Chapter 11
Router Discovery

address

Name

address - specifies parameters for logical addresses

Syntax

```
address interface_list
    [ advertise | ignore ]
    [ broadcast | multicast ]
    [ ineligible | preference preference ]
;
```

Note: At least one option for **address** must be specified for the command to parse.

Parameters

interface_list - specifies the set of addresses to which the other parameters apply

advertise - specifies that the addresses in the *interface_list* should be included in router advertisements

ignore - specifies that the addresses in the *interface_list* should not be included in router advertisements

broadcast - specifies that the addresses in the *interface_list* should be included in broadcast router advertisements

multicast - specifies that the addresses in the *interface_list* should be included in multicast router advertisements

preference *preference* - specifies that the addresses in the *interface_list* should be advertised with a preference of *preference*. Preference is a 32-bit, signed, two's complement integer, with higher values being more preferable.

ineligible - specifies that the addresses in the *interface_list* should be advertised with the worst possible preference, making them ineligible to become the default routers

Description

The **`address`** statement is used to specify the set of addresses that should be advertised, and the parameters with which they should be specified.

If an interface does not support multicast, but multicast is specified for that interface in the **`routerdiscovery`** clause, then that interface will not be advertised at all.

Default

```
address interface_list
    advertise
    multicast
    preference 0
;
```

Context

`routerdiscovery server` statement

Examples

```
routerdiscovery server on {
    address 223.1.25.3 ignore ;
    address 221.3.5.8 preference 10 ;
} ;
```

See Also

"Chapter 15 Router Discovery" on page 85 of *Configuring GateD*

`routerdiscovery server` on page 308

interface (client)

Name

`interface` - specifies the physical interfaces on which the router discovery client is to run

Syntax

```
interface phys_interface_list
    [ enable | disable ]
    [ multicast | broadcast ]
    [ quiet | solicit ]
;
```

Parameters

`phys_interface_list` - specifies the set of physical interfaces to which the rest of the parameters apply

`enable` - specifies that router discovery should be performed on the specified interfaces

`disable` - specifies that router discovery should not be performed on the specified interfaces

`multicast` - specifies that router discovery should be performed on the specified interfaces, unless disabled, and that router solicitations should be multicast rather than broadcast

`quiet` - specifies that no router solicitations should be sent on these interfaces even though router discovery will still be performed

`solicit` - specifies that initial router solicitations will be sent on these interfaces

Description

The **`interface`** statement specifies the set of interfaces on which the router discovery client should be run. It further allows the specification of how the protocol should be run on the given interfaces. If no interfaces are cited in the **`routerdiscovery client`** clause, the router discovery client will be **`on`** for all interfaces. If, however, any interfaces are cited within the **`routerdiscovery client`** clause, router discovery will be **`on`** for only those interfaces.

Defaults

```
interface all enable
    enable
    multicast
;
```

Context

`routerdiscovery client` statement

Examples

The following example runs router discovery on the interface fxp0, sending solicitations to the multicast address:

```
routerdiscovery client on {
        interface fxp0 enable multicast solicit;
};
```

See Also

“Chapter 15 Router Discovery” on page 85 of *Configuring GateD*

`routerdiscovery client` on page 306

interface (server)

Name

`interface` - specifies the physical interfaces on which to run the server

Syntax

```
interface phys_interface_list
    [ maxadvinterval max_time ]
    [ minadvinterval min_time ]
    [ lifetime life_time ]
;
```

Parameters

`phys_interface_list` - specifies a list of physical interfaces to run the router discovery server on. **`all`** can be used to specify all interfaces.

`maxadvinterval` *`max_time`* - specifies the maximum time allowed between sending broadcast or multicast router advertisements from the interface. This must be no less than 4 seconds and no more than 30 minutes. The default is 10 minutes.

`minadvinterval` *`min_time`* - specifies the minimum time allowed between sending unsolicited broadcast or multicast router advertisements from the interface. This must be no less than 3 seconds and no more than **`maxadvinterval`**. If **`minadvinterval`** is set to greater than **`maxadvinterval`**, GateD will reconfigure the value of **`minadvinterval`** to be equal to the value of **`maxadvinterval`** and log the change. The default is 0.75 * **`maxadvinterval`**.

`lifetime` *`life_time`* - specifies how long the addresses in a given router advertisement are valid. Lifetime must be no less than **`maxadvinterval`** and no more than 2 hours and 30 minutes. If **`lifetime`** is set to less than **`maxadvinterval`**, GateD will reconfigure **`lifetime`** to be equal to **`maxadvinterval`**. The default is 3 * **`maxadvinterval`**.

Description

The **`interface`** statement specifies the set of physical interfaces on which the router discovery server is to run. It also allows specification of various timers associated with the physical interfaces. Note a slight difference in convention from the rest of GateD: **`interface`** specifies a list of physical interfaces (such as le1, ef0, and en1), while **`address`** specifies a list of IP addresses.

If no interfaces are cited in the **`routerdiscovery server`** clause, the router discovery server will be **`on`** for all interfaces. If, however, any interfaces are cited within the **`routerdiscovery server`** clause, routerdiscovery will be **`on`** for only those interfaces.

Default

```
interface all
    ( maxadvinterval 00:10:00
    minadvinterval .75*max_time
```

```
          lifetime 3*max_time )
      ;
```

Context

`routerdiscovery server` statement

Examples

The following example runs the router discovery server on interface fxp0, sending advertisements no more often than once every minute, and no less often than once every 6 minutes. All routers that it advertises out interface fxp0 will be advertised with a lifetime of 10 minutes.

```
     routerdiscovery server on {
             interface fxp0 minadvinterval 1:00 maxadvinterval 6:00
                 lifetime 10:00;
     }
```

See Also

"Chapter 15 Router Discovery" on page 85 of *Configuring GateD*

`routerdiscovery server` on page 308

preference

Name

`preference` - specifies the preferability of the address as a default router address, relative to other router addresses on the same subnet

Syntax

`preference` *`preference`*

Parameters

`preference` - the *`preference`* value to be used for routes learned from the router discovery protocol

Description

The **`preference`** command specifies the preference value to be used when comparing routes learned via the router discovery protocol against routes learned via other protocols. Higher values are more preferred.

Defaults

`preference 0`

Context

`routerdiscovery client` statement

Examples

The following example runs router discovery on the interface fxp0, sending solicitations to the multicast address:

```
routerdiscovery client on {
        preference 3 ;
        interface fxp0 enable multicast solicit ;
} ;
```

The following example demonstrates preference as applied to a **`routerdiscovery`** server:

```
routerdiscovery server on {
        address 1.2.3.4
        preference 50 ;
} ;
```

See Also

"Chapter 15 Router Discovery" on page 85 of *Configuring GateD*

`routerdiscovery client` on page 306

routerdiscovery client

Name

`routerdiscovery client` - enables or disables the client portion of the router discovery protocol

Syntax

```
routerdiscovery client ( on | off ) [ {
    traceoptions trace_options ;
    preference preference ;
    interface phys_interface_list
        [ enable | disable ]
        [ multicast | broadcast ]
        [ quiet | solicit ]
    ;
} ] ;
```

Parameters

`on | off` - enables or disables the client portion of the router discovery protocol

`traceoptions` - specifies which information to log for router discovery

`preference` - specifies how router discovery default routes are compared to default routes from other protocols

`interface` - specifies the physical interfaces on which the router discovery client is to run

Description

The Router Discovery Protocol is an IETF standard protocol, RFC 1256, used to inform hosts of the existence of routers. It is intended to be used instead of having hosts wiretap routing protocols, such as RIP. It is used in place of, or in addition to, statically-configured default routes in hosts. RFC 1256 can be found at:
http://ietf.org/rfc/rfc1256.txt

The protocol is split into two portions: the **`server`** portion, which runs on routers, and the **`client`** portion, which runs on hosts. GateD treats these much like two separate protocols, only one of which can be enabled at a time.

Default

```
routerdiscovery client off  [ {
    preference 0 ;
    interface all enable
        enable
        multicast
    ;
} ] ;
```

Context

global statement

Examples

The following example runs router discovery on the interface fxp0, sending solicitations to the multicast address:

```
routerdiscovery client on {
        interface fxp0 enable multicast solicit;
};
```

See Also

“Chapter 15 Router Discovery” on page 85 of *Configuring GateD*

routerdiscovery server

Name

`routerdiscovery server` - enables or disables the server portion of the router discovery protocol

Syntax

```
routerdiscovery server ( on | off ) [ {
    traceoptions trace_options ;
    interface phys_interface_list
        [ maxadvinterval max_time ]
        [ minadvinterval min_time ]
        [ lifetime life_time ]
    ;
    address interface_list
        [ advertise | ignore ]
        [ broadcast | multicast ]
        [ ineligible | preference preference ]
    ;
} ] ;
```

Parameters

`on | off` - enables or disables the server portion of the router discovery protocol

`traceoptions` - specifies which information to log for router discovery

`address` - specifies parameters for logical addresses

`interface` - specifies the physical interfaces on which the router discovery server is to run

Description

The Router Discovery Protocol is an IETF standard protocol, RFC 1256, used to inform hosts of the existence of routers. It is intended to be used instead of having hosts wiretap routing protocols, such as RIP. It is used in place of, or in addition to, statically-configured default routes in hosts. RFC 1256 can be found at:
http://ietf.org/rfc/rfc1256.txt

The protocol is split into two portions: the **`server`** portion, which runs on routers, and the **`client`** portion, which runs on hosts. GateD treats these much like two separate protocols, only one of which can be enabled at a time.

The router discovery server runs on routers and announces their existence to hosts. The router discovery server announces hosts' existence by periodically multicasting or broadcasting a router advertisement to each interface on which it is enabled. These router advertisements contain a list of all the routers' addresses on a given interface and the preference of each address for use as the default router on that interface.

Initially, these router advertisements occur every few seconds, then fall back to every few minutes. In addition, a host can send a router solicitation to which the router will respond with a unicast router advertisement (unless a multicast or broadcast advertisement is due momentarily).

Each router advertisement contains an advertisement **`lifetime`** field indicating for how long the advertised addresses are valid. This lifetime is configured such that another router advertisement will be sent before the lifetime has expired. A lifetime of zero is used to indicate that one or more addresses are no longer valid.

On systems supporting IP multicasting, the router advertisements are, by default, sent to the all-hosts multicast address, **`224.0.0.1`**. However, the use of **`broadcast`** can be specified. When router advertisements are being sent to the all-hosts multicast address, or an interface is configured for the limited-broadcast address, **`255.255.255.255`**, all IP addresses configured on the physical interface are included in the router advertisement. When the router advertisements are being sent to a net or subnet broadcast, only the address associated with that net or subnet is included.

A host listens for router advertisements via the all-hosts multicast address (**`224.0.0.1`**) if IP multicasting is available and enabled, or on the interface's broadcast address. When starting up, or when reconfigured, a host can send a few router solicitations to the all-routers multicast address, **`224.0.0.2`**, or the interface's broadcast address.

When a router advertisement with non-zero lifetime is received, the host installs a default route to each of the advertised addresses. If the preference is ineligible, or the address is not on an attached interface, the route is marked unusable but is retained. If the preference is usable, the metric is set as a function of the preference such that the route with the best preference is used. If more than one address with the same preference is received, the one with the lowest IP address will be used. These default routes are not exportable to other protocols.

When a router advertisement with a zero lifetime is received, the host deletes all routes with next-hop addresses learned from that router. In addition, any routes learned from ICMP redirects pointing to these addresses will be deleted. The same will happen when a router advertisement is not received to refresh these routes before the lifetime expires.

Default

```
routerdiscovery server off  [ {
    interface all
        ( maxadvinterval 00:10:00
        minadvinterval .75*max_time
        lifetime 3*max_time )
    ;
    address interface_list
        advertise
        multicast
        preference 0
    ;
} ] ;
```

Context

global statement

Examples

The following example runs the router discovery server on interface fxp0, sending advertisements no more often than once every minute, and no less often than once every 6 minutes. All routers that it advertises out interface fxp0 will be advertised with a lifetime of 10 minutes.

```
routerdiscovery server on {
        interface fxp0 minadvinterval 1:00 maxadvinterval 6:00
            lifetime 10:00;
}
```

See Also

“Chapter 15 Router Discovery” on page 85 of *Configuring GateD*

`routerdiscovery client` on page 306

traceoptions

Name

`traceoptions` - specifies which information to log for router discovery

Syntax

`traceoptions` *`traceoptions`* **`;`**

Parameters

`traceoptions`

Description

`traceoptions` specifies the router discovery tracing options. The Router Discovery Client and Server support the **`state`** trace flag, which traces various protocol occurrences.

The Router Discovery Client and Server do not directly support any packet tracing options. Tracing of router discovery packets is enabled via the ICMP Statement. See “Chapter 16 Internet Control Message Protocol (ICMP)” on page 89 in *Configuring GateD* for more information on ICMP.

Defaults

defaults to global traceoptions

Context

`routerdiscovery server` statement

`routerdiscovery client` statement

Examples

The following example turns on all traceoptions for the router discovery server, and logs the information in a file called "rdisc" in the directory /var/tmp.

```
routerdiscovery server on {
        traceoptions "/var/tmp/rdisc" all;
};
```

The following example turns on all traceoptions for the router discovery client, and logs the information in a file called "rdisc" in the directory /var/tmp.

```
routerdiscovery client on {
        traceoptions "/var/tmp/rdisc" all;
};
```

See Also

“Chapter 15 Router Discovery” on page 85 of *Configuring GateD*

“Chapter 4 Trace Statements” on page 15 of *Configuring GateD*

`routerdiscovery client` on page 306