- Route Filtering and Route-Maps -

**Prefix-Lists**

Prefix-lists are used to match *routes* as opposed to *traffic*. Two things are matched:

- The prefix (the *network* itself)
- The prefix-length (the length of the *subnet mask*)

Consider the following prefix-list:

```
Router(config)# ip prefix-list MYLIST 10.1.1.0/24
```

The above *prefix-list* matches the 10.1.1.0/24 network exactly. It will not match 10.1.0.0/16, or 10.1.1.4/30.

A **range** of prefix-lengths can be specified:

```
Router(config)# ip prefix-list MYLIST 10.1.1.0/24 le 30
Router(config)# ip prefix-list MYLIST 10.1.1.0/24 ge 26 le 30
```

The first command dictates that the first 24 bits of the prefix must match (meaning, the prefix *must* begin 10.1.1), and the subnet mask must be less than or equal to 30 bits.

The second command dictates again that the first 24 bits of the prefix must match, and the subnet mask must be between 26 to 30 bits (or equal to).

To match all prefixes:

```
Router(config)# ip prefix-list MYLIST 0.0.0.0/0 le 32
```

To view information about all prefix lists:

```
Router# show ip prefix-list detail
```
Distribute-Lists

Distribute-lists are used to filter routing updates, either inbound or outbound. Routes must first be matched using an access-list or prefix-list, and then applied using a distribute-list under the routing process:

To use an access-list to identify routes:

```
Router(config)# access-list 10 permit ip 172.16.0.0 0.0.255.255
Router(config)# router rip
Router(config-router)# distribute-list 10 in serial0/0
```

The above distribute-list will control routes sent inbound on serial0/0. Specifically, the referenced access-list will only permit routes matching 172.16 in the first two octets.

To use a prefix-list to identify routes:

```
Router(config)# ip prefix-list MYLIST 10.1.0.0/16
Router(config)# router rip
Router(config-router)# distribute-list prefix MYLIST out fastethernet0/0
```

The above distribute-list will control routes sent outbound on fastethernet0/0. Specifically, the referenced prefix-list will only match the exact 10.1.0.0/16 route.
Route-Maps

Route-maps are advanced access-lists that serve several functions on IOS devices, including (but not limited to):

- Controlling redistribution between routing protocols.
- Adjusting the attributes of routes (especially for BGP).
- Implementing Policy Based Routing (PBR).

As with access-lists, route-maps are organized as a sequential set of rules or statements, each with a permit or deny condition. However, access-lists can merely permit or deny traffic, while a route-map can additionally modify or perform a specific action on traffic.

Route-maps follow a very simple logic:

- Traffic must be first matched, based on specified criteria.
- A particular attribute or action is set on the matched traffic.

Each statement in a route-map is assigned a sequence number, and contains a series of match and set statements. The route-map is parsed from the lowest sequence number to the highest, and will stop once a match is found.

The following demonstrates the syntax of a route-map:

```
Router(config)# access-list 1 permit 10.1.1.0 0.0.0.255
Router(config)# route-map MYMAP permit 10
Router(config-route-map)# match ip address 1
Router(config-route-map)# set ip next-hop 192.168.1.1
```

First, an access-list was created that matched traffic from 10.1.1.0/24.

Then, a route-map called MYMAP was created, and assigned a sequence number of 10 with a permit condition. If a route-map contains multiple statements, the sequence number dictates the order of those statements.

The route-map will then match any traffic listed in access-list 1. Notice that the syntax to call an access-list match ip address.

Finally, the desired attributed is set to this traffic. In this instance, the ip next hop attribute has been modified to 192.168.1.1.
Route-Maps (continued)

A single route-map statement can contain multiple match commands:

```
Router(config)# route-map MYMAP permit 10
Router(config-route-map)# match ip address 1 2 3
```

The above line would match traffic in access-list 1, or access-list 2, or access-list 3. Thus, when match criteria is contained within a single line, a logical OR is applied.

However, if match criteria is specified on separate lines:

```
Router(config-route-map)# match ip address 1
Router(config-route-map)# match ip address 2
```

Then the traffic must match access-list 1 and access-list 2 (a logical AND). **Remember this distinction!**

If no match criteria is specified, all traffic is matched!

Additionally, a single route-map statement can contain multiple set commands:

```
Router(config)# route-map MYMAP permit 10
Router(config-route-map)# match ip address 1
Router(config-route-map)# set weight 50
Router(config-route-map)# set local-preference 200
```

Any traffic matching access-list 1 will have both set attributes applied.

There is an implicit deny any statement at the bottom of every route-map. The impact of this deny any statement is dependent on the function of the access-list:

- If using a route-map for policy-based routing or adjusting attributes, any routes/traffic not specifically matched will remain unchanged.
- If using a route-map for redistribution, any routes not specifically matched (and permitted) will not be redistributed.

**Route-Map Criteria**

The following are example attributes that can be **matched** by a route-map:

- **match ip address**
- **match interface**
- **match ip address prefix-list**
- **match ip next-hop**
- **match metric**
- **match route-type**
- **match tag**
- **match community**

```
Router(config)# route-map MYMAP permit 10
```

```
Router(config-route-map)# match ip address 1
```

```
Router(config-route-map)# match interface serial0/0
```

```
Router(config-route-map)# match ip address prefix-list MYLIST
```

```
Router(config-route-map)# match ip next-hop 192.168.1.2
```

```
Router(config-route-map)# match metric 40
```

```
Router(config-route-map)# match route-type internal
```

```
Router(config-route-map)# match tag 33
```

```
Router(config-route-map)# match community 123
```

The following are example attributes that can be **set** by a route-map:

- **set interface**
- **set ip next-hop**
- **set metric**
- **set tag**
- **set community**
- **set local-preference**
- **set weight**
- **set ip precedence**

```
Router(config)# route-map MYMAP permit 10
```

```
Router(config-route-map)# set interface fastethernet0/1
```

```
Router(config-route-map)# set ip next-hop 10.1.1.1
```

```
Router(config-route-map)# set metric 200
```

```
Router(config-route-map)# set tag 44
```

```
Router(config-route-map)# set community 321
```

```
Router(config-route-map)# set local-preference 250
```

```
Router(config-route-map)# set weight 300
```

```
Router(config-route-map)# set ip precedence 2
```
**Route-Map Examples**

The following route-map is applying a BGP attribute to a specific route:

```
Router(config)# access-list 1 permit 10.1.1.0 0.0.0.255
Router(config)# route-map MYMAP permit 10
Router(config-route-map)# match ip address 1
Router(config-route-map)# set metric 100
Router(config-route-map)# route-map MYMAP permit 20
```

```
Router(config)# router bgp 100
Router(config-router)# neighbor 172.16.1.1 route-map MYMAP out
```

The following route-map is controlling routes being redistributed between routing protocols:

```
Router(config)# access-list 1 deny 192.168.1.0 0.0.255
Router(config)# access-list 1 deny 192.168.2.0 0.0.255
Router(config)# access-list 1 permit any

Router(config)# route-map MYMAP permit 10
Router(config-route-map)# match ip address 1
Router(config-route-map)# set tag 150
```

```
Router(config)# router ospf 1
Router(config-router)# redistribute eigrp 10 metric 3 subnets route-map MYMAP
```

The following route-map is manipulating inbound traffic on a specific interface:

```
Router(config)# access-list 1 permit 10.1.1.0 0.0.0.255

Router(config)# route-map MYMAP permit 10
Router(config-route-map)# match ip address 1
Router(config-route-map)# set ip next-hop 192.168.1.1
```

```
Router(config)# interface s0/0
Router(config-if)# ip policy route-map MYMAP
```