Chapter 8
Open Shortest Path First Protocol (OSPF)

advertise-subnet

Name

`advertise-subnet` - specifies whether OSPF will, when advertising point-to-point interfaces, advertise the network number and netmask of the point-to-point interface instead of a host-route to the remote IP

Syntax

`advertise-subnet on | off ;`

Parameters

`on` | **`off`** - enables/disables the option

Description

`advertise-subnet` specifies whether OSPF will, when advertising point-to-point interfaces, advertise the network number and netmask of the point-to-point interface instead of a host-route to the remote IP. Because sometimes the netmask is set improperly on point-to-point interfaces, the default is **`off`**. The global **`advertise-subnet`** will be the default unless it is overridden in the area or interface statement.

Default

`advertise-subnet off ;`

Context

`ospf` statement

`area` statement

`interface` statement

`virtuallink` statement

Examples

```
ospf yes {
    advertise-subnet on;
```

```
        area 1.2.3.4 {
            interface ppp0 cost 1;
        };
    };
```

See Also

`interface` on page 110

always-update-summary

Name

`always-update-summary -` forces GateD to update summary LSAs

Syntax

`always-update-summary on | off ;`

Parameters

`on | off -` enables/disables this option

Description

If this option is set, summary LSAs whose supporting path type has changed are always regenerated in the SPF computation. This occurs regardless of whether the contents of the LSAs have changed. This flag is provided to disable an optimization that can cause GateD to fail commercially available, component-level tests.

Default

on

Context

`ospf` statement

Examples

```
ospf yes {
    always-update-summary on;
    area 1.2.3.4 {
        interface ppp0 cost 1;
    };
};
```

See Also

`ospf` on page 124

area

Name

`area` - defines an OSPF area

Syntax

`area` *`areanumber`*

Parameters

`areanumber` - dotted-quad area-ID for this area

Description

Each OSPF router must be configured into at least one OSPF area. If more than one area is configured, at least one must be the backbone. The backbone interface can be a virtual link.

Default

The default area configuration (if no areas are configured by the user) is a backbone with all interfaces.

Context

`ospf` statement

Examples

```
ospf yes {
    area 1.2.3.4 {
        interface fxp0 cost 1;
    };
};
```

See Also

`interface` on page 110

`backbone` on page 101

auth

Name

`auth` - specifies the type of authentication and key values

Syntax

`auth [ none | simple` *`auth_key`* **`| md5`** *`md5-keyset`* **`]`**

Parameters

`none` - specifies no authentication

`simple` - specifies simple (clear password) authentication

`md5` - specifies md5 cryptographic authentication

Description

`auth` is used by OSPF authentication to generate and verify the authentication field in the OSPF header. The global authentication will be the default unless it is specified in the area or interface statement. *`auth_key`* is specified by one to eight decimal digits (with a value between 0 and 255) separated by periods, a 1- to 8-byte hexadecimal string preceded by 0x, or a one- to eight- character string in double quotes. See “Authentication” on page 99 of *Configuring GateD* for a more detailed description.

Specify MD5 authentication with the *`md5-keyset`*, which is specified as:

```
key md5-key id id-number [ {
    [start-generate date-time;]
    [stop-generate date-time;]
    [start-accept date-time;]
    [stop-accept date-time;]
}];
```

where *`md5-keyset`* is a one- to 16-character string in double quotes, *`id-number`* is an integer with a value between 1 and 255, and *`date-time`* is in the format YYYY/MM/DD HH:MM. If any time fields are used, all are required.

Default

`auth none ;`

Context

`ospf` statement

`area` statement

`ospf interface` statement

`virtuallink` statement

Examples

Example 1

```
ospf yes {
    auth simple “foobar”;
    backbone {
        interface fxp0 cost 1;
    };
};
ospf yes {
    area 1.2.3.4 {
        auth simple “foo”;
        interface fxp1 cost 1;
    };
    backbone {
        auth simple “bar”;
        interface fxp2 cost 1;
    };
};
```

Example 2

```
ospf yes {
    traceoptions all;
    backbone {
        interface 192.168.32.17 cost 1
        {
            retransmitinterval 5;
            transitdelay 1;
            priority 1;
            hellointerval 10;
            routerdeadinterval 40;
           auth md5 key “key1” id 123 { stop-accept 2000/01/01 12:00; };
        };
    };
};
```

See Also

`interface` on page 110

`area` on page 98

backbone

Name

`backbone` - defines an OSPF backbone area

Syntax

`backbone`

Parameters

`backbone` - defines the backbone area

Description

Each OSPF router must be configured into at least one OSPF area. If more than one area is configured, at least one must be the backbone. The backbone interface may be a virtual link. Note that **`area`** 0.0.0.0 or **`area`** 0 is equivalent to **`backbone`**.

Default

The default area configuration (if no areas are configured by the user) is a backbone with all interfaces.

Context

`ospf` statement

Examples

```
ospf yes {
    backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

`interface` on page 110

`area` on page 98

cost

Name

`cost` - specifies the default cost when importing non-OSPF routes into OSPFASE

Syntax

`cost` *`defasecost`*;

Parameters

`defasecost` - default cost of routes exported into Autonomous System External (ASE)

Description

`cost` is used when exporting a non-OSPF route from the GateD routing table into OSPF as an ASE route. **`cost`** can be explicitly overridden in export policy.

Default

`cost 1 ;`

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        cost 1;
    };
    area 1.2.3.4 {
        interface fxp0;
    };
};
```

See Also

`defaults` on page 103

`nssa-cost` on page 117

defaults

Name

`defaults` - specifies the defaults used when importing OSPF ASE routes or Not So Stubby Area (NSSA) routes into the GateD routing table and when exporting routes from the GateD routing table into OSPF ASE or NSSA

Syntax

`defaults`

Parameters

`preference` *`defasepref`*

`nssa-preference` *`defnssapref`*

`cost` *`defasecost`*

`nssa-cost` *`defnssacost`*

`tag [ as ]` *`tagvalue`*

`type 1 | 2`

`nssa-type 1 | 2`

`inherit-metric`

`nssa-inherit-metric`

`ribs unicast [ multicast ]`

Description

These parameters specify the defaults used when importing OSPF ASE routes or NSSA routes into the GateD routing table and exporting routes from the GateD routing table into OSPF ASEs or NSSAs.

Default

The individual options have the following default values:

```
preference 150;
 nssa-preference 150;
cost 1;
 nssa-cost 1;
tag as 0 ;
type 2;
 nssa-type 2;
ribs unicast;
```

Context

`ospf` statement

Examples

```
ospf  yes {
    defaults {
        preference 10;
        cost 1;
        tag 2112;
        type 1;
        inherit-metric;
        ribs unicast multicast;
    };
    area 1.2.3.4 {
        interface fxp0 cost 1;
    };
};
```

See Also

`preference` on page 127

`cost` on page 102

`tag` on page 145

`type` on page 150

`inherit-metric` on page 109

`ribs` on page 132

`area` on page 98

`nssa-cost` on page 117

`nssa-inherit-metric` on page 118

`nssa-preference` on page 119

`nssa-type` on page 120

`nssa` on page 116

disable

Name

`disable` - disable the interface

Syntax

`disable;`

Parameters

none

Description

This option causes interface(s) matching the policy to be disabled.

Default

This option is disabled by default.

Context

`ospf interface` statement

Examples

```
ospf yes {
    interface fxp cost 1;
    interface fxp0 disable;
};
```

See Also

`ospf interface` on page 110

`enable` on page 106

enable

Name

`enable` - enable the interface

Syntax

```
enable;
```

Parameters

none

Description

This option is provided for symmetry with 'disable'. It is the default.

Default

This option is enabled by default.

Context

`ospf interface` statement

Examples

```
    ospf yes {
        interface fxp0 { enable };
    };
```

See Also

`ospf interface` on page 110

`disable` on page 105

hellointerval

Name

`hellointerval` - the length of time, in seconds, between hello packets that the router sends on an interface

Syntax

`hellointerval` *`time`* **`;`**

Parameters

`time` - the default length of time to use between sending hello packets

Description

`hellointerval` is the length of time, in seconds, between hello packets that the router sends on the interface. This option can be specified at the global level and overridden at the area and interface levels.

Default

`hellointerval 10 ;`

Context

`ospf` statement

`area` statement

`interface` statement

`virtuallink` statement

Examples

```
ospf yes {
    hellointerval 20;
    area 1.2.3.4 {
        interface fxp0 cost 1;
    };
    backbone {
        hellointerval 10;
        interface fxp1 cost 2;
    };
    area 2.3.4.5 {
        hellointerval 30;
        interface fxp2;
    };
    };
```

See Also

`routerdeadinterval` on page 133

`interface` on page 110

inherit-metric

Name

`inherit-metric` - configures an OSPF ASE route to inherit the metric of the external route when no metric is specified on the export policy

Syntax

```
inherit-metric ;
```

Parameters

none

Description

`inherit-metric` allows an OSPF ASE route to inherit the metric of the external route when no metric is specified on the export policy. **`inherit-metric`** maintains compatibility with all the current export functions. A metric specified on the export policy will take precedence. The **`cost`** specified in the **`default`** statement (*`defasecost`*) will be used if **`inherit-metric`** is not specified.

Default

The default is to not inherit the external metric.

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        inherit-metric;
    };
    backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

`defaults` on page 103

`cost` on page 102

`nssa-inherit-metric` on page 118

interface

Name

`interface` - configures an interface to run OSPF

Syntax

```
              interface interface_list [ cost ifcost ] [ {
                  enable | disable ;
                  retransmitinterval iftime ;
                  transitdelay iftime ;
                  priority ifpriority ;
                  hellointerval if_time ;
                  routerdeadinterval iftime ;
                  pollinterval iftime ;
                  passive ;
                  advertise-subnet on | off ;
                  auth [none | simple auth_key | md5 md5-keyset] ;
              } ] ;
             interface interface_name | interface_address nonbroadcast [ cost
                  ifnbcost ] [ {
                  strict-routers on | off ;
                  routers {
                      gatewaylist [ eligible ] ;
                  } ;
                  retransmitinterval ifnbtime ;
                  transitdelay ifnbtime ;
                  priority ifnbpriority ;
                  hellointerval ifnb_time ;
                  routerdeadinterval ifnbtime ;
                  pollinterval ifnbtime ;
                  passive ;
                  advertise-subnet on | off ;
                  auth [none | simple auth_key | md5 md5-keyset] ;
              } ] ;
             interface interface_name | interface_address point-to-multipoint
[
                      cost ptmcost ] [  {
                  strict-routers on | off ;
                  routers {
                      gatewaylist ;
                  } ;
                  retransmitinterval ptmtime ;
                  transitdelay ptmtime ;
                  priority ptmpriority ;
                  hellointerval ptmtime ;
                  routerdeadinterval ptmtime ;
                  pollinterval ptmtime ;
                  passive ;
                  advertise-subnet on | off ;
                  auth [none | simple auth_key | md5 md5-keyset]  ;
              } ] ;
```

Parameters

`retransmitinterval` *`time`*

`transitdelay` *`time`*

`priority` *`priority`*

`hellointerval` *`time`*

`routerdeadinterval` *`time`*

`pollinterval` *`time`*

`passive`

`advertise-subnet on | off`

`auth [ none | simple` *`auth_key`* **`| md5`** *`md5-key`* **`]`**

`interface_name` - name of interface to use

`interface_address` - logical address of interface to use

`strict-routers on | off` - nonbroadcast and point-to-multipoint only

`routers` - nonbroadcast and point-to-multipoint only

`gateway_list` - nonbroadcast and point-to-multipoint only

Description

This form of the **`interface`** clause is used to configure an interface in OSPF. Each interface has a cost. The costs of all interfaces that a packet must cross to reach a destination are summed to get the cost to that destination. The default cost is **`1`**, but another non-zero value may be specified.

The **`nonbroadcast`** form of the **`interface`** clause is used to specify a nonbroadcast interface on an NBMA medium. Because an OSPF broadcast medium must support IP multicasting, a broadcast-capable medium that does not support IP multicasting must be configured as a nonbroadcast interface. This includes the loopback interface on many operating systems.

The **`point-to-multipoint`** form of the **`interface`** clause is used to specify a point-to-multipoint interface. This form can be used when the network does not provide full connectivity to all routers on the network.

Both nonbroadcast and point-to-multipoint require a router’s section to specify the routers with which to exchange hellos. This is because they cannot use multicast to locate neighbors.

Default

If an area is configured, the default is no interfaces enabled. When no areas are configured, the default configuration is a backbone area with all interfaces enabled.

For the nonbroadcast form, the default mode is broadcast/point-to-point. The default cost is **`1`**.

For the point-to-point form, the default mode is point-to-point. The default cost is **`1`**.

Context

`ospf area` statement

Examples

Example 1

```
ospf yes {
    area 2.2.2.2 {
        interface fxp3 cost 1 {
            retransmitinterval 30;
            transitdelay 2;
            priority 62;
            hellointerval 10;
            routerdeadinterval 30;
            auth simple “simple”;
        };
    };
};
```

Example 2

```
ospf yes {
    area 2.2.2.2 {
        interface fxp0 nonbroadcast cost 1 {
            routers {
                192.168.0.1 ;
                192.168.1.1 ;
            };
        };
    };
};
```

Example 3

```
ospf yes {
    backbone {
        interface fxp0 cost 1 { passive; };
    };
};
```

See Also

area on page 98

retransmitinterval on page 130

transitdelay on page 148

priority on page 128

hellointerval on page 107

routerdeadinterval on page 133

pollinterval on page 126

auth on page 99

routers on page 135

strict-routers on page 137

networks

Name

`networks` - describes the networks comprising an area on an Area Border Router (ABR)

Syntax

```
networks {
    network mask mask [ restrict ];
    network masklen number [ restrict ];
    host nethost [ restrict ] ;
    [...]
};
```

Parameters

`network` - the network prefix to summarize

`mask` - mask of the network prefix

`number` - mask length of the network prefix

`nethost` - host route to be summarized

Description

The **`networks`** list describes the scope of an area on an ABR. Intra-area LSAs that fall within the specified ranges are not advertised into other areas as inter-area routes. Instead, the specified ranges are advertised as summary network LSAs. If restrict is specified, the summary network LSAs and all LSAs within the range are not advertised. Intra-area LSAs that do not fall into any range are also advertised as summary network LSAs. On well-designed networks, **`networks`** reduces the amount of routing information propagated between areas. The entries in this list are either networks, subnetwork/mask pairs, or subnetwork/masklen pairs. See Chapter 26, “Route Filtering” on page 129 of *Configuring GateD* for more detail about specifying ranges. Specifying **`networks`** on a non-ABR will have no effect.

Default

none

Context

`ospf area` statement

Examples

```
ospf yes {
    area 1.2.3.4 {
        networks {
            10.0.0.0 mask 255.0.0.0;
```

```
                192.168.0.0 mask 255.255.0.0 restrict;
                172.14.27 masklen 24;
            };
            interface fxp0 cost 1;
        };
    };
```

See Also

`nssanetworks` on page 121

`area` on page 98

nssa

Name

`nssa` - configures the area as a Not-So-Stubby-Area (NSSA) according to RFC 1587

Syntax

```
nssa [ cost defaultcost type 1 | 2 ] ;
```

Parameters

`defaultcost` - specifies that a type-7 default should be originated into the NSSA area with the given cost

`type 1 | 2` - the type of metric to be used in the default NSSA LSA

Description

`nssa` configures the area as an NSSA according to RFC 1587. If the router is an Area Border Router (ABR) and has the highest Router-ID of all the ABRs in the area, it will translate a type-7 LSA to type-5 LSA. The translation is affected by the **`nssanetworks`** clause, which is similar in operation to the **`networks`** clause. **`nssa`** and **`stub`** are mutually exclusive.

When **`cost`** is specified, the type of NSSA metric must be given with **`type`**. This sets the type of NSSA metric originated in the default NSSA LSA.

Default

The default type of area is non-stub non-NSSA. If **`nssa`** is specified, the default is no advertisement of a type-7 default LSA.

Context

`ospf area` statement

Examples

```
ospf yes {
    area 1.1.1.1 {
        nssa cost 1 type 1 ;
        interface 10.1.1.1 cost 1 ;
    };
};
```

See Also

`area` on page 98

`stub` on page 138

`nssanetworks` on page 121

nssa-cost

Name

`nssa-cost` - specifies the default cost when importing non-OSPF routes into OSPF NSSA

Syntax

`nssa-cost` *`defnssacost`***`;`**

Parameters

`defnssacost` - default cost of routes exported into a Not So Stubby Area (NSSA)

Description

`nssa-cost` is used when exporting a non-OSPF route from the GateD routing table into OSPF as an NSSA route. **`nssa-cost`** can be explicitly overridden in export policy.

Default

```
cost 1 ;
```

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        nssa-cost 1;
    };
    area 1.2.3.4 {
        interface fxp0;
    };
};
```

See Also

`defaults` on page 103

`cost` on page 102

nssa-inherit-metric

Name

`nssa-inherit-metric` - configures an OSPF NSSA route to inherit the metric of the external route when no metric is specified on the export policy

Syntax

```
inherit-metric ;
```

Parameters

none

Description

`nssa-inherit-metric` allows an OSPF NSSA route to inherit the metric of the external route when no metric is specified on the export policy. **`nssa-inherit-metric`** maintains compatibility with all the current export functions. A metric specified on the export policy will take precedence. The **`cost`** specified in the **`default`** statement (*`defnssacost`*) will be used if **`nssa-inherit-metric`** is not specified.

Default

The default is to not inherit the external metric.

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        nssa-inherit-metric;
    };
    backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

`defaults` on page 103

`cost` on page 117

`inherit-metric` on page 109

nssa-preference

Name

`nssa-preference` - specifies how preferred OSPF NSSA routes will be, compared to other protocols, when selecting active routes

Syntax

`nssa-preference` *`defnssapref`* **`;`**

Parameters

`defnssapref` - default NSSA preference

Description

`preference` specifies how active routes that are learned from the OSPF NSSA (compared to other protocols) will be selected. When a route has been learned from more than one protocol, the active route will be selected from the protocol with the lowest preference. Each protocol has a default preference in this selection.

Default

`nssa-preference 150 ;`

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        nssa-preference 160;
    };
        backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

`defaults` on page 103

`preference` on page 127

“Preferences and Route Selection” on page 11 of *Configuring GateD*

nssa-type

Name

`nssa-type` - changes the default type of routes exported from the GateD routing table into OSPF NSSA

Syntax

```
nssa-type 1 | 2  ;
```

Parameters

`1 | 2` - sets the default metric type to 1 or 2

Description

Routes exported from the GateD routing table into OSPF default to becoming type 1 NSSAs. This default can be explicitly changed here and overridden in export policy.

Default

```
nssa-type 1
```

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        nssa-type 2;
    };
    backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

`defaults` on page 103

`type` on page 150

nssanetworks

Name

`nssanetworks` - lists the net ranges that should be translated into type-5 LSAs from NSSA type-7 LSAs

Syntax

```
nssanetworks {
    network mask stubmask [ restrict ] ;
    network masklen number [ restrict ] ;
    host stubhost [ restrict ] ;
};
```

Parameters

network - specifies the network range to be aggregated into type-5

number - specifies the subnetwork mask length

restrict - specifies that this range should not be summarized into type-5

stubmask - specifies the subnetwork mask

stubhost - host route to be summarized

Description

nssanetworks lists the net ranges that should be translated into type-5 LSAs from NSSA type-7 LSAs. The default behavior is to translate type-7 LSAs that do not fall within a configured net range. This clause is valid only in an NSSA. It will be ignored when set in a non-NSSA.

Default

There are no default ranges. You must specify a range if **nssanetworks** is used.

The default cost is specified by the **nssa-cost** clause in the **defaults** clause.

Context

ospf area statement

Examples

```
ospf yes {

    area 1.2.3.4 {

        nssa;

        nssanetworks {

            10.0.0.0 mask 255.0.0.0 restrict;

            192.168.0.0 mask 255.255.0.0;

        };

        interface fxp0 cost 1;
```

```
            };
        };
```

See Also

`area` on page 98

`nssa` on page 138

`nssa-cost` on page 117

opaque-capability

Name

`opaque-capability` - configures support for RFC 2370 Opaque LSAs

Syntax

`opaque-capability on | off ;`

Parameters

`on | off` - enables or disables the option

Description

`opaque-capability` configures support for RFC 2370, “Opaque LSAs”. An internal API is provided for viewing and originating/flushing these types of LSAs. Because this can unnecessarily increase the size of the router’s Link-State Database, and it does not affect normal protocol operation, the default is **`off`**. Routers that do not support Opaque LSAs should continue to interoperate with those that do support them.

Default

`opaque-capability off`

Context

`ospf` statement

Examples

```
ospf yes {
    opaque-capability on;
    backbone {
        interface fxp0 cost 1;
    };
 };
```

See Also

`area` on page 98

`interface` on page 110

`ospf` on page 124

ospf

Name

`ospf` - enables or disables OSPF

Syntax

`ospf on | off {` *`ospf_parameters`* **`}`**

Parameters

`ospf_parameters` - all parameters in this section

Description

The **`ospf`** statement enables or disables OSPF. By default OSPF is disabled. *`ospf_parameters`* includes all the parameters in this section.

Default

```
ospf off ;
```

Context

global

Examples

```
    ospf yes ;
```

See Also

`area` on page 98

`interface` on page 110

passive

Name

`passive` - disables reception and transmission on an interface

Syntax

```
passive ;
```

Parameters

none

Description

`passive` causes GateD to not send or receive packets on this interface. For example, **`passive`** is used when this is the only router on the network. **`passive`** has the effect of originating a stub link to this interface into the domain.

Note: OSPF **`passive`** is not used to learn other routers' announcements, which is the way RIP **`passive`** is used. To learn about routes, if your host is connected to a single network on which there are multiple routers, use Router Discovery combined with ICMP redirects to learn a default route and the best route. If your host is connected directly to multiple networks, this method might not produce the best routes.

Default

The default is non-passive.

Context

`ospf area interface` statement

`ospf virtuallink` statement

Examples

```
ospf yes {
    backbone {
        interface fxp0 cost 1 { passive; };
    };
};
```

See Also

`interface` on page 110

`hellointerval` on page 107

pollinterval

Name

`pollinterval` - the length of time, in seconds, between OSPF packets that the router sends before adjacency is established with a neighbor

Syntax

`pollinterval` *`time`* **`;`**

Parameters

`time` - time, in seconds, between hello packets

Description

`pollinterval` is the length of time, in seconds, between OSPF packets that the router sends before adjacency is established with a neighbor. **`pollinterval`** can be specified at the global level and can be overridden at the area and interface levels. This can be used to reduce network overhead in cases where a router may or may not have a neighbor on a given interface at the expense of initial convergence time.

Default

`pollinterval 120 ;`

Context

`ospf` statement

`area` statement

`interface` statement

`virtuallink` statement

Examples

```
ospf yes {
    backbone {
        interface fxp0 cost 1 {
        pollinterval 120;
        };
    };
};
```

See Also

`interface` on page 110

`hellointerval` on page 107

preference

Name

`preference` - specifies how preferred OSPF ASE routes will be, compared to other protocols, when selecting active routes

Syntax

`preference` *`defasepref`* **`;`**

Parameters

`defasepref` - default ASE preference

Description

`preference` specifies how active routes that are learned from the OSPF ASE (compared to other protocols) will be selected. When a route has been learned from more than one protocol, the active route will be selected from the protocol with the lowest preference. Each protocol has a default preference in this selection.

Default

`preference 150 ;`

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        preference 160;
    };
        backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

`defaults` on page 103

`nssa-preference` on page 119

"Preferences and Route Selection" on page 11 of *Configuring GateD*

priority

Name

`priority` - a number between 0 and 255 that specifies the priority for becoming the designated router (DR)

Syntax

`priority` *`level`* **`;`**

Parameters

`level` - priority level for becoming DR

Description

`priority` is a number between 0 and 255 that specifies the priority for becoming the DR. When more than one router attached to a network attempts to become the DR, the one with the highest priority wins. If the competing routers have the same priority, the one with the highest router ID becomes the DR. The router coming in second in the election becomes the backup DR. A router with a router priority set to 0 is ineligible to become the DR. **`priority`** can be specified at the global level and overridden at the area and interface levels.

Note: **`priority`** applies only to broadcast or NBMA media.

Default

`priority 0 ;`

Context

`ospf` statement

`area` statement

`interface` statement

Examples

```
ospf yes {
    area 1.2.3.4 {
        priority 1;
        interface fxp0 cost 1;
        interface fxp1 cost 2 { priority 2; };
    };
        backbone {
        interface fxp2 cost 1;
    };
};
```

See Also

`interface` on page 110

`area` on page 98

retransmitinterval

Name

`retransmitinterval` - sets the default for the number of seconds between link state advertisement retransmissions for adjacencies

Syntax

`retransmitinterval` *`time`* **`;`**

Parameters

`time` - time, in seconds, for retransmission interval

Description

`retransmitinterval` sets the default for the number of seconds between link state advertisement retransmissions for adjacencies. If a Link State Protocol (LSP) is not acknowledged within **`retransmitinterval`** seconds, it is resent. This setting is another convergence/network traffic trade-off. **`retransmitinterval`** may be specified at the global level and overridden at the area and interface levels.

Default

`retransmitinterval 5 ;`

Context

`ospf` statement

`ospf area` statement

`ospf area interface` statement

`ospf area virtuallink` statement

Examples

```
ospf yes {
    backbone {
        retransmitinterval 10;
        interface fxp0 cost 1;
    };
};
```

See Also

`interface` on page 110

`area` on page 98

`virtuallink` on page 151

rfc1583compatibility

Name

rfc1583compatibility - enables RFC 1583 compatibility mode for the SPF calculation

Syntax

```
rfc1583compatibility on | off ;
```

Parameters

on | off - toggles whether or not the RFC 1583 preference rules are used

Description

Set **rfc1583compatibility** to **off** if all the routers using an OSPF implementation in your domain are based on RFC 2328 or later. This option should be set the same way on all routers in the domain. If any of the routers do not have this option, you should always choose **on**. When disabled, the preference rules for best route election are changed to eliminate certain kinds of possible routing loops.

Default

```
rfc1583compatibility on ;
```

Context

ospf statement

Examples

```
ospf on {
    rfc1583compatibility on;
    area 1.2.3.4 {
        interface fxp0 cost 1;
    };
};
```

See Also

ospf on page 124

ribs

Name

ribs - specifies the Routing Information Base (RIB) in which OSPF internal routes are installed

Syntax

```
ribs unicast [ multicast ] ;
```

Parameters

unicast - specifies that routes should be installed in the unicast RIB

multicast - specifies that routes should be installed in the multicast RIB

Description

ribs specifies the RIB in which OSPF internal routes are installed. The unicast RIB is required and is the default. In code bases that support extended RIBs, OSPF routes can be installed in the multicast RIB. **ribs** has no effect on self-originated ASE or NSSA routes, because they are exported from another protocol.

Default

```
ribs unicast
```

Context

ospf defaults statement

Examples

```
ospf yes {
    defaults {
        ribs unicast multicast;
    };
    backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

defaults on page 103

routerdeadinterval

Name

`routerdeadinterval` - configures the number of seconds that may elapse without receiving a neighbor's hello packets before the neighbor will be declared down

Syntax

`routerdeadinterval` *`time`* **`;`**

Parameters

`time` - time, in seconds, for the interval

Description

`routerdeadinterval` is the number of seconds that may elapse without receiving a router's hello packets before the router's neighbors will declare it down. **`routerdeadinterval`** can be specified at the global level and overridden at the area and interface levels. A general rule is for **`routerdeadinterval`** to equal three times the HELLO interval. Do not set **`routerdeadinterval`** to be less than **`hellointerval`**, or convergence will not occur.

Default

`routerdeadinterval 40 ;`

Context

`ospf` statement

`ospf area` statement

`ospf area interface` statement

`ospf area virtuallink` statement

Examples

```
ospf yes {
    backbone {
        interface fxp0 cost 1 {
            routerdeadinterval 20;
        };
    };
};
```

See Also

`interface` on page 110

`hellointerval` on page 107

`area` on page 98

`virtuallink` on page 151

routers

Name

routers - specifies one or more neighbors and an indication of their eligibility to become a designated router

Syntax

```
routers {
    gateway [eligible];
     [...]
};
```

Parameters

gateway - the IP address of another router on the network

eligible - specifies whether the *gateway* router is eligible to become a designated router (DR) (nonbroadcast mode only).

Description

By definition, it is not possible to send broadcast or multicast packets to discover OSPF neighbors on a nonbroadcast medium, so all neighbors must be configured. The gateway list includes one or more neighbors and an indication of their eligibility to become a DR.

Default

There are no default addresses. The default eligibility is not eligible.

Context

ospf area interface nonbroadcast statement

ospf area interface point-to-multipoint statement

Examples

```
ospf yes {
    backbone {
        interface fxp0 nonbroadcast cost 1 {
            routers {
                10.1.1.1 eligible;
            };
        };
    };
};
```

See Also

`interface` on page 110

`priority` on page 128

strict-routers

Name

`strict-routers` - configures GateD to ignore packets from routers not specified in the **`routers`** statement

Syntax

`strict-routers on | off ;`

Parameters

`on` | **`off`** - enables/disables the option

Description

If **`on`** (the default), GateD ignores packets from routers not specified in the **`routers`** statement.

Default

`strict-routers on`

Context

`ospf area interface nonbroadcast` statement

`ospf area interface point-to-multipoint` statement

Examples

```
ospf yes {
    backbone {
        interface fxp0 point-to-multipoint cost 1 {
            routers {
                10.1.1.1;
            };
            strict-routers on;
        };
    };
};
```

See Also

`interface` on page 110

`routers` on page 135

stub

Name

stub - configures an area as a stub area

Syntax

stub [cost *stub_default_cost* **] ;**

Parameters

stub_default_cost - cost of type-3 default summary to advertise into the area

Description

A **stub** area is one in which there are no ASE or NSSA routes. Each router in the area must specify that the area is a stub, or adjacencies will not form. If a **cost** is specified, **cost** is used to inject a default route into the area with the specified cost originating from this router. **cost** should be specified only on an Area Border Router (ABR). It is possible to use **stub** on multiple ABRs and give them different **cost**s. **stub** and **nssa** are mutually exclusive.

Default

The default type of area is non-stub and non-NSSA. The default for a stub area is to not advertise a type-3 summary.

Context

ospf area statement

Examples

```
ospf yes {
     area 1.2.3.4 {
        stub cost 1;
        interface fxp0 cost 1;
    };
};
```

See Also

area on page 98

nssa on page 116

stubhosts

Name

`stubhosts` - specifies directly attached hosts that should be advertised as reachable from this router and the costs with which they should be advertised

Syntax

```
stubhosts {
    host cost cost;
};
```

Parameters

`host` - IP address of host to advertise in router LSA

`cost` - cost of link to be advertised

Description

The **`stubhosts`** list specifies directly attached hosts that should be advertised as reachable from the router and the costs with which they should be advertised. Point-to-point interfaces on which it is not desirable to run OSPF should be specified here. It is also useful to assign an additional address to the loopback interface (one not on the 127 network) and advertise it as a stubhost. If this address is the same one used as the router ID, it enables routing to OSPF routers by router ID, instead of by interface address. Routing by router ID is more reliable than routing to one of the router's interface addresses, which may not always be reachable.

Default

none

Context

`ospf area` statement

Examples

```
ospf yes {
    area 1.2.3.4 {
        stubhosts {
            10.1.1.1 cost 1;
        };
        interface fxp0 cost 1;
    };
};
```

See Also

`stubnetworks` on page 141

`stub` on page 138

stubnetworks

Name

`stubnetworks` - specifies directly attached networks that should be advertised as reachable from this router and the costs with which they should be advertised

Syntax

```
stubnetworks {
    network mask stubmask cost cost ;
    network masklen number cost cost ;
    host stubhost cost cost ;
};
```

Parameters

`network` - network to be advertised

`stubmask` - netmask of the network to be advertised

`number` - mask length of network to be advertised

`cost` - cost of the network

`stubhost` - host route to be summarized

Description

The **`stubnetworks`** list specifies directly attached networks that should be advertised as reachable from this router and the costs with which they should be advertised. Interfaces on which it is not desirable to run OSPF should be specified here. No checking is currently done on whether the specified network is actually reachable from this router, so care should be taken.

Default

none

Context

`ospf area` statement

Examples

```
ospf yes {
    backbone {
        stubnetworks {
            10 mask 255.0.0.0 cost 1;
            192.168 masklen 16 cost 5;
        };
        interface fxp0 cost 1;
    };
    };
```

See Also

stubhosts on page 139

passive on page 125

summaryfilters

Name

`summaryfilters` - contains route filters that specify which summary LSAs to filter from a non-transit area

Syntax

```
summaryfilters {
    route_filter
    [...]
};
```

Parameters

`route_filter` - See the section on route filters for syntax.

Description

The **`summaryfilters`** statement contains route filters that specify which summary LSAs to filter from the a non-transit (typically **`stub`** or **`nssa`**) area. That is, if a summary would normally be injected into the area, it is compared against the summary-filters list, and if a match is found, the announcement of the summary LSAs into the stub area will be suppressed. For normal operation, summary-filters should only be used in **`stub`** or **`nssa`** areas that have a default route being generated (see **`stub`** or **`nssa`**). In this usage, you can filter all summary LSAs (not including the generated default) to further reduce the amount of routing information present in the stub area's routers. Use of **`summaryfilters`** in non-stub non-NSSA areas is not recommended because it can break routing.

Default

By default all summary is passed into the area.

Context

`ospf area` statement

Examples

```
ospf yes {
    area 1.2.3.4 {
        stub cost 1;
        summary-filters {
            10.0.0.0 mask 255.0.0.0;
        };
        interface fxp0 cost 1;
    };
};
```

See Also

`stub` on page 138

`nssa` on page 116

`area` on page 98

“Chapter 28 Route Filtering” on page 129 of “Configuring GateD”

tag

Name

tag - used to propagate data from an exterior gateway protocol (such as BGP) through OSPF

Syntax

tag [as] *tagvalue* **;**

Parameters

as - specifies that *tagvalue* is a BGP AS number

tagvalue - a 32-bit value for the tag

Description

OSPF ASE routes have a 32-bit tag field that is not used by the OSPF protocol, but can be used when exporting to protocols other than OSPF. When OSPF is interacting with BGP, the **tag** field can be used to propagate AS path information, in which case the **as** keyword is specified, and the tag is limited to 12 bits of information. If not specified, the tag is set to **0**.

Default

tag [**as**] **0 ;**

Context

ospf defaults statement

Examples

```
ospf yes {
    defaults {
        tag 2112;
    };
    area 1.2.3.4 {
        interface fxp0 cost 1;
    };
};
```

See Also

defaults on page 103

traceoptions

Name

`traceoptions` - specifies the tracing options for OSPF

Syntax

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

Parameters

`trace_options_ospf` - tracing options for the OSPF protocol

Description

`traceoptions` specifies the tracing options for OSPF.

The following are OSPF-specific traceoptions:

`lsabuild` - Trace the creation of link-state advertisements.
`lsatransmit` (or **`lsatx`**) - Trace the link-state packets transmitted.
`lsareceive` (or **`lsarx`**) - Trace the link-state packet received.
`spf` - Trace the Shortest Path First (SPF) calculations.
`debug` - Trace OSPF at the debugging level of detail.

Packet tracing options (which can be modified with **`detail`**, **`send`**, and **`recv`**) include:

`hello` - Trace OSPF hello packets, which are used to determine neighbor reachability.
`dd` - Trace OSPF Database Description (DD) packets, which are used in synchronizing OSPF databases.
`request` - Trace OSPF link-state request packets, which are used in synchronizing OSPF databases.
`lsu` - Trace OSPF link-state update packets, which are used in synchronizing OSPF databases.
`ack` - Trace OSPF link-state ack packets, which are used in synchronizing OSPF databases.

Default

The default trace options are inherited from the global trace options.

Context

`ospf` statement

Examples

```
ospf yes {
    traceoptions “/tmp/log” all;
        area 1.2.3.4 {
            interface fxp0 cost 1;
    };
    };
```

See Also

"Trace Statements" on page 15 of *Configuring GateD*

transitdelay

Name

`transitdelay` - sets the estimated number of seconds required to transmit a link state update

Syntax

`transitdelay` *`time`* **`;`**

Parameters

`time` - time, in seconds, for the transit delay

Description

`transitdelay` sets the estimated number of seconds required to transmit a link state update. **`transitdelay`** takes into account transmission and propagation delays and must be greater than 0. **`transitdelay`** can be specified at the global level and overridden at the area and interface levels.

Default

`transitdelay 1 ;`

Context

`ospf` statement

`ospf area` statement

`ospf area interface` statement

`ospf area virtuallink` statement

Examples

```
ospf yes {
    area 1.2.3.4 {
        transitdelay 5;
        interface fxp0 cost 1;
        interface fxp1 cost 2 {
            transitdelay 10;
        };
        interface fxp2 cost 3;
    };
};
```

See Also

`interface` on page 110

`area` on page 98

`virtuallink` on page 151

type

Name

`type` - changes the default type of routes exported from the GateD routing table into OSPF ASE

Syntax

```
type 1 | 2  ;
```

Parameters

`1 | 2` - sets the default metric type to 1 or 2

Description

Routes exported from the GateD routing table into OSPF default to becoming type 1 ASEs. This default can be explicitly changed here and overridden in export policy.

Default

```
type 1
```

Context

`ospf defaults` statement

Examples

```
ospf yes {
    defaults {
        type 2;
    };
    backbone {
        interface fxp0 cost 1;
    };
};
```

See Also

`defaults` on page 103

`nssa-type` on page 120

virtuallink

Name

`virtuallink` - configures a virtual link on the backbone

Syntax

`virtuallink neighborid` *`router_id`* **`transitarea`** *`area`*

Parameters

`router_id` - specifies the router-ID of the other end of this link

`area` - specifies the transit area to be used for the link

Description

Virtual links are used to establish or increase connectivity of the backbone area. The **`neighborid`** is the **`router_id`** of the other end of the virtual link. The area specified in the **`transitarea`** must also be configured on the system. All standard interface parameters defined by the **`interface`** clause can be specified on a virtual link.

Note: **`virtuallink`** is available on the backbone only.

Default

none

Context

`ospf area` statement

Examples

```
ospf yes {
    backbone {
        interface fxp0 cost 1;
        virtuallink neighborid 2.1.1.2 transitarea 1.2.3.4;
    };
    area 1.2.3.4 {
        interface fxp1 cost 2;
    };
};
```

See Also

`interface` on page 110