into the switch, which creates static routes. Dynamic routing protocols import these routes. For BGP and OSPF, you can filter static routes when the protocol imports them, as shown in the following figure.



Configuration Examples

These examples apply filters to BGP routes in the following situations:

- [Filtering When Writing BGP Routes to the RIB: Using an AS Path Filter](#)
- [Filtering When Writing BGP Routes to the RIB: Using a Route Map](#)
- [Filtering Before Advertising Routes with BGP: Using an AS Path Filter](#)
- [Filtering Before Advertising Routes with BGP: Using a Route Map](#)
- [Filtering Inbound and Outbound BGP Routes: Using Communities](#)
- [Filtering When Importing Routes from BGP to OSPF](#)

Filtering When Writing BGP Routes to the RIB: Using an AS Path Filter

This example extends the basic BGP configuration shown in [“Basic BGP Configuration” on page 30-42 of Chapter 30, Border Gateway Protocol version 4 \(BGP-4\)](#), which connects two switches as EBGP peers and gives:

- Switch A an IP address of 10.0.0.2 and AS number of 65000
- Switch B an IP address of 10.0.0.1 and AS number of 65001

This example uses the **inpathfilter** filter on a BGP peer. It filters received BGP update messages on the basis of their AS path attributes. Therefore, this example stops the switch from using routes that originated in a particular AS, or that passed through a particular AS.

To set Switch A to filter out update messages that originate from AS 300

1. **Add an AS path list entry.**

```
add ip aspathlist=1 entry=1 exclude="300$"
```

This AS path list includes all update messages that have originated from any AS except AS 300.

2. **Apply the AS path list to the BGP peer.**

```
set bgp peer=10.0.0.1 inpathfilter=1
```

To set Switch A to filter out update messages that originate from AS 300 or pass through AS 200

1. **Add an AS path list entry to exclude update messages that originate in AS 300**

```
add ip aspathlist=1 entry=1 exclude="300$"
```

This AS path list includes all update messages that have originated from any AS except AS 300.

2. **Add an AS path list entry to exclude update messages that go through AS 200**

```
add ip aspathlist=1 entry=2 exclude="200"
```

If a route originated from AS 300 and passes through AS 200, then its update message matches the first entry in the **aspathlist**. BGP does not check the update message against the second entry.

3. **Apply the AS path list to the BGP peer.**

```
set bgp peer=10.0.0.1 inpathfilter=1
```

Filtering When Writing BGP Routes to the RIB: Using a Route Map

This example extends the basic BGP configuration shown in [“Basic BGP Configuration” on page 30-42 of Chapter 30, Border Gateway Protocol version 4 \(BGP-4\)](#), which connects two switches as EBGP peers and gives:

- Switch A an IP address of 10.0.0.2 and AS number of 65000
- Switch B an IP address of 10.0.0.1 and AS number of 65001

This example uses the **inroutemap** filter on a BGP peer. The **inroutemap** filter is applied after all other filters have acted on an update message. This particular route map filters received BGP update messages on the basis of their AS path attributes. It achieves the same effect as the first part of the previous [“Filtering When Writing BGP Routes to the RIB: Using an AS Path Filter”](#) example.

Route map filters are sometimes more useful than path filters because route maps can modify the attributes of a received BGP update message. Path filters only include or exclude messages. However, this example does not demonstrate how to modify the message attributes, because it is meaningless to modify attributes and then discard the message.

To set Switch A to filter out update messages that originate from AS 300

1. Add an AS path list entry.

```
add ip aspathlist=1 entry=1 include="300$"
```

2. Add a route map entry.

```
add ip routemap=as300 entry=1 match aspathlist=1  
action=exclude
```

This entry matches AS paths that are included in the path list, and excludes them.

3. Apply the AS path list to the BGP peer.

```
set bgp peer=10.0.0.1 inroutemap=as300
```

By default, the switch uses all update messages from this peer that do not match the route map.

Filtering Before Advertising Routes with BGP: Using an AS Path Filter

This example extends the basic BGP configuration shown in [“Basic BGP Configuration” on page 30-42 of Chapter 30, Border Gateway Protocol version 4 \(BGP-4\)](#), which connects two switches as EBGP peers and gives:

- Switch A an IP address of 10.0.0.2 and AS number of 65000
- Switch B an IP address of 10.0.0.1 and AS number of 65001

This example uses the **outpathfilter** filter on a BGP peer. It filters transmitted BGP update messages on the basis of their AS path attributes. Therefore, this example stops the peer from learning routes that originated in a particular AS.

To stop Switch A from advertising update messages to a peer, when the update messages originate from AS 550

1. Add an AS path list entry.

```
add ip aspathlist=2 entry=1 exclude="550$"
```

2. Apply the AS path list to the BGP peer.

```
set bgp peer=10.0.0.1 outpathfilter=1
```

By default, the peer receives all update messages that do not match the path filter.

Filtering Before Advertising Routes with BGP: Using a Route Map

This example extends the basic BGP configuration shown in [“Basic BGP Configuration” on page 30-42 of Chapter 30, Border Gateway Protocol version 4 \(BGP-4\)](#), which connects two switches as EBGP peers and gives:

- Switch A an IP address of 10.0.0.2 and AS number of 65000
- Switch B an IP address of 10.0.0.1 and AS number of 65001

This example uses the **out routemap** filter on a BGP peer. The **out routemap** filter is applied after all other filters have acted on an update message. This particular route map filters transmitted BGP update messages on the basis of their AS path attributes. It achieves the same effect as the previous [“Filtering Before Advertising Routes with BGP: Using an AS Path Filter”](#) example.

Route map filters are sometimes more useful than path filters because route maps can modify the attributes of a received BGP update message. Path filters only include or exclude messages. However, this example does not demonstrate how to modify the message attributes, because it is meaningless to modify attributes and then discard the message.

To stop Switch A from advertising update messages to a peer, when the update messages originate from AS 550

1. Add an AS path list entry.

```
add ip aspathlist=2 entry=1 include="550"
```

2. Add a route map entry.

```
add ip routemap=as550 entry=1 match aspathlist=2  
action=exclude
```

This entry matches AS paths that are included in the path list, and excludes them.

3. Apply the route map to the BGP peer.

```
set bgp peer=10.0.0.1 out routemap=as550
```

By default, the peer receives all update messages that do not match the route map.

Filtering Inbound and Outbound BGP Routes: Using Communities

This example extends the basic BGP configuration shown in “[Basic BGP Configuration](#)” on page 30-42 of [Chapter 30, Border Gateway Protocol version 4 \(BGP-4\)](#), which connects two switches as EBGp peers and gives:

- Switch A an IP address of 10.0.0.2 and AS number of 65000
- Switch B an IP address of 10.0.0.1 and AS number of 65001

This example filters inbound and outbound routes on the basis of the community the route belongs to. Switch A assigns routes to different communities depending on the route’s subnet. Switch B only accepts routes that belong to one of these communities.

To use the community attributes

1. On Switch A, create route maps that set the community attribute.

The community number is given in the form *as-number:community*.

```
add ip routemap=map0 entry=1 set community=2:1
add ip routemap=map1 entry=1 set community=2:2
add ip routemap=map2 entry=1 set community=2:3
add ip routemap=map3 entry=1 set community=2:4
add ip routemap=map4 entry=1 set community=2:5
add ip routemap=map5 entry=1 set community=2:6
add ip routemap=map6 entry=1 set community=2:7
```

2. On Switch A, associate the route maps with subnets.

When BGP imports the routes, the route maps set the community attribute.

```
add bgp net=192.168.0.0/24 routemap=map0
add bgp net=192.168.1.0/24 routemap=map1
add bgp net=192.168.2.0/24 routemap=map2
add bgp net=192.168.3.0/24 routemap=map3
add bgp net=192.168.4.0/24 routemap=map4
add bgp net=192.168.5.0/24 routemap=map5
add bgp net=192.168.6.0/24 routemap=map6
add bgp net=192.168.7.0/24 routemap=map6
add bgp net=192.168.8.0/24 routemap=map6
add bgp net=192.168.9.0/24 routemap=map6
add bgp net=192.168.10.0/24 routemap=map6
```

Note that the community attribute of the last five routes are set to the same value (2:7).

3. On Switch A, set BGP to send the community attribute to the peer (Switch B).

```
set bgp peer=10.0.0.1 sendcommunity=yes
```

4. On Switch B, create a community list.

```
add ip communitylist=1 entry=1 include=2:7
add ip communitylist=1 entry=2 exclude=internet
```

This community list consists of those routes with the community attribute value set to 2:7. All other routes are excluded from the community list.

5. On Switch B, use the community list in a route map.

```
add ip routemap=mapin entry=1 match communitylist=1
add ip routemap=mapin entry=2 action=exclude
```

6. On Switch B, apply the route map to updates from the peer (Switch A).

```
set bgp peer=10.0.0.2 sendcommunity=yes inroutemap=mapin
```

Filtering When Importing Routes from BGP to OSPF

This example assumes that you want to import the route 192.168.72.0 into the OSPF routing domain, but no other routes. This route is received on the gateway router as a BGP route. The following steps show the sequence of commands to use in this scenario.

1. Set up the IP filter:

```
add ip filter=300 type=routing source=192.168.72.0
smask=255.255.255.255 action=include
```

2. Set up OSPF BGP import parameters:

```
set ospf bgpimport=on bgpfilter=300 bgplimit=1
```

3. Check that BGP has added the route to the IP route table:

```
show ip route=192.168.72.0
```

The route should be visible in the output of the command.

4. Check that OSPF has imported the route:

```
show ospf lsa=192.168.72.0
```

The output should show that there is an AS external LSA with this ID.

Command Reference

This section describes the commands available on the switch to configure IP route filtering.

The shortest valid command is denoted by capital letters in the Syntax section. See [“Conventions” on page lxvi of About this Software Reference](#) in the front of this manual for details of the conventions used to describe command syntax. See [Appendix A, Messages](#) for a complete list of messages and their meanings.

add ip aspathlist

Syntax `ADD IP ASPATHlist=1..99 [ENTry=1..4294967295]
 INCLude=aspath-reg-exp`

`ADD IP ASPATHlist=1..99 [ENTry=1..4294967295]
 EXCLude=aspath-reg-exp`

Description This command adds an entry to an AS path list, and creates the list if it does not already exist. You must specify the index number of the AS path list, and may also specify the position of the entry in the list.

When the switch searches through an AS path list, the first entry that causes a match stops the search, returning the result **include** or **exclude** depending on the type of entry. A totally empty AS path list is identical to an AS path list that matches all AS paths and is of type **include**. Any non-empty AS path list has an implicit entry at the end that matches all AS paths and is of type **exclude**.

Parameter	Description
ASPATHlist	The ID number of the AS path list. You can create up to 99 AS path lists. Default: no default
ENTry	The desired position of the new entry in the AS path list once the entry has been added. Entries are numbered from 1 to the number of entries in the list. Default: The entry is added to the end of the list

Parameter (cont.)	Description (cont.)
INCLude	<p>An AS path regular expression, which specifies the AS path values that this entry includes in this AS path list. When you use the AS path list in a route map or filter, the map or filter carries out its specified action on update messages with a matching AS path attribute value.</p> <p>Regular expressions are a list of one or more AS numbers, separated by spaces. To match from the first number in the list, start the expression with the ^ character. To match the last number, end with the \$ character. If the expression contains spaces, surround it with double quotes. For more information about valid syntax, see Table 31-1 on page 31-10. For example:</p> <p>include="23334 45634 88988" includes any path containing these numbers</p> <p>include="^23334 45634 88988\$" includes only that exact path</p> <p>include=^23334 includes any path that begins with 23334</p> <p>Default: no default</p>
EXCLude	<p>An AS path regular expression, which specifies the AS path values that this entry excludes from this AS path list. When you use the AS path list in a route map or filter, the map or filter does not carry out its specified action on update messages with a matching AS path attribute value.</p> <p>Regular expressions are a list of one or more AS numbers, separated by spaces. To match from the first number in the list, start the expression with the ^ character. To match the last number, end with the \$ character. If the expression contains spaces, surround it with double quotes. For more information about valid syntax, see Table 31-1 on page 31-10. For example:</p> <p>exclude="23334 45634 88988" excludes any path containing these numbers</p> <p>exclude="^23334 45634 88988\$" excludes only that exact path</p> <p>exclude=23334\$ excludes any path that ends with 23334</p> <p>Default: no default</p>

Examples To add an entry to AS path list 1 that matches all AS paths and excludes them, use the command:

```
add ip aspath=1 excl=.*
```

To add an entry to AS path list 2 that matches an empty AS path and includes it, use the command:

```
add ip aspath=2 incl=^$
```

Related Commands

- [add ip routemap](#)
- [delete ip aspathlist](#)
- [show ip aspathlist](#)

add ip communitylist

Syntax

```
ADD IP COMmunitylist=1..99 [ENTry=1..4294967295]
    INCLude={INTernet|NOExport|NOAdvertise|
    NOEXPORTSubconfed|aa:xx}[, ...]

ADD IP COMmunitylist=1..99 [ENTry=1..4294967295]
    EXCLude={INTernet|NOExport|NOAdvertise|
    NOEXPORTSubconfed|aa:xx}[, ...]
```

Description This command adds an entry to a community list, and creates the list if it does not already exist. You must specify the index number of the community list, and may also specify the position of the entry in the list.

Parameter	Description
COMmunitylist	The ID number of the community list. You can create up to 99 lists. Default: no default
ENTry	The desired position of the new entry in the community list once the entry has been added. Entries are numbered from 1 to the number of entries in the list. Default: The entry is added to the end of the list
INCLude	A community name, community number, or comma-separated list of names and numbers, which specifies the communities that this entry includes in this community list. When you use the community list in a route map or filter, the map or filter carries out its specified action on update messages with a matching community attribute value. Default: no default
INTernet	The community of routes that can be advertised to all BGP peers.
NOExport	The community of routes that must not be advertised outside a BGP confederation boundary (a standalone autonomous system that is not part of a confederation should be considered a confederation itself).
NOAdvertise	The community of routes that must not be advertised to other BGP peers.
NOEXPORTSubconfed	The community of routes that must not be advertised to external BGP peers (this includes peers in other members' autonomous systems inside a BGP confederation).
aa:xx	The number of a community. aa and xx are both integers from 0 to 65534. aa is the AS number. xx is a value chosen by the ASN administrator.

Parameter (cont.)	Description (cont.)
EXCLude	<p>A community name, community number, or comma-separated list of names and numbers, which specifies the communities that this entry excludes from this community list. When you use the community list in a route map or filter, the map or filter does not carry out its specified action on update messages with a matching community attribute value.</p> <p>Default: no default</p>
INTernet	The community of routes that can be advertised to all BGP peers.
NOExport	The community of routes that must not be advertised outside a BGP confederation boundary (a standalone autonomous system that is not part of a confederation should be considered a confederation itself).
NOAdvertise	The community of routes that must not be advertised to other BGP peers.
NOEXPORTSubconfed	The community of routes that must not be advertised to external BGP peers (this includes peers in other members' autonomous systems inside a BGP confederation).
aa:xx	The number of a community. aa and xx are both integers from 0 to 65534. aa is the AS number. xx is a value chosen by the ASN administrator.

Examples To add an entry to community list 1 that matches communities attributes that contain the community NOEXPORT and excludes them, use the command:

```
add ip com=1 excl=noe
```

Related Commands

- [add ip routemap](#)
- [delete ip communitylist](#)
- [show ip communitylist](#)

add ip prefixlist

Syntax ADD IP PREFIXList=*name* ENTry=1..65535
[ACTion={MATch|NOMatch}] [MASklength=*range*]
[PREfix=*ipadd*]

Description This command adds a numbered *entry* to a prefix list. If the prefix list does not already exist, this command first creates it. You can create up to 400 prefix lists, with up to 1000 entries in each list.

Parameter	Description				
PREFIXList	<p>A name to identify the prefix list. A string 1 to 15 characters long. Valid characters are uppercase letters (A-Z), lowercase letters (a-z), digits (0-9) and the underscore. If <i>name</i> contains spaces, it must be in double quotes.</p> <p>Default: no default</p>				
ENTry	<p>An integer to specify the position of the new entry in the prefix list. When the switch uses a prefix list, it checks the entries in order, starting with the lowest, until it finds a match. Therefore, give more specific entries lower numbers than general entries. If you leave gaps between entry numbers, you can add future entries between existing entries.</p> <p>Each prefix list has an implicit final entry that matches all addresses, with an action of nomatch.</p> <p>Default: no default</p>				
ACTion	<p>Whether matching prefixes are included or excluded by the process that is using the prefix list.</p> <p>You can use multiple entries in a prefix list with actions of match and nomatch to build up a list of prefixes. Prefixes with action=match are included in the list. Then to use this list of prefixes, create a route map that matches it and apply the route map in a route filtering process. The route map also has an action parameter, which determines whether the filtering process includes or excludes the prefixes in the list.</p> <p>Default: match</p> <table><tr><td>MATch</td><td>The prefix list includes the prefix.</td></tr><tr><td>NOMatch</td><td>The prefix list excludes the prefix.</td></tr></table>	MATch	The prefix list includes the prefix.	NOMatch	The prefix list excludes the prefix.
MATch	The prefix list includes the prefix.				
NOMatch	The prefix list excludes the prefix.				

Parameter (cont.)	Description (cont.)
MASKlength	<p>The range of prefix mask lengths matched by this entry in the prefix list. The <i>range</i> is either a single CIDR mask from 0 to 32, or two masks separated by a hyphen. These options are valid for setting the mask length:</p> <ul style="list-style-type: none"> as a mask length range (masklength=a-b). For a route to match against this entry, its prefix mask length must be between <i>a</i> and <i>b</i> inclusive. <i>a</i> must be less than <i>b</i>. as a single mask length (masklength=a). For a route to match against this entry, its prefix mask length must be exactly <i>a</i>. as an implicit mask length, by not specifying masklength (for example, prefix=192.168.0.0). For a route to match against this entry, its prefix mask length must correspond exactly to the mask for the class of the given address; in this example, 24. <p>Default: The natural mask for the prefix, based on whether it is a class A, B, or C network</p>
PREFIX	<p>The network address matched by this entry in the prefix list, specified in dotted decimal notation.</p> <p>If you do not specify a prefix, the switch sets it to 0.0.0.0. This is correct if you are matching all routes or the default route.</p> <p>Default: 0.0.0.0</p>

Examples To match only routes from the 192.168.0.0/16 network, use the command:

```
add ip prefixlist=sample1 entry=1 action=match
    prefix=192.168.0.0 masklength=16
```

To match all routes in all 192.168.0.0 networks, except those in the 192.168.7.0 network, use the commands:

```
add ip prefixlist=sample2 entry=1 action=nomatch
    prefix=192.168.7.0 masklength=24-32

add ip prefixlist=sample2 entry=2 action=match
    prefix=192.168.0.0 masklength=16-32
```

To exclude the default route, use the command:

```
add ip prefixlist=sample3 entry=1 action=nomatch masklength=0
```

To include all routes, use the command:

```
add ip prefixlist=sample4 entry=1 action=match
    masklength=0-32
```

Related Commands

- [add ip routemap](#)
- [delete ip prefixlist](#)
- [set ip prefixlist](#)
- [set ip routemap](#)
- [show ip prefixlist](#)

add ip route filter

Syntax `ADD IP ROUTe FILTER [=filter-id] IP=ipadd MASK=ipadd
 ACTION={INCLUDE|EXCLUDE|SWITCH}
 [DIRECTION={RECEIVE|SEND|BOTH}] [INTERFACE=interface]
 [NEXTHop=ipadd] [POLICY=0..7] [PROTOCOL={ANY|OSPF|RIP}]`

Description This command creates a route filter. A route filter can control which routes RIP receives and advertises, and which external routes OSPF copies into its LSA database.

Parameter	Description
Filter	The ID number of the filter, from 1 to 100. Default: no default (the filter is added to the end of the list of currently-defined filters)
IP	The network address to match. You can use the wildcard character ("*") to match a network range. For example, 192.168.*.* matches all destination networks that start with 192.168. The wildcard character can only replace a complete number. For example, 192.168.*.* is valid but 192.16*.*.* is not. Default: no default
MASK	The network mask of the network to match. You can use the wildcard character ("*") to match a network mask range. For example, 255.255.*.* matches all destination network masks that start with 255.255. The wildcard character can only replace a complete number. For example, 255.255.*.* is valid but 255.25*.*.* is not. Default: no default
Action	What the switch does with routes that match the filter. Default: no default
	INCLUDE The switch includes matching routes in its RIB or the advertisement.
	EXCLUDE The switch excludes matching routes from its RIB or the advertisement.
	SWITCH The switch learns matching routes and adds them to the special default IP route table in hardware. The default IP route table can contain up to 16 summary routes.

Parameter (cont.)	Description (cont.)
Dlirection	<p>Whether the switch applies this filter to routes that the routing protocol receives or routes that it advertises. The routing protocol is specified using the protocol parameter.</p> <p>Default: both</p>
RECeive	<p>The switch applies this filter to routes that the routing protocol receives, to determine whether to write those routes into the RIB.</p> <p>If protocol=ospf, a route filter with direction=receive filters routes when copying them from the LSA database to the RIB. If a filter excludes a matching route from the RIB, OSPF does not advertise a summary LSA for that route because summary LSA messages are derived from the filtered RIB. This means that incorrect filters can prevent Area Border Routers from advertising routes to other areas. Plan your filters carefully.</p>
SENd	<p>The switch applies this filter to routes, to determine whether the routing protocol will advertise the routes.</p> <p>If protocol=ospf, a route filter with direction=send only matches AS external routes (BGP, RIP and static routes) and summary routes.</p>
BOTH	<p>The switch applies this filter to determine which routes to write into the RIB and which routes to advertise.</p>
INTErface	<p>The interface to which the filter applies. The switch only uses this filter on routes that are received on this interface, or that will be advertised out this interface. Valid interfaces are:</p> <ul style="list-style-type: none"> PPP (such as ppp0, ppp1-1) FR (such as fr0, fr0-1) X.25 DTE (such as x25t0, x25t0-1) VLAN (such as vlan1, vlan1-1) <p>To see a list of interfaces currently available, use the show ip interface command on page 23-186 of Chapter 23, Internet Protocol (IP).</p> <p>If a logical interface is specified, the route filter is only applied to the specified logical interface. If a logical interface is not specified, 0 is assumed and the route filter is only applied to logical interface 0.</p> <p>If protocol=ospf, this parameter has no effect. The switch always applies the filter on all interfaces.</p> <p>Default: no default (the switch applies this filter to routes on all interfaces)</p>
NEXThop	<p>The IP address of the next hop router. If you specify this, the switch applies this filter to routes that specify this next hop.</p> <p>Default: no default</p>
POLlcy	<p>The value of the route's Type of Service, from 0 to 7. The filter matches routes with this TOS setting.</p> <p>Default: no default</p>

Parameter (cont.)	Description (cont.)
PROTOcol	The routing protocol to which the filter applies. If direction is receive , then protocol specifies the routing protocol that receives the route information. If direction is send , then protocol specifies the routing protocol that advertises the routes. Default: any
OSPF	Open Shortest Path First
RIP	Routing Information Protocol
ANY	Both RIP and OSPF

Examples To add a route filter that includes RIP-derived routes from all sources, use the command:

```
add ip rou fil=1 prot=rip ac=incl di=both ip=*. *.*.*.*
mask=*. *.*.*.*
```

To exclude all routes received from the 10.0.0.0 network from the route table, but include all other received routes in the route table, use the commands:

```
add ip rou fil=1 ip=10.0.0.0 mask=255.0.0.0 ac=excl di=rec
add ip rou fil=2 ip=*. *.*.*.* mask=*. *.*.*.* ac=incl
```

The second filter is necessary to override the effect of the implicit “exclude all” following the last entry in a filter list.

Related Commands

- [delete ip route filter](#)
- [set ip route filter](#)
- [show ip route filter](#)

add ip routemap

Syntax for an empty entry	<pre>ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}]</pre>
Syntax to match any route type	<pre>ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch INTERface=interface ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch METric=0..4294967295[-0..4294967295] ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch NEXThop=ipadd ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch PREFIXList=prefixlist-name ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch TAG=1..65535</pre>
Syntax to match OSPF routes	<pre>ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch ROUTEsource=prefixlist-name ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch ROUTEType={INTRA INTER TYPE1 TYPE2 OTHER}</pre>
Syntax to match BGP routes	<pre>ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch ASPath=1..99 ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch COMMunity=1..99 [EXAct={NO YES}] ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action=INCLude] MAtch MED=0..4294967295 ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] MAtch ORIGIN={EGP IGP INComplete}</pre>
Syntax to set any route type	<pre>ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action=INCLude] SET METric=0..4294967295 ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action={INCLude EXCLude}] SET TAG=1..65535</pre>
Syntax to set OSPF routes	<pre>ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action=INCLude] SET TYpe={1 2}</pre>
Syntax to set BGP routes	<pre>ADD IP ROUTEMap=routemap ENTry=1..4294967295 [Action=INCLude] SET ASPath={1..65534[,...]}</pre>

```
ADD IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET
COMmunity={NOExport|NOAdvertise|NOEXPORTSubconfed|
aa:xx}[,...] [ADD={NO|YES}]
```

```
ADD IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET BGPdampid=1..100
```

```
ADD IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET LOCalpref=0..4294967295
```

```
ADD IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET MED={0..4294967295|REMOVE}
```

```
ADD IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET ORIGIn={IGP|EGP|INCOmplete}
```

Description This command adds a numbered *entry* to a route map, or adds a *clause* to an existing entry in a route map. If the route map does not already exist, this command first creates it.

Route maps are made up of a list of entries. Each entry contains:

- zero or one **match** clause, to determine which routes or BGP update messages the entry applies to. If an entry does not have a match clause, the effect is that it matches everything.
- one **action**, to determine whether matching routes or BGP update messages are included or excluded by the process that is using the route map (by default matching items are included).
- zero, one, or more **set** clauses, to change certain features of matching routes or the attributes of matching BGP updates. Most **set** clauses change the attributes of matching update messages. Each entry can have at most one **set** clause of a given type.

Parameter table order The following tables list the parameters that apply to both **match** and **set** clauses, then the **match** clauses, then the **set** clauses. The match and set clause tables first list parameters that apply to all routes, then parameters that only apply to OSPF routes, then parameters that only apply to BGP routes.

Parameters for both *match* and *set* clauses

Parameter	Description
ROUTEMap	<p>The name of the route map to add the entry or clause to. The <i>routemap</i> is a character string 0 to 15 characters long. Valid characters are uppercase and lowercase letters, digits (0-9), and the underscore character.</p> <p>Default: no default</p>
ENTry	<p>An integer to specify the position of the new entry in the route map. When a routing protocol uses a route map, it checks the entries in order, starting with the lowest, until it finds a match. If you leave gaps between entry numbers, you can add future entries between existing entries.</p> <p>Be careful when specifying the entry number. If you make an error in the number (for example, enter <code>entry=11</code> instead of <code>entry=1</code>), the switch adds a new entry to the route map.</p> <p>Default: no default</p>
ACtion	<p>Whether matching prefixes or update messages are included or excluded by the process that is using the route map.</p> <p>The action parameter applies to the entire entry, but you can change it at the same time as you add a clause. The most recently entered value of this parameter applies to the entire entry.</p> <p>It is not meaningful to have action=exclude in an entry with a set clause.</p> <p>Default: the current setting. If there is no current setting, include</p>

Parameters for *match* clauses

Parameter	Description
MATch	<p>Adds a match clause to the entry, to determine which routes or BGP update messages the entry applies to. A route map entry can have zero or one match clauses. An entry without a match clause matches all routes or updates.</p>
INTerface (all route types)	<p>A switch interface. A route matches the route map entry if its interface matches the specified interface. Valid interfaces are:</p> <ul style="list-style-type: none"> PPP (such as <code>ppp0</code>, <code>ppp1-1</code>) FR (such as <code>fr0</code>, <code>fr0-1</code>) X.25 DTE (such as <code>x25t0</code>, <code>x25t0-1</code>) VLAN (such as <code>vlan1</code>, <code>vlan1-1</code>) <p>To see a list of interfaces currently available, use the show interface command on page 11-87 of Chapter 11, Interfaces.</p> <p>Default: no default</p>
METric (all route types)	<p>The route metric or a range of metric values. A route matches the route map entry if its metric equals this value or is in this range.</p> <p>Default: no default</p>
NEXThop (all route types)	<p>The IP address of the next node in the path to the route's destination, specified in dotted decimal notation. For BGP, an update message matches the route map entry if its <code>next_hop</code> attribute matches this address.</p> <p>Default: no default</p>

	Parameter (cont.)	Description (cont.)
	PREFIXList (all route types)	The name of a prefix list. A route matches the route map entry if the prefix list contains that route. To create a list use the add ip prefixlist command on page 31-44 . Default: no default
	TAG (all route types)	A tag that identifies a particular route. A route matches this route map entry if it has been tagged with this value. There are two ways of tagging routes. You can use a route map to set the route's tag, or for static routes you can use the tag parameter of the add ip route command on page 23-80 of Chapter 23, Internet Protocol (IP) . For BGP, you can use a route map that matches on tag when you use the add bgp import command to import static routes from the RIB to BGP. However, BGP routes do not have a tag field in their path attributes. Therefore, you cannot use tag to filter routes that are sent to BGP peers or to match update messages that are received from BGP peers. For OSPF, you can use a route map that matches on tag when you use the add ospf redistribute command to import routes from the RIB to OSPF. Default: no default
parameters for OSPF routes only	ROUTEsource (OSPF routes only)	The name of a prefix list that lists one or more router IDs. A route matches the route map entry if the prefix list contains the router ID of the router that advertised the route to OSPF. To create a prefix list use the add ip prefixlist command on page 31-44 . Note that the mask for a router ID must be 255.255.255.255, so the mask length must be 32. Default: no default
	ROUTEType (OSPF routes only)	The type of route, which indicates whether the route is within the OSPF area, to another area with the same AS, or to another AS. See "Routing with OSPF" on page 29-9 of Chapter 29, Open Shortest Path First (OSPF) for more information about these route types. Default: no default
	INTRA	A route matches the route map entry if it is an OSPF intra-area route.
	INTER	A route matches the route map entry if it is an OSPF inter-area route.
	TYPE1	A route matches the route map entry if it is an OSPF External Type 1 route.
parameters for BGP routes only	TYPE2	A route matches the route map entry if it is an OSPF External Type 2 route.
	OTHER	A route matches the route map entry if it is not one of the above route types.
	ASPath (BGP routes only)	The ID number of an AS path list. An update message matches the route map entry if its AS path attribute matches the AS path list. To configure an AS path list use the add ip aspathlist command on page 31-40 . Default: no default
	COMMunity (BGP routes only)	The ID number of a community list. An update message matches the route map entry if its community attribute matches the community list. To configure a community list use the add ip communitylist command on page 31-42 . Default: no default

Parameter (cont.)	Description (cont.)
EXAct (BGP routes only)	Whether the community attribute in an update message must precisely match the route map's community list. Only valid when you specify both match and community . Default: no
	YES An update message only matches the route map entry if its community attribute contains all the communities specified in the community list and only those communities.
	NO An update message still matches the route map entry if its community attribute contains all the communities specified in the community list plus extra communities.
MED (BGP routes only)	The update message's Multi_Exit_Discriminator attribute. EBGp uses the MED to determine the optimal path for reaching the advertised prefixes. A lower metric indicates a preferred path. IBGP does not use this attribute. An update message matches the route map entry if its MED attribute matches this value. Default: no default
ORIGin (BGP routes only)	An origin attribute value, which indicates BGP's source for the routes at their originating AS. An update message matches the route map entry if its origin attribute matches this value. Default: no default
	IGP The original source of the route was IGP.
	EGP The original source of the route was EGP.
	INCOmplete The original source of the route was neither IGP or EGP. This includes statically-configured routes.

Parameters for set clauses

Parameter	Description
SET	Adds a set clause to the entry. These clauses modify characteristics of routes that match the entry or, for BGP routes, an attribute in update messages that match the entry. A route map entry can have zero, one or more set clauses, but can only modify each attribute once. An entry without a set clause does not modify any attributes.
METric (all route types)	The metric to give to the route. For RIP, values higher than 16 are treated as if they are 16. Default: no default
TAG (all route types)	A number to label matching routes with. Tagging routes allows you to identify the route's original source, for example, in the output of the show ip route command. For example, you can tag BGP routes when you redistribute them into OSPF or RIP, which enables you to identify the routes that came from BGP. Default: no default
Type (OSPF routes only)	The OSPF external route type to set the route to. This enables you to ensure that all externally-sourced OSPF routes are the same type and therefore use the same method to calculate route metrics. Default: no default
ASPath (BGP routes only)	A comma-separated list of 1 to 10 AS numbers. These numbers are added to the beginning of the update message's AS path attribute. Default: no default

Parameter (cont.)	Description (cont.)
COMmunity (BGP routes only)	<p>A comma-separated list of 1 to 10 communities, identified by name or number. If the add parameter is yes, these communities are added to the update message's community attribute. If the add parameter is no (its default), these communities replace the update message's community attribute.</p> <p>Note that you must also set the peer's sendcommunity parameter to yes if you want the peer to include the community attribute in the update messages it sends. By default, peers do not include the community attribute in outgoing updates.</p> <p>Default: no default</p>
INTernet	The community of routes that can be advertised to all BGP peers.
NOExport	The community of routes that must not be advertised outside a BGP confederation boundary (a standalone autonomous system that is not part of a confederation should be considered a confederation itself).
NOAdvertise	The community of routes that must not be advertised to other BGP peers.
NOEXPORTSubconfed	The community of routes that must not be advertised to external BGP peers (this includes peers in other members' autonomous systems inside a BGP confederation).
aa:xx	The number of a community. aa and xx are both integers from 0 to 65534. aa is the AS number. xx is a value chosen by the ASN administrator.
ADD (BGP routes only)	<p>Whether the list of communities specified by the community parameter is added to the community attribute, or replaces the community attribute. Only valid when you specify both set and community.</p> <p>Default: no</p>
YES	The communities are added to the update message's community attribute.
NO	The communities replace the update message's community attribute.
BGPDampid (BGP routes only)	<p>The BGP route flap damping ID that is given to matching routes. This is the same as the ID number of the parameter set that maintains that route's FoM upon it exhibiting instability. If the parameter set does not exist, the default parameter set is applied to matching routes.</p> <p>For more information about using route maps when configuring route flap damping, see "Damping routes on specific peers" on page 30-27 of Chapter 30, Border Gateway Protocol version 4 (BGP-4).</p> <p>Default: no default</p>
LOCalpref (BGP routes only)	<p>The metric to write into the update message's local_preference attribute. IBGP uses the local preference to determine which path it should use inside the AS to reach the advertised prefix. A lower metric indicates a preferred path. EBGP does not use this attribute.</p> <p>Default: no default</p>

Parameter (cont.)	Description (cont.)
MED (BGP routes only)	The metric to write into the update message's Multi_Exit_Discriminator attribute. EBGp uses the MED to determine the optimal path for reaching the advertised prefixes. A lower metric indicates a preferred path. IBGP does not use this attribute. Default: no default
	0..4294967295 This value is written into the MED attribute of the matched update message.
	REMOVE The MED attribute is removed from the matched update message.
ORIGin (BGP routes only)	The value to write into the update message's origin attribute. The origin indicates BGP's source for the routes at their originating AS. Default: no default
	IGP The original source of the route was IGP.
	EGP The original source of the route was EGP.
	INCOmplete The original source of the route was neither IGP or EGP. This includes statically-configured routes.

Examples To add a route map entry that sets the community attribute to 489816064 for all BGP routes, use the command:

```
add ip routemap=set_comm ent=10 set com=489816064
```

The above command creates the route map, adds an entry to it, and adds a set clause to the entry. No match clause is required because we wish to match all routes. To use this route map for routes being sent to BGP peer 192.168.1.1, use the command:

```
set bgp peer=192.168.1.1 outr=set_comm
```

To add a route map entry number 10 that selects all routes with an OSPF metric from 5 to 15, use the command:

```
add ip routemap=metric_ent=10 ma met=5-15
```

Related Commands

- [delete ip routemap](#)
- [set ip routemap](#)
- [show ip routemap](#)

delete ip aspathlist

Syntax DELEte IP ASPATHlist=1..99 [ENTry=1..4294967295]

Description This command deletes an entry from an AS path list or deletes an entire AS path list. You cannot delete an AS path list if a route map is using it, or if a peer is using it as a filter. First use the **match** parameter of the [delete ip routemap command on page 31-59](#) to delete the route map entry, or the **set bgp peer** command on page 30-93 of Chapter 30, Border Gateway Protocol version 4 (BGP-4) to remove the filter association.

Parameter	Description
ASPATHlist	The ID number of the AS path list to delete, or to remove an entry from. Default: no default
ENTry	The number of the entry to delete. If you do not specify an entry, the whole AS path list is deleted. Default: no default

Examples To delete the third entry in AS path list 1, use the command:

```
del ip aspath=1 ent=3
```

To delete AS path list 1 and all its entries, use the command:

```
del ip aspath=1
```

Related Commands [add ip aspathlist](#)
[show ip aspathlist](#)

delete ip communitylist

Syntax `DELEte IP COMMunitylist=1..99 [ENTry=1..4294967295]`

Description This command deletes an entry from a community list or the entire list. You cannot delete a community list if a route map is using it. First use the **match** parameter of the [delete ip routemap command on page 31-59](#) to delete the route map entry.

Parameter	Description
COMmunitylist	The ID number of the community list to delete, or to remove an entry from. Default: no default
ENTry	The number of the entry to delete. If you do not specify an entry, the whole community list is deleted. Default: no default

Examples To delete the entire community list 1, use the command:

```
del ip com=1
```

Related Commands [add ip communitylist](#)
[show ip communitylist](#)

delete ip prefixlist

Syntax `DELEte IP PREFIXList[=name] [ENTry=1..65535]`

Description This command deletes one of the following:

- an entry from a particular prefix list if you specify a name in the **prefixlist** parameter and an **entry** number
- a prefix list if you specify a name in the **prefixlist** parameter but do not specify an **entry** number
- all prefix lists if you do not specify a name in the **prefixlist** parameter or an **entry** number

You cannot delete a prefix list if a route map is using it. Delete the route map entry first.

Examples To delete entry 2 from the prefix list “office”, use the command:

```
del ip prefixl=office entry=2
```

Related Commands [add ip prefixlist](#)
[delete ip routemap](#)
[set ip routemap](#)
[show ip prefixlist](#)

delete ip route filter

Syntax DELEte IP ROUte FILter=1..100

Description This command deletes a route filter. A route filter controls which routes are sent and received by the routing protocols.

The **filter** parameter specifies the index in the filter list of the filter to delete. The specified entry must exist.

Examples To delete route filter 3, use the command:

```
del rou fil=3
```

Related Commands [add ip route filter](#)
[set ip route filter](#)
[show ip route filter](#)

delete ip routemap

Syntax `DELEte IP ROUTEMap=routemap`

`DELEte IP ROUTEMap=routemap ENTRy=1..4294967295`

`DELEte IP ROUTEMap=routemap ENTRy=1..4294967295
 MAtch={ASPath|COMMunity|INTERface|MED|METric|NEXThop|
 ORIGin|PREFIXList|ROUTESource|ROUTEType|TAG}`

`DELEte IP ROUTEMap=routemap ENTRy=1..4294967295
 SET={ASPath|COMMunity|LOCALpref|MED|METric|ORIGin|TAG|
 TYPe}`

Description This command deletes one of:

- an entire route map
- a single entry in a route map, or
- a match or set clause in an entry in a route map

You cannot delete a whole route map, or a route map's last entry, if OSPF or a BGP peer is using it, or if a BGP aggregate, network or import process is using it.

Parameter	Description
ROUTEMap	The name of the route map to be deleted or the name of the route map from which an entry, match clause, or set clause is to be deleted. Default: no default
ENTRy	The number of the entry in the route map to be deleted, or the number of the entry from which a match clause or set clause is to be deleted. The entry must already exist in the route map. If you do not specify an entry, the whole route map is deleted. Default: no default
MAtch	The type of match clause to be deleted from the route map entry. Since only one match clause is allowed in a route map entry, this uniquely identifies the clause. Default: no default
SET	The type of set clause to be deleted from the route map entry. Since only one set clause of each type is allowed in a route map entry, this uniquely identifies the clause. Default: no default

Examples To delete the localpref set clause from entry 10 in route map “set_loc_pref”, use the command:

```
del ip routem=set_loc_pref ent=10 set=loc
```

To delete the next hop match clause from entry 10 in route map “nexthop”, use the command:

```
del ip routem=nexthop ent=10 ma=next
```

Related Commands [add ip routemap](#)
[set ip routemap](#)
[show ip routemap](#)

set ip prefixlist

Syntax SET IP PREFIXList=*name* ENTry=1..65535
 [ACTion={MATch|NOMatch}] [MASklength=*range*]
 [PREFIX=*ipadd*]

Description This command modifies an existing entry in a prefix list.

Parameter	Description				
PREFIXList	A name that identifies the prefix list. Default: no default				
ENTry	An integer that specifies the position of the entry in the prefix list. Default: no default				
ACTion	Whether matching prefixes are included or excluded by the process that is using the prefix list. You can use multiple entries in a prefix list with actions of match and nomatch to build up a list of prefixes. Prefixes with action=match are included in the list. Then to use this list of prefixes, create a route map that matches it and apply the route map in a route filtering process. The route map also has an action parameter, which determines whether the filtering process includes or excludes the prefixes in the list. Default: match <table> <tr> <td>MATch</td><td>The prefix list includes the prefix.</td></tr> <tr> <td>NOMatch</td><td>The prefix list excludes the prefix.</td></tr> </table>	MATch	The prefix list includes the prefix.	NOMatch	The prefix list excludes the prefix.
MATch	The prefix list includes the prefix.				
NOMatch	The prefix list excludes the prefix.				
MASKlength	The range of prefix mask lengths matched by this entry in the prefix list. The <i>range</i> is either a single CIDR mask from 0 to 32, or two masks separated by a hyphen. These options are valid for setting the mask length: as a mask length range (masklength=a-b). For a route to match against this entry, its prefix mask length must be between <i>a</i> and <i>b</i> inclusive. <i>a</i> must be less than <i>b</i> . as a single mask length (masklength=a). For a route to match against this entry, its prefix mask length must be exactly <i>a</i> . as an implicit mask length, by not specifying masklength (for example, prefix=192.168.0.0). For a route to match against this entry, its prefix mask length must correspond exactly to the mask for the class of the given address; in this example, 24. Default: The natural mask for the prefix, based on whether it is a class A, B, or C network				
PREFIX	The network address matched by this entry in the prefix list, specified in dotted decimal notation. If you do not specify a prefix, the switch assumes the prefix is unchanged, and this entry's previously given prefix is used.				

Examples To modify entry 1 in prefix list sample1 so that it matches only routes from the 192.168.0.0/16 network, use the command:

```
set ip prefixlist=sample1 entry=1 action=match
prefix=192.168.0.0 masklength=16
```

Related Commands

- add ip prefixlist
- add ip routemap
- delete ip prefixlist
- set ip routemap
- show ip prefixlist

set ip route filter

Syntax SET IP ROUTe FILTER=*filter-id* [IP=*ipadd*] [MASK=*ipadd*]
[ACTion={INCLude|EXCLude|SWItch}]
[DIREction={RECeive|SENd|BOTH}] [INTerface=*interface*]
[NEXThop=*ipadd*] [POLIcy=0..7] [PROTOcol={ANY|OSPF|RIP}]

Description This command modifies a route filter. A route filter controls which routes are sent and received by the routing protocols. Route filters do not apply to static or interface routes.

Parameter	Description
FILTER	The ID number of the filter, from 1 to 100. Default: no default
IP	The network address to match. You can use the wildcard character ("*") to match a network range. For example, 192.168.*.* matches all destination networks that start with 192.168. The wildcard character can only replace a complete number. For example, 192.168.*.* is valid but 192.16*.*.* is not. Default: no default
MASK	The network mask of the network to match. You can use the wildcard character ("*") to match a network mask range. For example, 255.255.*.* matches all destination network masks that start with 255.255. The wildcard character can only replace a complete number. For example, 255.255.*.* is valid but 255.25*.*.* is not. Default: no default
ACTion	What the switch does with routes that match the filter. Default: no default
	INCLude The switch includes matching routes in its RIB or the advertisement.
	EXCLude The switch excludes matching routes from its RIB or the advertisement.
	SWItch The switch learns matching routes and adds them to the special default IP route table in hardware. The default IP route table can contain up to 16 summary routes.

Parameter (cont.)	Description (cont.)
Dlirection	<p>Whether the switch applies this filter to routes that the routing protocol receives or routes that it advertises. The routing protocol is specified using the protocol parameter.</p> <p>Default: both</p>
RECeive	The switch applies this filter to routes that the routing protocol receives, to determine whether to write those routes into the RIB.
SENd	<p>The switch applies this filter to routes, to determine whether the routing protocol will advertise the routes. Note that the nature of the OSPF protocol affects how route filtering works on OSPF Link State Advertisement (LSA). A route filter with direction=send only matches Autonomous System (AS) external routes. Also, the switch ignores the interface parameter, so it applies the filter on all interfaces.</p>
BOTH	The switch applies this filter to determine which routes to write into the RIB and which routes to advertise.
INTErface	<p>The interface to which the filter applies. The switch only uses this filter on routes that are received on this interface, or that will be advertised out this interface. Valid interfaces are:</p> <ul style="list-style-type: none"> PPP (such as ppp0, ppp1-1) FR (such as fr0, fr0-1) X.25 DTE (such as x25t0, x25t0-1) VLAN (such as vlan1, vlan1-1) <p>To see a list of interfaces currently available, use the show ip interface command on page 23-186 of Chapter 23, Internet Protocol (IP).</p> <p>If a logical interface is specified, the route filter is only applied to the specified logical interface. If a logical interface is not specified, 0 is assumed and the route filter is only applied to logical interface 0.</p> <p>If protocol=ospf, the switch ignores this setting when filtering routes to advertise.</p> <p>Default: no default (the switch applies this filter to routes on all interfaces)</p>
NEXThop	<p>The IP address of the next hop router. If you specify this, the switch applies this filter to routes that specify this next hop.</p> <p>Default: no default</p>
POLlcy	<p>The value of the route's Type of Service, from 0 to 7. The filter matches routes with this TOS setting.</p> <p>Default: no default</p>
PROToCol	<p>The routing protocol to which the filter applies. If direction is receive, then protocol specifies the routing protocol that receives the route information. If direction is send, then protocol specifies the routing protocol that advertises the routes.</p> <p>Default: any</p>
OSPF	Open Shortest Path First
RIP	Routing Information Protocol
ANY	Both RIP and OSPF

Examples To modify route filter 1 to include only OSPF-derived routes, use the command:

```
set ip rou fil=1 prot=ospf
```

Related Commands [add ip route filter](#)
[delete ip route filter](#)
[show ip route filter](#)

set ip routemap

Syntax to change the action

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}]
```

Syntax to change a match clause

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch ASPath=1..99
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch COMMunity=1..99
[EXAct={NO|YES}]
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch INTERface=interface
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] MAtch MED=0..4294967295
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch
METric=0..4294967295 [-0..4294967295]
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch NEXThop=ipadd
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch
ORIGin={EGP|IGP|INComplete}
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch
PREFIXList=prefixlist-name
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch
ROUTEsource=prefixlist-name
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch
ROUTEType={INTRA|INTER|TYPE1|TYPE2|OTHER}
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action={INCLude|EXCLude}] MAtch TAG=1..65535
```

Syntax to change a set clause

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET ASPath={1..65534[,...]}
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET BPGDampid=1..100
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET
COMMunity={NOExport|NOAdvertise|NOEXPORTSubconfed|
aa:xx}[,...] [ADD={NO|YES}]
```

```
SET IP ROUTEMap=routemap ENTry=1..4294967295
[Action=INCLude] SET LOCALpref=0..4294967295
```

```

SET IP ROUTEMap=routemap ENTRY=1..4294967295
  [Action=INCLUDE] SET MED={0..4294967295|REMOVE}

SET IP ROUTEMap=routemap ENTRY=1..4294967295
  [Action=INCLUDE] SET METRIC=0..4294967295

SET IP ROUTEMap=routemap ENTRY=1..4294967295
  [Action=INCLUDE] SET ORIGIN={IGP|EGP|INCOMPLETE}

SET IP ROUTEMap=routemap ENTRY=1..4294967295
  [Action={INCLUDE|EXCLUDE}] SET TAG=1..65535

SET IP ROUTEMap=routemap ENTRY=1..4294967295
  [Action=INCLUDE] SET TYPE={1|2}

```

Description This command does one of the following:

- changes the **action** of an entry in a route map
- modifies an entry's **match** clause
- modifies an entry's **set** clause

This command does not create or delete an entry or clause. To create a new entry or clause, use the [add ip routemap command on page 31-49](#). To delete an entry or clause, use the [delete ip routemap command on page 31-59](#).

Parameter table order The following tables list the parameters that apply to both **match** and **set** clauses, then the **match** clauses, then the **set** clauses. The match and set clause tables first list parameters that apply to all routes, then parameters that only apply to OSPF routes, then parameters that only apply to BGP routes.

**Parameters for both
match and set
clauses**

Parameter	Description
ROUTEMap	The name of the route map to which the entry or clause belongs. Default: no default
ENTRY	The ID number of the entry to change. Default: no default
Action	Whether matching prefixes or update messages are included or excluded by the process that is using the route map. The action parameter applies to the entire entry. It is not meaningful to have action=exclude in an entry with a set clause. Default: the current setting. If there is no current setting, include

**Parameters for
match clauses**

Parameter	Description
MATCH	Modifies the match clause in the entry. The match clause determines which routes or BGP update messages the entry applies to. A route map entry can have zero or one match clauses. An entry without a match clause matches all routes or updates.

Parameter (cont.)	Description (cont.)
INterface (all route types)	<p>A switch interface. A route matches the route map entry if its interface matches the specified interface. Valid interfaces are:</p> <ul style="list-style-type: none"> PPP (such as ppp0, ppp1-1) FR (such as fr0, fr0-1) X.25 DTE (such as x25t0, x25t0-1) VLAN (such as vlan1, vlan1-1) <p>To see a list of interfaces currently available, use the show interface command on page 11-87 of Chapter 11, Interfaces.</p> <p>Default: no default</p>
METric (all route types)	<p>The route metric or a range of metric values. A route matches the route map entry if its metric equals this value or is in this range.</p> <p>Default: no default</p>
NEXThop (all route types)	<p>The IP address of the next node in the path to the route's destination, specified in dotted decimal notation. For BGP, an update message matches the route map entry if its next_hop attribute matches this address.</p> <p>Default: no default</p>
PREFIXList (all route types)	<p>The name of a prefix list. A route matches the route map entry if the prefix list contains that route. To create a list use the add ip prefixlist command on page 31-44.</p> <p>Default: no default</p>
TAG (all route types)	<p>A tag that identifies a particular route. A route matches this route map entry if it has been tagged with this value. There are two ways of tagging routes. You can use a route map to set the route's tag, or for static routes you can use the tag parameter of the add ip route command on page 23-80 of Chapter 23, Internet Protocol (IP).</p> <p>For BGP, you can use a route map that matches on tag when you use the add bgp import command to import static routes from the RIB to BGP. However, BGP routes do not have a tag field in their path attributes. Therefore, you cannot use tag to filter routes that are sent to BGP peers or to match update messages that are received from BGP peers.</p> <p>For OSPF, you can use a route map that matches on tag when you use the add ospf redistribute command to import routes from the RIB to OSPF.</p> <p>Default: no default</p>
parameters for OSPF routes only	<p>ROUTEsource (OSPF routes only)</p> <p>The name of a prefix list that lists one or more router IDs. A route matches the route map entry if the prefix list contains the router ID of the router that advertised the route to OSPF.</p> <p>To create a prefix list use the add ip prefixlist command on page 31-44. Note that the mask for a router ID must be 255.255.255.255, so the mask length must be 32.</p> <p>Default: no default</p>

Parameter (cont.)		Description (cont.)
	ROUTEType (OSPF routes only)	The type of route, which indicates whether the route is within the OSPF area, to another area with the same AS, or to another AS. See “Routing with OSPF” on page 29-9 of Chapter 29, Open Shortest Path First (OSPF) for more information about these route types. Default: no default
	INTRA	A route matches the route map entry if it is an OSPF intra-area route.
	INTER	A route matches the route map entry if it is an OSPF inter-area route.
	TYPE1	A route matches the route map entry if it is an OSPF External Type 1 route.
	TYPE2	A route matches the route map entry if it is an OSPF External Type 2 route.
	OTHER	A route matches the route map entry if it is not one of the above route types.
parameters for BGP routes only	ASPath (BGP routes only)	The ID number of an AS path list. An update message matches the route map entry if its AS path attribute matches the AS path list. To configure an AS path list use the add ip aspathlist command on page 31-40 . Default: no default
	COMmunity (BGP routes only)	The ID number of a community list. An update message matches the route map entry if its community attribute matches the community list. To configure a community list use the add ip communitylist command on page 31-42 . Default: no default
	EXAct (BGP routes only)	Whether the community attribute in an update message must precisely match the route map's community list. Only valid when you specify both match and community . Default: no
	YES	An update message only matches the route map entry if its community attribute contains all the communities specified in the community list and only those communities.
	NO	An update message still matches the route map entry if its community attribute contains all the communities specified in the community list plus extra communities.
	MED (BGP routes only)	The update message's Multi_Exit_Discriminator attribute. EBGp uses the MED to determine the optimal path for reaching the advertised prefixes. A lower metric indicates a preferred path. IBGP does not use this attribute. An update message matches the route map entry if its MED attribute matches this value. Default: no default

Parameter (cont.)	Description (cont.)
ORIGin (BGP routes only)	An origin attribute value, which indicates BGP's source for the routes at their originating AS. An update message matches the route map entry if its origin attribute matches this value. Default: no default
IGP	The original source of the route was IGP.
EGP	The original source of the route was EGP.
INCOmplete	The original source of the route was neither IGP or EGP. This includes statically-configured routes.

Parameters for set clauses

Parameter	Description
SET	Modifies a set clause in the entry. These clauses modify characteristics of routes that match the entry or, for BGP routes, an attribute in update messages that match the entry. A route map entry can have zero, one or more set clauses, but can only modify each attribute once. An entry without a set clause does not modify any attributes.
METric (all route types)	The metric to give to the route. For RIP, values higher than 16 are treated as if they are 16. Default: no default
TAG (all route types)	A number to label matching routes with. Tagging routes allows you to identify the route's original source, for example, in the output of the show ip route command. For example, you can tag BGP routes when you redistribute them into OSPF or RIP, which enables you to identify the routes that came from BGP. Default: no default
TyPe (OSPF routes only)	The OSPF external route type to set the route to. This enables you to ensure that all externally-sourced OSPF routes are the same type and therefore use the same method to calculate route metrics. Default: no default
ASPath (BGP routes only)	A comma-separated list of 1 to 10 AS numbers. These numbers are added to the beginning of the update message's AS path attribute. Default: no default

Parameter (cont.)	Description (cont.)
COMmunity (BGP routes only)	<p>A comma-separated list of 1 to 10 communities, identified by name or number. If the add parameter is yes, these communities are added to the update message's community attribute. If the add parameter is no (its default), these communities replace the update message's community attribute.</p> <p>Note that you must also set the peer's sendcommunity parameter to yes if you want the peer to include the community attribute in the update messages it sends. By default, peers do not include the community attribute in outgoing updates.</p> <p>Default: no default</p>
INTernet	The community of routes that can be advertised to all BGP peers.
NOExport	The community of routes that must not be advertised outside a BGP confederation boundary (a standalone autonomous system that is not part of a confederation should be considered a confederation itself).
NOAdvertise	The community of routes that must not be advertised to other BGP peers.
NOEXPORTSubconfed	The community of routes that must not be advertised to external BGP peers (this includes peers in other members' autonomous systems inside a BGP confederation).
aa:xx	The number of a community. aa and xx are both integers from 0 to 65534. aa is the AS number. xx is a value chosen by the ASN administrator.
ADD (BGP routes only)	<p>Whether the list of communities specified by the community parameter is added to the community attribute, or replaces the community attribute. Only valid when you specify both set and community.</p> <p>Default: no</p>
YES	The communities are added to the update message's community attribute.
NO	The communities replace the update message's community attribute.
BGPDampid (BGP routes only)	<p>The BGP route flap damping ID that is given to matching routes. This is the same as the ID number of the parameter set that maintains that route's FoM upon it exhibiting instability. If the parameter set does not exist, the default parameter set is applied to matching routes.</p> <p>For more information about using route maps when configuring route flap damping, see "Damping routes on specific peers" on page 30-27 of Chapter 30, Border Gateway Protocol version 4 (BGP-4).</p> <p>Default: no default</p>
LOCalpref (BGP routes only)	<p>The metric to write into the update message's local_preference attribute. IBGP uses the local preference to determine which path it should use inside the AS to reach the advertised prefix. A lower metric indicates a preferred path. EBGP does not use this attribute.</p> <p>Default: no default</p>

Parameter (cont.)	Description (cont.)
MED (BGP routes only)	The metric to write into the update message's Multi_Exit_Discriminator attribute. EBGp uses the MED to determine the optimal path for reaching the advertised prefixes. A lower metric indicates a preferred path. IBGP does not use this attribute. Default: no default
	0..4294967295 This value is written into the MED attribute of the matched update message.
	REMOVE The MED attribute is removed from the matched update message.
ORIGin (BGP routes only)	The value to write into the update message's origin attribute. The origin indicates BGP's source for the routes at their originating AS. Default: no default
	IGP The original source of the route was IGP.
	EGP The original source of the route was EGP.
	INComplete The original source of the route was neither IGP or EGP. This includes statically-configured routes.

Examples To change a route map entry number 10 so that it selects all routes with an OSPF metric from 5 to 15, use the command:

```
set ip routemap=metric_ent=10 ma met=5-15
```

To change the MED for an existing set MED clause in entry 10 of the route map called set_med, use the command:

```
set ip routemap=set_med ent=10 set med=234
```

Related Commands

- [add ip routemap](#)
- [delete ip routemap](#)
- [show ip routemap](#)

show ip aspathlist

Syntax `SHOW IP ASPATHlist[=1..99]`

Description This command displays information about a specific AS path list or all lists in the switch ([Figure 31-2](#), [Table 31-3](#)).

Figure 31-2: Example output from the **show ip aspathlist** command

IP AS path lists		
List	Entry	Regular expression

1	1	Include ^\$
	2	Exclude .*

34	1	Exclude ^123
	2	Include 345 234.+123
	3	Exclude .*

Table 31-3: Parameters in output of the **show ip aspathlist** command

Parameter	Meaning
List	AS path list number from 1 to 99.
Entry	Entry in the AS path list from 1 to the number of entries in the list.
Regular expression	AS path regular expression for this entry. This is preceded by "exclude" or "include" to indicate what the switch does when there is a match. For a description of regular expressions, see Table 31-1 on page 31-10 .

Examples To display AS path list number 23, use the command:

```
sh ip aspath=23
```

Related Commands [add ip aspathlist](#)
[delete ip aspathlist](#)

show ip communitylist

Syntax SHOW IP COMMunitylist[=1..99] [OLDcommunityformat]

Description This command displays information about a specific community list or all lists in the switch (Figure 31-3, Table 31-4).

The **communitylist** parameter specifies the community list to display. If a list is not specified, all are displayed.

The **oldcommunityformat** parameter specifies that community numbers are displayed in the old format. This is an integer calculated by:

$$\text{AS number} \times 65536 + \text{community value}$$

Figure 31-3: Example output from the **show ip communitylist** command

IP community lists		
List	Entry	Community list
1	1	Include noexport,1234:2345
	2	Exclude 34567:123
23	1	Exclude 12:34
	2	Include internet

Table 31-4: Parameters in output of the **show ip communitylist** command

Parameter	Meaning
List	Number of community list from 1 to 99.
Entry	Entry in the community list from 1 to the number of entries in the list.
Community list	The community list for this entry, preceded by "exclude" or "include" to indicate whether a match means that the community attribute should be excluded or included in the function for which the community list is being used.

Examples To display all IP community lists, use the command:

```
sh ip com
```

Related Commands [add ip communitylist](#)
[delete ip communitylist](#)

show ip prefixlist

Syntax `SHoW IP PREFIXList [=name]`

Description This command displays information about prefix lists on the switch. If you specify a prefix list name, detailed information about that prefix list and its entries is displayed (Figure 31-5, Table 31-6). Otherwise, summary information about all existing prefix lists is displayed (Figure 31-4, Table 31-5).

Figure 31-4: Example summary output from the **show ip prefixlist** command

IP Prefix Lists		
Name	Entries	In Use

Sample	11	Yes
Test	3	No

Table 31-5: Parameters in output of the **show ip prefixlist** command

Parameter	Meaning
Name	The name of the prefix list.
Entries	The number of entries in the prefix list.
In Use	Whether the prefix list is currently assigned to a route map.

Figure 31-5: Example detailed output from the **show ip prefixlist** command

IP Prefix List

Name Sample

In Use Yes

Entries:

Number	Action	Prefix	Length	Range
1	Match	192.168.0.0		16
3	No Match	0.0.0.0		25-30
10	No Match	10.10.10.0		24-30

Table 31-6: Parameters in detailed output of the **show ip prefixlist** command

Parameter	Meaning
Name	Name of the prefix list.
In Use	Whether the prefix list is currently assigned to a route map.
Number	The entry number of the prefix list entry. The switch checks entries in order, starting with the lowest entry number.
Action	Whether the prefix list includes ("match") or excludes ("nomatch") any prefix that is within the entry's prefix range.
Prefix	IP network address for the entry to match on.
Length Range	Range of CIDR mask lengths that the entry can match on.

Examples To see the entries in prefix list “office”, use the command:

```
sh ip prefixl=office
```

Related Commands

- [add ip prefixlist](#)
- [add ip routemap](#)
- [delete ip prefixlist](#)
- [set ip routemap](#)

show ip route filter

Syntax SHow IP ROuTe FiLter

Description This command displays information about configured IP route filters ([Figure 31-6](#), [Table 31-7](#)).

Figure 31-6: Example output from the **show ip route filter** command

IP Route Filters					
Ent.	IP Address Protocol	Mask Direction	Nexthop Interface	Policy Action	Matched
1	0.0.0.0 RIP	0.0.0.0 Both	Any -	0 Include	0
Request: 1		Passes: 1		Fails: 0	

Table 31-7: Parameters in output of the **show ip route filter** command

Parameter	Meaning
Ent.	Filter number.
IP Address	IP address of the network that is filtered.
Mask	Network mask for the network address.
Nexthop	Next hop to which the filter applies.
Policy	Policy or type of service to which the filter applies.
Matched	Number of times this pattern has been matched.
Protocol	Routing protocol to which the filter applies.
Direction	Whether the filter applies to routes the switch receives, advertises, or both.
Interface	Interface to which the filter applies.
Action	Whether matching routes are included, excluded, or copied to the switch’s hardware default IP route table.

Related Commands

- [add ip route filter](#)
- [delete ip route filter](#)
- [set ip route filter](#)

show ip routemap

Syntax `SHoW IP ROUTEMap[=routemap] [OLDcommunityformat]`

where *routemap* is a character string 0 to 15 characters long. Valid characters are uppercase and lowercase letters, digits (0-9), and the underscore.

Description This command displays information about all IP route maps or a specific one (Figure 31-7, Table 31-8).

The **routemap** parameter specifies the name of the route map to display. If one is not specified, information about all route maps is displayed.

The **oldcommunityformat** parameter specifies that community numbers are displayed in the old format. This is an integer calculated by:

$$\text{AS number} \times 65536 + \text{community value}$$

Figure 31-7: Example output from the **show ip routemap** command

```

IP route maps

Map name
  Entry      Action
    Clauses
-----
bgp
  1          Include
    match    Community    12 Exact=no
    set      LocalPref    3245
    set      Med           8726
    set      Origin        incomplete
  12345      Include
    set      Community    12 noadvertise Add=yes
  4294967295 Include
    set      AS-path      44
    set      Local Pref   3245
    set      Med           8762
    set      Origin        igp
-----
ospf
  1          Include
    match    Interface    vlan2
    set      Metric        234
-----

```

Table 31-8: Parameters in output of the **show ip routemap** command

Parameter	Meaning
Map name	Name of the route map.
Entry	Entry number for the route map entry. Entry numbers can be any number, but all entries within a route map are sorted by entry number.
Action	Whether the action for this route map entry is include or exclude.

Table 31-8: Parameters in output of the **show ip routemap** command (cont.)

Parameter	Meaning
Clauses	The match and set clauses for this route map entry. Each entry can have only one match clause, and only one set clause of a given type. For information about the clauses, see the add ip routemap command on page 31-49 .

Examples To display the IP route map with the name “import_static_map”, use the command:

```
sh ip routem=import_static_map
```

Related Commands [add ip routemap](#)
[delete ip routemap](#)
[set ip routemap](#)