Defining Route Maps

This chapter describes route maps and includes the following sections:

Information About Route Maps

Route maps are used when redistributing routes into an OSPF, RIP, or EIGRP routing process. They are also used when generating a default route into an OSPF routing process. A route map defines which of the routes from the specified routing protocol are allowed to be redistributed into the target routing process.

Route maps have many features in common with widely known ACLs. These are some of the traits common to both:

- They are an ordered sequence of individual statements, each has a permit or deny result. Evaluation of ACL or route maps consists of a list scan, in a predetermined order, and an evaluation of the criteria of each statement that matches. A list scan is aborted once the first statement match is found and an action associated with the statement match is performed.
- They are generic mechanisms—Criteria matches and match interpretation are dictated by the way that they are applied. The same route map applied to different tasks might be interpreted differently.

These are some of the differences between route maps and ACLs:

- Route maps frequently use ACLs as matching criteria.
- The main result from the evaluation of an access list is a yes or no answer—An ACL either permits or denies input data. Applied to redistribution, an ACL determines if a particular route can (route matches ACLs permit statement) or can not (matches deny statement) be redistributed. Typical route maps not only permit (some) redistributed routes but also modify information associated with the route, when it is redistributed into another protocol.
- Route maps are more flexible than ACLs and can verify routes based on criteria which ACLs can not verify. For example, a route map can verify if the type of route is internal.

- Each ACL ends with an implicit deny statement, by design convention; there is no similar convention for route maps. If the end of a route map is reached during matching attempts, the result depends on the specific application of the route map. Fortunately, route maps that are applied to redistribution behave the same way as ACLs: if the route does not match any clause in a route map then the route redistribution is denied, as if the route map contained deny statement at the end.

The dynamic protocol **redistribute** command allows you to apply a route map. In ASDM, this capability for redistribution can be found when you add or edit a new route map (see the "Defining a Route Map" section on page 1-4). Route maps are preferred if you intend to either modify route information during redistribution or if you need more powerful matching capability than an ACL can provide. If you simply need to selectively permit some routes based on their prefix or mask, we recommends that you use a route map to map to an ACL (or equivalent prefix list) directly in the **redistribute** command. If you use a route map to selectively permit some routes based on their prefix or mask, you typically use more configuration commands to achieve the same goal.

Note You must use a standard ACL as the match criterion for your route map. Using an extended ACL will not work, and your routes will never be redistributed. We recommend that you number clauses in intervals of 10, to reserve numbering space in case you need to insert clauses in the future.

This section includes the following topics:

Permit and Deny Clauses

Route maps can have permit and deny clauses. In the **route-map ospf-to-eigrp** command, there is one deny clause (with sequence number 10) and two permit clauses. The deny clause rejects route matches from redistribution. Therefore, the following rules apply:

- If you use an ACL in a route map using a permit clause, routes that are permitted by the ACL are redistributed.
- If you use an ACL in a route map deny clause, routes that are permitted by the ACL are not redistributed.
- If you use an ACL in a route map permit or deny clause, and the ACL denies a route, then the route map clause match is not found and the next route-map clause is evaluated.

Match and Set Clause Values

Each route map clause has two types of values:

- A match value **s**elects routes to which this clause should be applied.
- A set value modifies information that will be redistributed into the target protocol.

For each route that is being redistributed, the router first evaluates the match criteria of a clause in the route map. If the match criteria succeed, then the route is redistributed or rejected as dictated by the permit or deny clause, and some of its attributes might be modified by the values set from the Set Value tab in ASDM or from the **set** commands. If the match criteria fail, then this clause is not applicable to the route, and the software proceeds to evaluate the route against the next clause in the route map. Scanning of the route map continues until a clause is found whose **match** command(s), or Match Clause as set from the Match Clause tab in ASDM, match the route or until the end of the route map is reached.

A match or set value in each clause can be missed or repeated several times, if one of these conditions exists:

- If several **match** commands or Match Clause values in ASDM are present in a clause, all must succeed for a given route in order for that route to match the clause (in other words, the logical AND algorithm is applied for multiple match commands).
- If a **match** command or Match Clause value in ASDM refers to several objects in one command, either of them should match (the logical OR algorithm is applied). For example, in the **match ip address 101 121** command, a route is permitted if access list 101 or access list 121 permits it.
- If a **match** command or Match Clause value in ASDM is not present, all routes match the clause. In the previous example, all routes that reach clause 30 match; therefore, the end of the route map is never reached.
- If a **set** command, or Set Value in ASDM, is not present in a route map permit clause, then the route is redistributed without modification of its current attributes.

Note Do not configure a **set** command in a route map deny clause because the deny clause prohibits route redistribution—there is no information to modify.

A route map clause without a **match** or **set** command, or Match or Set Value as set on the Match or Set Value tab in ASDM, performs an action. An empty permit clause allows a redistribution of the remaining routes without modification. An empty deny clause does not allows a redistribution of other routes (this is the default action if a route map is completely scanned, but no explicit match is found).

Licensing Requirements for Route Maps

The following table shows the licensing requirements for route maps:

Model	License Requirement
All models	Base License.

Guidelines and Limitations

This section includes the guidelines and limitations for this feature.

Context Mode Guidelines

Supported in single context mode and multiple context mode.

Firewall Mode Guidelines

Supported only in routed firewall mode. Transparent firewall mode is not supported.

IPv6 Guidelines

Does not support IPv6.

Additional Guidelines

Route maps do not support access lists that include a user, user group, or fully qualified domain name objects.

Defining a Route Map

You must define a route map when specifying which of the routes from the specified routing protocol are allowed to be redistributed into the target routing process.

To define a route map, enter the following command:

Command	Purpose
route-map *name* {**permit** | **deny**} [*sequence_number*] **Example:** `hostname(config)# route-map name {permit} [12]`	Creates the route map entry. Enters route-map configuration mode. Route map entries are read in order. You can identify the order using the *sequence_number* argument, or the ASA uses the order in which you add route map entries.

Customizing a Route Map

This section describes how to customize the route map and includes the following topics:

- Defining a Route to Match a Specific Destination Address, page 1-4
- Configuring the Metric Values for a Route Action, page 1-5

Defining a Route to Match a Specific Destination Address

To define a route to match a specified destination address, perform the following steps:

Detailed Steps

	Command	Purpose
Step 1	**route-map** *name* {**permit** | **deny**} [*sequence_number*] **Example:** `hostname(config)# route-map name {permit} [12]`	Creates the route map entry. Enters route-map configuration mode. Route map entries are read in order. You can identify the order using the *sequence_number* option, or the ASA uses the order in which you add route map entries.
Step 2	Enter one of the following **match** commands to match routes to a specified destination address:	

Command	Purpose
match ip address *acl_id* [*acl_id*] [...] [prefix-list] **Example:** hostname(config-route-map)# match ip address *acl_id* [*acl_id*] [...]	Matches any routes that have a destination network that matches a standard ACL or prefix list. If you specify more than one ACL, then the route can match any of the ACLs.
match metric *metric_value* **Example:** hostname(config-route-map)# match metric *200*	Matches any routes that have a specified metric. The *metric_value* can range from 0 to 4294967295.
match ip next-hop *acl_id* [*acl_id*] [...] **Example:** hostname(config-route-map)# match ip next-hop *acl_id* [*acl_id*] [...]	Matches any routes that have a next hop router address that matches a standard ACL. If you specify more than one ACL, then the route can match any of the ACLs.
match interface *if_name* **Example:** hostname(config-route-map)# match interface *if_name*	Matches any routes with the specified next hop interface. If you specify more than one interface, then the route can match either interface.
match ip route-source *acl_id* [*acl_id*] [...] **Example:** hostname(config-route-map)# match ip route-source *acl_id* [*acl_id*] [...]	Matches any routes that have been advertised by routers that match a standard ACL. If you specify more than one ACL, then the route can match any of the ACLs.
match route-type {**internal** | **external** [**type-1** | **type-2**]} **Example:** hostname(config-route-map)# match route-type internal type-1	Matches the route type.

Configuring the Metric Values for a Route Action

If a route matches the **match** commands, then the following **set** commands determine the action to perform on the route before redistributing it.

To configure the metric value for a route action, perform the following steps:

Detailed Steps

	Command	Purpose
Step 1	**route-map** *name* {**permit** | **deny**} [*sequence_number*] **Example:** hostname(config)# route-map *name* {permit} [*12*]	Creates the route map entry. Enters route-map configuration mode. Route map entries are read in order. You can identify the order using the *sequence_number* argument, or the ASA uses the order in which you add route map entries.
Step 2	To set a metric for the route map, enter one or more of the following **set** commands:	
	set metric *metric_value* **Example:** hostname(config-route-map)# set metric *200*	Sets the metric value. The *metric_value* argument can range from 0 to 294967295.
	set metric-type {**type-1** | **type-2**} **Example:** hostname(config-route-map)# set metric-type type-2	Sets the metric type. The *metric-type* argument can be type-1 or type-2.

Configuration Example for Route Maps

The following example shows how to redistribute routes with a hop count equal to 1 into OSPF.

The ASA redistributes these routes as external LSAs with a metric of 5 and a metric type of Type 1.

```
hostname(config)# route-map 1-to-2 permit
hostname(config-route-map)# match metric 1
hostname(config-route-map)# set metric 5
hostname(config-route-map)# set metric-type type-1
```

The following example shows how to redistribute the 10.1.1.0 static route into eigrp process 1 with the configured metric value:

```
hostname(config)# route outside 10.1.1.0 255.255.255.0 192.168.1.1
hostname(config-route-map)# access-list mymap2 line 1 permit 10.1.1.0 255.255.255.0
hostname(config-route-map)# route-map mymap2 permit 10
hostname(config-route-map)# match ip address mymap2
hostname(config-route-map)# router eigrp 1
hostname(config)# redistribute static metric 250 250 1 1 1 route-map mymap2
```

Feature History for Route Maps

Table 1-1 lists each feature change and the platform release in which it was implemented.

Table 1-1 Feature History for Route Maps

Feature Name	Platform Releases	Feature Information
Route maps	7.0(1)	We introduced this feature. We introduced the following command: **route-map**.
Enhanced support for static and dynamic route maps	8.0(2)	Enhanced support for dynamic and static route maps was added.
Support for Stateful Failover of dynamic routing protocols (EIGRP, OSPF, and RIP) and debugging of general routing-related operations	8.4(1)	We introduced the following commands: **debug route**, **show debug route**. We modified the following command: **show route**.
Dynamic Routing in Multiple Context Mode	9.0(1)	Route maps are supported in multiple context mode.