Chapter 28
Route Exportation

export proto bgp

Name

`export proto bgp` - specifies the BGP peers that will receive routes

Syntax

```
export proto bgp as autonomous_system restrict ;
export proto bgp as autonomous_system
    [ comm-add { communities_list } ]
    [ comm-delete { communities_list } ]
    [ ext-comm-add { extended_communities_list } ]
    [ ext-comm-del { extended_communities_list } ]
    [ metric metric ] {
        export_source_statements
    } ;
```

Parameters

`autonomous_system` - autonomous system number from 1 to 65535

`comm-add {` *`communities_list`* **`}`** - communities to be added. See “Chapter 30 BGP Communities” on page 133 in *Configuring GateD* for more information. By default no communities are added.

`comm-delete {` *`communities_list`* **`}`** - communities to be deleted. See “Chapter 30 BGP Communities” on page 133” in *Configuring GateD* for more information. By default no communities are deleted.

`ext-comm-add {` *`extended_communities_list`* **`}`** - extended communities to be added. See “Chapter 30 BGP Communities” on page 133 in *Configuring GateD* for more information. By default no extended communities are added.

`ext-comm-del {` *`extended_communities_list`* **`}`** - extended communities to be deleted. See “Chapter 30 BGP Communities” on page 133” in *Configuring GateD* for more information. By default no extended communities are deleted.

`metric` - BGP Multi-exit discriminator metric from 0 to 65535, by default the BGP **`metricout`** value is used.

export_source_statements - zero or more source statements: **proto bgp**, **proto rip**, **proto ripng**, **proto ospf**, **proto ospfase**, **proto direct**, **proto static**, **proto kernel**, **proto isis**, or **proto aggregate**.

Description

The **proto bgp** export target statement specifies the neighboring AS to which routes will be distributed for those routes that match the associated *export_source_statements*. Additionally, the **proto bgp** command can be used to set community path attributes through the **comm-add**, **comm-delete**, **ext-comm-add**, and **ext-comm-del** commands.

Like other *export_target_statement*s, **proto bgp** allows one to set a metric on the propagated routes. Of course, BGP does not have a metric in the sense that the link state protocols do, instead using a complicated scheme for selecting a route. (Refer to "Route Selection" on page 65 in *Configuring GateD* for more information.) Setting the metric with proto bgp sets the MED used in steps 6 and 7 of this process. More detail on MEDs can be found in "Multi-Exit Discriminator Overview and Examples" on page 79 in *Configuring GateD*. From a scoping perspective, metric as set by this command is tighter scope than that set in either the group or peer statement, and looser scope than that set in an *export_source_statement*.

Defaults

BGP will, by default, export all default routes if there is no explicit export policy that applies to this peer.

Context

global statement

Examples

Example 1

Configure GateD to export all EBGP routes to its IBGP peers, where *myAS* is my local AS number.

```
export proto bgp as myAS {
    proto bgp aspath "(.+)" origin any {
        all;
    };
};
```

Example 2

The following example exports all routes learned via RIP and its external BGP peers to all BGP peers in AS 65534, adds to the advertisement the well-known community NO_EXPORT_SUBCONFED, and strips the arbitrary community 0x123 0x321.

```
export proto bgp as 65534
     comm-add { community no-export-subconfed; }
     comm-delete { comm-hex 0x123 0x321; } {
```

```
            proto rip {
                all;
            } ;
            proto bgp aspath "(.+)" origin any {
                all;
            };
        } ;
```

Example 3

The following example exports all OSPF routes to BGP peers in AS 65533.

```
export proto bgp as 65533 {
    proto ospf {
        all;
    };
};
```

Example 4

The following example exports into BGP, to all peers in AS 65533, all, and only, routes generated via an aggregate statement.

```
export proto bgp as 65533 {
    proto aggregate {
        all;
    };
};
```

Example 5

The following example exports into BGP, to all peers in AS 65533, all, and only, routes learned with a leading AS of 65532.

```
export proto bgp as 65533 {
    proto bgp aspath "( 65532 .* )" origin any {
        all;
    };
};
```

See Also

Application of the Border Gateway Protocol in the Internet (RFC 1164) at http://ietf.org/rfc/rfc1164.txt

"AS Path Regular Expressions" on page 131 in *Configuring GateD*

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

comm-add on page 595

export proto isis

Name

export proto isis - specifies how to inject routes into IS-IS

Syntax

```
export proto isis restrict ;
export proto isis [ metric-type type ] [ level level ] [ metric metric ] {
    export_source_statements
} ;
```

Parameters

type - either **internal** or **external**

level - the IS-IS level, either **1** or **2**. (Defaults to **2**.)

metric - an IS-IS metric from 1 to 63

export_source_statements - zero or more source statements: **proto bgp**, **proto rip**, **proto ospf**, **proto ospfase**, **proto direct**, **proto static**, **proto kernel**, **proto isis**, or **proto aggregate**.

Description

The **proto isis** export target specifies that routes matching the subsequent export sources should be exported via the IS-IS protocol. Normally, only networks associated with interfaces on which IS-IS is being run will be advertised. In order to override this behavior (for example, to advertise directly connected networks on which IS-IS is not being run, or to distribute routes learned from another protocol via IS-IS), a **proto isis** export target statement must be used.

As with other *export_target_statements*, the arguments associated with **proto isis** are specific to the IS-IS protocol, and a clear understanding of the protocol is important in order to implement appropriate policy. For example, explaining the difference between level 1 and level 2 routers is beyond the scope of this document.

The **metric-type** keyword is used to specify whether the metric for routes exported to IS-IS via the export source statement associated with this export target are directly comparable to normal IS-IS metrics. Note that the "wide" metric TLV does not support the marking of external metrics. If this type of reachability is originated the metric will be "internal" regardless of the setting here.

If the metric is to be considered comparable and used in the IS-IS SPF algorithm, then **internal** should be specified. If the metrics are incompatible and should not be used, then **external** should be specified. If no **metric-type** keyword is issued, then the behavior will be that specified by the export-defaults metric-type statement.

It is worth noting that the IS-IS concept of internal and external metrics is not directly analogous to the OSPF concept of type 1 and type 2 ASEs. Specifically, IS-IS has internal and external reachability, rather than the concept of ASE. External reachability (and the associated metric for routes which are externally reachable) is in behavior the same as if

an OSPF route were learned as a type 2 ASE route. However, internal reachability and the associated metrics are directly related to IS-IS routes. (For example, they are not less preferred than IS-IS routes, whereas type 1 ASE routes in OSPF are less preferred than OSPF routes.)

The **level** keyword is used to override the level specified by the export-defaults level command. Because this is an overriding operation, if it is desired to export routes both to level 1 and level 2 routers, two export commands will be required with differing **proto isis** export targets and matching export sources.

Finally, the **metric** keyword is used to set the cost on routes matching the subsequent export sources. The exact use of that metric depends on whether the routes are being exported as internal or external. If they are being exported as external, then any metric set is used as is, without adding any IS-IS link costs. This allows one, for example, to specify a certain ASBR for all traffic to matching destinations, regardless of its location within the IS-IS mesh. For internal routes, the behavior is identical to what would happen if it were an IS-IS route from the beginning.

Defaults

By default, nothing is exported into IS-IS.

Context

global statement

Examples

Example 1

The following example injects all BGP routes learned via peers in AS 65534 into IS-IS with external reachability.

```
export proto isis metric-type external {
    proto bgp as 65534 {
        all;
    };
};
```

Example 2

In order to learn the best route to all static routes configured on IS-IS routers, the following can be used.

```
export proto isis metric-type internal {
    proto static {
        all;
    };
};
```

Example 3

The following example configures GateD to export the 1::/64 static route into ISIS IPv6 external reachability

```
export proto isis metric-type external {
    proto static {
        1::/64;
    } ;
};
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export-defaults on page 171

export proto bgp on page 623

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

export proto ripng on page 638

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

export proto ospfase

Name

export proto ospfase - specifies how to inject routes into OSPF as ASE

Syntax

```
export proto ospfase restrict ;
export proto ospfase [ type type ] [ tag tagvalue ] [ metric metric ] {
    export_source_statements
};
```

Parameters

type - the type of AS External route, 1 or 2

tagvalue - an arbitrary number from 0 to 4294967295

metric - OSPF ASE cost from 0 to 16777215

export_source_statements - zero or more source statements: **proto bgp**, **proto rip**, **proto ospfase**, **proto direct**, **proto static**, **proto kernel**, **proto isis**, or **proto aggregate**.

Description

The **proto ospfase** export target specifies that routes matching the subsequent export sources should be exported through the OSPF protocol as AS external (ASE) routes. OSPF uses four different types of routes: intra-area, inter-area, type 1 ASE, and type 2 ASE. These four types of routes are listed in the order in which they are preferred, so if, for example, a route to the same destination is learned both via type 1 ASE and type 2 ASE, then the route learned via type 1 will always be used. Both types of ASE routes are routes to destinations external to OSPF (and usually external to the AS). Routes exported into OSPF ASE as type 1 ASE routes (via the type option) are supposed to be from interior gateway protocols (such as RIP) whose external metrics are directly comparable to OSPF metrics.

When a routing decision is being made, OSPF will add the internal cost to the AS border router to the external metric. Type 2 ASEs are used for exterior gateway protocols whose metrics are not comparable to OSPF metrics. In this case, only the metric associated with the ASE route is used, and the internal OSPF cost to the ASBR is ignored (except for tie breaking). If no type is specified, then the default is to use whatever was configured by the **type** command.

The **tag** command sets an arbitrary number in the tag field for use in policy matching. This tag has no meaning in the context of the OSPF protocol, but is instead used as a filter for matching routes via the various *export_source_statements*.

Finally, the **metric** keyword is used to set the cost on routes matching the subsequent export sources. The exact use of that metric depends on whether the routes are being exported as type 1 or type 2 ASE. If they are being exported as type 2, then any metric set is propagated unchanged throughout the OSPF network, allowing one, for example, to specify that all ASE traffic go through a single ASBR, regardless of that ASBR's location in

the OSPF cloud. If no metric is specified, then the metric will be set according to the **inherit-metric** command.

It is important to note that this only controls the distribution of routes as type 5 LSAs; for distributing routes via type 7 LSAs, separate **proto ospfnssa** export targets must explicitly be configured. Type-5 LSAs (AS External LSAs) are distributed throughout type-5 capable areas in an OSPF domain. They represent destinations external to the AS. Type-7 LSAs are distributed within Not So Stubby Areas (NSSAs) and may be translated or aggregated to type-5 LSAs by the Area Border Routers (ABRs) bordering NSSA area(s).

Defaults

By default, nothing is exported into OSPF ASE.

Context

global statement

Examples

Example 1

The following example injects all BGP routes learned via peers in AS 65534 in to OSPF ASE as type 2 routes. This is not recommended.

```
export proto ospfase type 2 {
    proto bgp as 65534 {
        all;
    };
};
```

Example 2

The following example ensures that all directly connected networks are distributed via OSPF.

```
export proto ospfase {
    proto direct {
        all;
    };
};
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfnssa on page 633

export proto rip on page 635

inherit-metric on page 109

"OSPF Overview" on page 45 in *Configuring GateD*

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

export proto ripng on page 638

proto static on page 669

type on page 150

stubnetworks on page 141

export proto ospfnssa

Name

export proto ospfnssa - specifies how to inject ASE routes as type 7 LSAs

Syntax

```
export proto ospfnssa restrict ;
export proto ospfnssa [ type type ] [ tag tagvalue ] [ metric metric ] {
    export_source_statements
};
```

Parameters

type - the type of AS External route, 1 or 2

tagvalue - an arbitrary number from 0 to 4294967295

metric - OSPF ASE cost from 0 to 16777215

export_source_statements - zero or more source statements: **proto bgp**, **proto rip**, **proto ospfase**, **proto direct**, **proto static**, **proto kernel**, **proto isis**, or **proto aggregate**.

Description

The **proto ospfnssa** export target specifies that routes matching the subsequent export sources should be exported via type 7 LSAs in OSPF. In short, this means that the behavior of this export target is identical to the behavior of the **proto ospfase** target, with the exception that the routes matching subsequent *export_source_statements* will only be advertised to areas that are configured to be "not so stubby."

Defaults

By default, nothing is exported via type 7 LSAs.

Context

global statement

Examples

The following example injects all BGP routes learned via peers in AS 65534 into OSPF ASE as type 2 routes, but only into those areas which are configured as NSSA.

```
export proto ospfnssa type 2 {
    proto bgp as 65534 {
        all;
    };
};
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto rip on page 635

export proto ripng on page 638

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

export proto rip

Name

export proto rip - specifies how to inject routes into RIP

Syntax

```
export proto rip
    [ interface interface_list | gateway gateway_list ]
    restrict ;
export proto rip
    [ tag tagvalue ]
    [ interface interface_list | gateway gateway_list ]
    [ metric metric ] {
        export_source_statements
    };
```

Parameters

interface_list - the list of interfaces over which routes from a subsequent export source will be sent or **all** for all interfaces on which RIP is configured

gateway_list - the list of next hop RIP routers to which routes from a subsequent export source will be sent

tagvalue - an arbitrary number from 0 to 65535

metric - RIP metric number from 1 to 15

export_source_statements - zero or more source statements: **proto bgp**, **proto rip**, **proto ospfase**, **proto direct**, **proto static**, **proto kernel**, **proto isis**, or **proto aggregate**.

Description

The **proto rip** export target specifies how routes are to be readvertised as RIP routes. The **interface** and **gateway** parameters specify that the routes matched in subsequent *export_source_statements* will only be advertised out the associated *interface_list* or *gateway_list*. In the absence of the **interface** or **gateway** keywords, routes will be sent out all interfaces on which RIP is configured.

Because RIP is a broadcast protocol (or multicast in the case of RIPv2), the effect of the **gateway** keyword can be somewhat counter-intuitive. Specifically, the user would expect RIP routes to only be advertised to the specific gateways mentioned in an instance of the **proto rip** export target statement. This is in fact the case if those gateways are source gateways. However, if source gateways are not used, then **gateway** effectively acts as a wildcard for **interface**. By specifying gateways, the routes which pass the export source filter associated with an export command will be either broadcast or multicast (as appropriate) to all RIP routers on the interfaces whose subnets include the specified gateways.

It is important to note that the *gateway_list* must only include gateways on directly connected interfaces; otherwise a parse error will occur. If the interface upon which a specified gateway is directly reachable may not be functional at the time GateD configuration takes place, then the **interfaces define** statement should be used to predefine the interface.

Finally, the **metric** keyword is used to set the RIP metric on routes matching the subsequent export sources. This keyword is used to override the RIP metric set with the rip **defaultmetric** command.

Defaults

```
export proto rip {
    proto direct {
        all;
    };
};
```

Context

global statement

Examples

Example 1

The following example exports all, and only, routes specified in the static clause of the GateD configuration file via RIP.

```
export proto rip {
    proto static {
        all;
    };
};
```

Example 2

The following example advertises all static, direct, and RIP routes, via RIP, out the interface 192.168.2.10.

```
export proto rip interface 192.168.2.10 {
    proto static {
        all;
    };
    proto direct {
        all;
    };
    proto rip {
```

```
            all;
        };
    };
```

See Also

“Chapter 31 Route Importation” on page 137 in *Configuring GateD*

defaultmetric on page 55

define on page 22

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

export proto ripng on page 638

proto static on page 669

sourcegateways on page 76

version on page 83

export proto ripng

Name

export proto ripng - specifies how to inject routes into RIPng

Syntax

```
export proto ripng
    [ interface interface_list | gateway gateway_list ]
    restrict ;
export proto ripng
    [ tag tagvalue ]
    [ interface interface_list | gateway gateway_list ]
    [ metric metric ] {
        export_source_statements
    };
```

Parameters

interface_list - the list of interfaces over which routes from a subsequent export source will be sent or **all** for all interfaces on which RIPng is configured

gateway_list - the list of next hop RIPng routers to which routes from a subsequent export source will be sent

tagvalue - an arbitrary number from 0 to 65535

metric - RIPng metric number from 1 to 15

export_source_statements - zero or more source statements: **proto bgp**, **proto direct**, **proto static**, **proto kernel**, **proto isis**, or **proto aggregate**.

Description

The **proto ripng** export target specifies how IPv6 routes are to be readvertised as RIPng routes. The interface and gateway parameters specify that the routes matched in subsequent *export_source_statements* will only be advertised out the associated *interface_list* or *gateway_list*. In the absence of the **interface** or **gateway** keywords, routes will be sent out all interfaces on which RIPng is configured.

Because RIPng uses multicast, the effect of the gateway keyword can be somewhat counter-intuitive. Specifically, the user would expect RIPng routes to only be advertised to the specific gateways mentioned in an instance of the **proto ripng** export target statement. This is in fact the case if those gateways are source gateways. However, if source gateways are not used, then **gateway** effectively acts as a wildcard for **interface**. By specifying gateways, the routes which pass the export source filter associated with an export command will be either broadcast or multicast (as appropriate) to all RIPng routers on the interfaces whose subnets include the specified gateways.

It is important to note that the *gateway_list* must only include gateways on directly connected interfaces; otherwise a parse error will occur. If the interface upon which a speci-

fied gateway is directly reachable may not be functional at the time GateD configuration takes place, then the **interfaces define** statement should be used to predefine the interface.

Finally, the **metric** keyword is used to set the RIPng metric on routes matching the subsequent export sources. This keyword is used to override the RIPng metric set with the **ripng defaultmetric** command.

Defaults

```
export proto ripng {
    proto direct {
        all;
    };
};
```

Context

global statement

Examples

Example 1

The following example exports all, and only, IPv6 routes specified in the static clause of the GateD configuration file via RIPng.

```
export proto ripng {
    proto static {
        all;
    };
};
```

Example 2

The following example advertises all static, direct, and RIPng IPv6 routes, via RIPng, out the interface fec0::01.

```
export proto ripng interface fec0::01 {
    proto static {
        all;
    };
    proto direct {
        all;
    };
};
```

See Also

“Chapter 31 Route Importation” on page 137 in *Configuring GateD*

defaultmetric on page 557

define on page 22

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

sourcegateways on page 76

version on page 83

fromribs

Name

fromribs - specifies the RIBs from which a route will be exported

Syntax

route_filter **fromribs** *riblist* **;**

Parameters

route_filter - a set of route filters specifying routes to match. These filters specify whether a route is to be accepted, and if so, with what attributes. See “Chapter 28 Route Filtering” on page 129 of *Configuring GateD* for more information.

riblist - one of the following:

unicast - specifies that the routes in the unicast RIB are to be exported

multicast - specifies that the routes in the multicast RIB are to be exported

unicast multicast - specifies that the routes in both the unicast and multicast RIBs are to be exported

Description

fromribs is used to indicate the RIBs from which a route will be exported. It is currently only applicable to *route_filters* within a BGP export target because only BGP can export routes from either the unicast or multicast RIBs. It should be used only on **route_filters** associated with the **bgp**, **direct**, or **static** export source statements, as only these protocols can import routes in the multicast RIB.

If just one of the unicast or multicast RIBs is specified for **fromribs** and a route to be exported is in both RIBs, the route will be exported to just the specified **fromribs**.

Defaults

If **fromribs** is not specified, routes from either RIB are accepted and exported according to **toribs**. By default, BGP exports routes in either the unicast or multicast RIBs. All other export protocols export only routes in the unicast RIB.

Context

route_filter

Examples

```
export proto bgp as 65534 {
    all fromribs multicast ;
} ;
```

This example exports only multicast routes learned from this AS.

See Also

"Chapter 32 Route Exportation" on page 145 in *Configuring GateD*

`proto bgp` on page 646

`proto direct` on page 649

`proto static` on page 669

proto aggregate

Name

proto aggregate - filters on routes which are aggregates of other routes

Syntax

```
proto aggregate restrict ;
proto aggregate [ noagg ] [ metric metric ]
    { [ route_filter [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filter*s. It indicates that any matching routes are not to be exported.

Description

The **proto aggregate** export source statement is used to match routes which have been aggregated from other routes for exportation via one of the export target statements, described in this chapter.

In the **restrict** version of this *export_source_statement*, any aggregated routes are prohibited from redistribution in the associated *export_target_statement*.

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target. Upon first inspection, this may seem like a non-sequitur as we are matching aggregate routes, but in fact it is not. Keep in mind that an aggregated route is distinct from one which contributes to an aggregate, and also that it is possible to aggregate from other, previously aggregated routes.

As with all **export** source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the *export_target_statement*. Setting a metric value in an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filters* are provided, then any routes matching these filters will be passed on to the export target within which this **proto aggregate** is included.

Any number of unique **proto aggregate** export sources can be used within the context of a single **export** target statement.

Defaults

By default, no **proto aggregate** export source exists.

Context

export source statement

Examples

Example 1

The following example exports into OSPF as ASE all routes which are aggregates of other routes.

```
export proto ospfase {
    proto aggregate {
        all;
    };
};
```

Example 2

The following example exports into RIP all routes which are aggregates of other routes, provided that they themselves are not contributing to a greater aggregate.

```
export proto rip {
    proto aggregate noagg {
        all;
    };
};
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

proto bgp

Name

proto bgp - filters on routes learned via BGP

Syntax

```
proto bgp ( as autonomous_system ) | ( aspath aspath_regular_expression
    origin ( any | igp | egp | incomplete ) )
    [ comm communities_list ]
    [ ext-comm extended_communities_list ]
    restrict;
proto bgp ( as autonomous_system ) | ( aspath aspath_regular_expression
    origin ( any | igp | egp | incomplete ) )
    [ comm communities_list ]
    [ ext-comm extended_communities_list ]
    [ noagg ] [ metric metric ]
    { [ route_filter
      [ fromribs riblist ] [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

autonomous_system - autonomous system number from 1 to 65535

aspath_regular_expression - The syntax of these regular expressions is described in “AS Path Regular Expressions” on page 131, and in section 4.2 of RFC 1164.

comm { *communities_list* **}** - specifies that this import statement applies to routes learned with the communities specified in *communities_list*

ext-comm { *extended_communities_list* **}** - specifies that this import statement applies to routes learned with the extended communities specified in *extended_communities_list*

metric - a metric with a range appropriate to the protocol as described in the associated export target

route_filter **-** route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

fromribs *riblist* **-** specifies the RIBs from which routes matching this route_filter will be exported. See “**fromribs**” on page 641 for more information.

Description

The **proto bgp** export source statement is used to match routes learned via the BGP protocol for exportation via one of the export target statements, described in this chapter.

The statement includes either an **as** or **aspath** component, an optional community matching component, an optional extended community component, and an optional **noagg** component. If any *route_filter*s are provided, then any routes matching those filters which also match the **as** portion and the optional community portions will be passed onto the export target for which this **proto bgp** is an export source. The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target.

Any number of **proto bgp** export sources can be used within the context of a given export target, so long as they are not repeated.

As with all export source statements, the optional *metric* must be a valid metric for the associated protocol in the export target statement. Setting a metric value in an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filter*s are provided, then any routes matching these filters that also match the **proto bgp** statement parameters will be passed on to the export target within which this **proto bgp** statement is included.

Defaults

By default there are no **proto bgp** export source statements.

Context

export source statement

Examples

Example 1

The following example exports into BGP, to all peers in AS 65533, all, and only, routes learned with a leading AS of 65532.

```
export proto bgp as 65533 {
    proto bgp aspath “( 65532 .* )” origin any {
        all;
    };
};
```

Example 2

The following example injects all BGP routes learned via peers in AS 65534 in to OSPF ASE as type 2 routes. This is not recommended.

```
export proto ospfase type 2 {
    proto bgp as 65534 {
        all;
```

```
        };
    };
```

Example 3

The following example exports into BGP, to all peers in AS 65533, all, and only, routes learned with a leading AS of 65532.

```
    export proto bgp as 65533 {
        proto bgp aspath "( 65532 .* )" origin any {
            all;
        };
    };
```

See Also

Application of the Border Gateway Protocol in the Internet (RFC 1164) at http://ietf.org/rfc/rfc1164.txt

"AS Path Regular Expressions" on page 131 in *Configuring GateD*

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

comm on page 593

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

fromribs on page 641

proto aggregate on page 643

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

proto direct

Name

proto direct - filter on directly connected interfaces

Syntax

```
proto direct [ ( interface interface_list ) ] restrict ;
proto direct [ ( interface interface_list ) ]
    [ noagg ] [ metric metric ]
    { [ route_filter
      [ fromribs riblist ] [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

interface_list - the list of interfaces on which the matching route must be present in order for it to be exported to the associated export target

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See "Chapter 28 Route Filtering" on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

fromribs *riblist* - specifies the RIBs from which routes matching this route_filter will be exported. See "**fromribs**" on page 641 for more information.

Description

The **proto direct** export source statement is used to match routes associated with directly connected interfaces for exportation via one of the export target statements, described in this chapter.

If the optional interface parameter is used, only routes associated with those interfaces matching the *interface_list* will be tested against the remaining filters and policy.

In the **restrict** version of this export source statement, any direct routes which match the optional interface parameter (or all direct routes in the absence of said parameter) will not be exported via the associated *export_target_statement*.

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target. Refer to "Chapter 33 Route Aggregation and Generation" on page 155 in *Configuring GateD* for more information.

As with all export source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the *export_target_statement*. Setting a metric value in

an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filter*s are provided, then any routes matching these filters that also match the optional interface specification will be passed on to the export target within which this **proto direct** is included.

Any number of unique **proto direct** export sources can be used within the context of a single export target statement.

Defaults

By default, no **proto direct** export source exists (except as described in the **export** command).

Context

export source statement

Examples

Example 1

The following example exports into RIP all directly connected routes.

```
export proto rip {
    proto direct {
        all;
    };
};
```

Example 2

The following example exports into RIP all directly connected routes, except those on the interface eth0.

```
export proto rip {
    proto direct interface eth0 restrict;
    proto direct {
        all;
    };
};
```

Example 3

The following example exports into RIP all directly connected routes, but routes from interface eth0 are exported with a higher metric. **Note**: The default rip metric is 1.

```
export proto rip {
    proto direct eth0 metric 2 {
        all;
```

```
        };
        proto direct {
            all;
        };
    };
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

proto aggregate on page 643

proto bgp on page 646

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

proto isis

Name

proto isis - filters on routes learned via IS-IS

Syntax

```
proto isis [ internal | external ] restrict ;
proto isis [ internal | external ]
    [ noagg ] [ metric metric ]
    { [ route_filter [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

metric **-** a metric with range appropriate to the protocol described in the associated export target

route_filter **-** route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

internal - explicitly propagates only those routes which have internal reachability

external - explicitly propagates only those routes which have external reachability

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

Description

The **proto isis** export source statement is used to match routes that have been propagated through IS-IS for exportation via one of the export target statements, described in this chapter.

The **internal** and **external** keywords are used to explicitly propagate only those routes which have internal or external reachability, respectively. In the absence of either keyword, the **proto isis** export source statement defaults to only matching IS-IS routes that have been received with internal reachability. As a result, in order to propagate all IS-IS routes, a separate export source must be created for both internal and external reachability.

In the **restrict** version of this *export_source_statement*, no IS-IS routes of the appropriate reachability will be exported via the associated *export_target_statement*. Strictly speaking, using the **restrict** version of this statement has no effect, as it would be the same behavior as if the statement had not been specified. (In other words, no IS-IS routes other than those matching any unrestricted version of the statement would be exported to the associated target.)

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target.

As with all export source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the export target statement. Setting a metric value in an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filter*s are provided, then any routes matching these filters that also match the **proto isis** statement parameters will be passed on to the export target for which this **proto isis** is an export source.

Any number of unique **proto isis** export sources can be used within the context of a single export target statement.

Defaults

By default, no **proto isis** export source exists.

Context

export source statement

Examples

Example 1

The following example exports into OSPF as Type 1 ASE all IS-IS routes with internal reachability.

```
export proto ospfase type 1 {
    proto isis {
        all;
    };
};
```

Example 2

The following example exports into RIP all IS-IS routes.

```
export proto rip {
    proto isis internal {
        all;
    };
    proto isis external {
        all;
    };
};
```

Example 3

The following example exports into OSPF as ASE type 2 routes, all IS-IS routes with external reachability.

```
export proto ospfase type 2 {
```

```
        proto isis external {
            all;
        };
    };
```

See Also

“Chapter 31 Route Importation” on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

“OSPF Overview” on page 45 in *Configuring GateD*

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

proto kernel

Name

proto kernel - filters on routes that are learned from the FIB

Syntax

```
proto kernel [ interface interface_list ] restrict ;
proto kernel [ interface interface_list ] [ metric metric ] [ noagg ]
    { [ route_filter [ restrict | ( metric metric ) ] ;
```

Parameters

interface_list - the list of interfaces on which the next hop for the matching route must reside in order for the matching route to be exported to the associated export target

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

Description

The **proto kernel** export source statement is nearly identical to the **proto static** export source statement. The only difference is that, rather than referring to routes statically configured by the routing daemon, this refers to routes which are learned from the FIB. Of course, it is not normally expected that routes will be learned from the FIB; however, there are two cases where this is possible. First, the routes could be remnant routes learned via the route socket after a software crash. In this case, it is strongly advised that the routes not be readvertised. Second, another administrative authority has directly added routes to the FIB. This is most commonly the case when GateD is used as a routing daemon on a UNIX workstation (in which case the FIB is the kernel RIB) and the super user adds routes via the route command. Because this is the only case where redistributing kernel routes is even remotely advisable, and the case is an historic one, the **proto kernel** export source should probably never be used.

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target.

Defaults

By default, no **proto kernel** export source exists.

Context

export source statement

Examples

The following example exports into OSPF as ASE all routes which are learned from the FIB.

```
export proto ospfase {
    proto kernel {
        all;
    };
};
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto ospf on page 657

proto ospfase on page 660

proto rip on page 663

proto static on page 669

proto ospf

Name

proto ospf - filter on routes learned via OSPF

Syntax

```
proto ospf [ type type ] [ tag tagvalue ] restrict ;
proto ospf [ type type ] [ tag tagvalue ] [ metric metric ] [ noagg ]
    { route_filter [ restrict | ( metric metric ) ] ;
} ;
```

Parameters

type **-** the type of AS External route, 1 or 2

tagvalue - an arbitrary number from 0 to 4294967295

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

Description

The **proto ospf** export source statement is used to match OSPF routes for exportation via one of the export target statements, described in this chapter. It only deals with routes learned directly from OSPF and does not consider routes that are AS external to OSPF. For these routes, see the **proto ospfase** export source statement.

In the **restrict** version of this *export_source_statement*, no OSPF routes that match the optional *tagvalue* (or no OSPF routes at all in the absence of the *tagvalue* parameter) will be exported via the associated *export_target_statement*. Strictly speaking, using the **restrict** version of this statement without the *tagvalue* parameter has no effect, as it would be the same behavior as if the statement had not been specified (that is, no OSPF routes other than those matching the unrestricted version of the statement would be exported to the associated target).

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target.

As with all export source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the export target statement. Setting a metric value in an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filter*s are provided, then any routes matching these filters which also match the **proto ospf** statement will be passed on to the export target for which this **proto ospf** is an export source.

Any number of unique **proto ospf** export sources can be used within the context of a single export target statement.

Defaults

By default, no **proto ospf** export source exists.

Context

export source statement

Examples

Example 1

The following example exports into RIP all OSPF routes.

```
export proto rip {
    proto ospf {
        all;
    };
};
```

Example 2

The following example exports into RIP all OSPF routes, except those received with a tag of 12345.

```
export proto rip {
    proto ospf {
        all;
    };
    proto ospf tag 12345 restrict;
};
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

proto aggregate on page 643

proto ospfase

Name

proto ospfase - filters on routes learned via OSPF that are AS External

Syntax

```
proto ospfase [ type type ] [ tag tag ] restrict ;
proto ospfase [ type type ] [ tag tag ] [ metric metric ] [ noagg ]
    { [ route_filter [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

type - the type of AS External route, 1 or 2

tagvalue - an arbitrary number from 0 to 4294967295

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

Description

The **proto ospfase** export source statement is used to match routes that have been propagated through OSPF as ASE for exportation via one of the export target statements, described in this chapter. To export routes learned as OSPF routes, see proto ospf.

In the **restrict** version of this *export_source_statement*, no OSPF ASE routes that match the optional *tagvalue* (or no OSPF routes at all in the absence of the *tagvalue* parameter) and are of the optional *type* will be exported via the associated *export_target_statement*. In the absence of the *type* parameter, this export source applies to both type 1 and type 2 ASE routes. For more information on the difference between type 1 and type 2 ASE routes, refer to “OSPF Overview” on page 45 in *Configuring GateD*. Strictly speaking, using the **restrict** version of this statement with neither the **tag** nor the **type** parameter has no effect, as it would be the same behavior as if the statement had not been specified. (In other words, no OSPF ASE routes other than those matching any unrestricted version of the statement would be exported to the associated target.)

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target.

As with all export source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the export target statement. Setting a metric value in an

export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filter*s are provided, then any routes matching these filters which also match the **proto ospfase** statement parameters will be passed on to the export target within which this **proto ospfase** is included.

Any number of unique **proto ospfase** export sources can be used within the context of a single export target statement.

Defaults

By default, no **proto ospfase** export source exists.

Context

export source statement

Examples

Example 1

The following example exports into RIP all OSPF ASE routes.

```
export proto rip {
    proto ospfase {
        all;
    };
};
```

Example 2

The following example exports into RIP all OSPF ASE routes, except those received with a tag of 12345.

```
export proto rip {
    proto ospfase {
        all;
    };
    proto ospfase tag 12345 restrict;
};
```

Example 3

The following example exports into RIP all OSPF ASE routes, but type 1 routes are exported with a higher RIP metric.

```
export proto rip {
    proto ospfase type 2 {
        all;
    };
```

```
        proto ospfase type 1 metric 2 {
            all;
        };
    };
```

Example 4

In this example, type 2 OSPF ASE routes that have a tag of 65535 are exported to AS 65535 via BGP, provided that they are not contributing to an aggregate.

```
    export proto bgp as 65535 {
        proto ospfase type 2 tag 65535 noagg {
            all;
        };
    };
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto rip on page 663

proto static on page 669

proto rip

Name

proto rip - filter on routes learned via RIP

Syntax

```
proto rip
    [ tag tagvalue | interface interface_list | gateway gateway_list ]
    restrict ;
proto rip
    [ interface interface_list | gateway gateway_list | tag tagvalue ]
    [ metric metric ] [ noagg ]
    { [ route_filter
      [ fromribs riblist ] [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

interface_list - the list of interfaces over which the matching route must have been received in order for it to be exported to the associated export target

gateway_list - the list of next hop RIP routers from which routes must have been learned in order for them to be exported via the associated export target

tagvalue - an arbitrary number from 0 to 65535

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

Description

The **proto rip** export source statement is used to match RIP routes for exportation via one of the export target statements, described in this chapter.

If the optional interface parameter is used, only routes learned over those interfaces matching the *interface_list* will be tested against the remaining filters and policy. Similarly, if the optional gateway parameter is used, then only those routes learned from routers matching the *gateway_list* will be considered for exportation.

In the **restrict** version of this *export_source_statement*, any RIP routes which match the optional **tag**, **interface**, or **gateway** parameters (or all RIP routes in the absence of said parameters) will be excluded from exportation via the associated *export_target_statement*.

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target. Similarly, the **tag** matching filter is used to specify that any route matching the other filters must also have this administratively set *tagvalue*.

As with all export source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the export target statement. Setting a metric value in an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filter*s are provided, then any routes matching these filters which also match the **proto rip** statement parameters will be passed on to the export target within which this **proto rip** is included. Note, however, that this export source cannot be used in conjunction with the **proto rip** export target statement. (Technically, exporting from RIP into RIP can be specified, but it has no effect, regardless of additional parameters.)

Any number of unique **proto rip** export sources can be used within the context of a single export target statement.

Defaults

By default, no **proto rip** export source exists.

Context

export source statement

Examples

Example 1

The following example exports into OSPF as ASE all RIP routes.

```
export proto ospfase {
    proto rip {
        all;
    };
};
```

Example 2

The following example exports into OSPF as ASE all RIP routes, except that those tagged with 12345 are exported with a higher cost.

```
export proto ospfase {
    proto rip tag 12345 metric 5000 {
        all;
    };
    proto rip {
        all;
    };
```

```
};
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto static on page 669

proto ripng

Name

proto ripng - filter on routes learned via RIPng

Syntax

```
proto ripng
    [ tag tagvalue | interface interface_list | gateway gateway_list ]
    restrict ;
proto ripng
    [ interface interface_list | gateway gateway_list | tag tagvalue ]
    [ metric metric ] [ noagg ]
    { [ route_filter [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

interface_list - the list of interfaces over which the matching route must have been received in order for it to be exported to the associated export target

gateway_list - the list of next hop RIPng routers from which routes must have been learned in order for them to be exported via the associated export target

tagvalue - an arbitrary number from 0 to 65535

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

Description

The **proto ripng export** source statement is used to match RIPng IPv6 routes for exportation via one of the export target statements, described in this chapter.

If the optional interface parameter is used, only routes learned over those interfaces matching the *interface_list* will be tested against the remaining filters and policy. Similarly, if the optional gateway parameter is used, then only those routes learned from routers matching the *gateway_list* will be considered for exportation.

In the **restrict** version of this *export_source_statement*, any RIPng routes which match the optional **tag**, **interface**, or **gateway** parameters (or all RIPng routes in the absence of said parameters) will be excluded from exportation via the associated *export_target_statement*.

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target. Similarly, the **tag** matching filter is used to specify that any route matching the other filters must also have this administratively set *tagvalue*.

As with all export source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the export target statement. Setting a metric value in an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filter*s are provided, then any routes matching these filters which also match the **proto ripng** statement parameters will be passed on to the export target within which this **proto ripng** is included. Note, however, that this export source cannot be used in conjunction with the **proto ripng** export target statement. (Technically, exporting from RIPng into RIPng can be specified, but it has no effect, regardless of additional parameters.)

Any number of unique **proto ripng** export sources can be used within the context of a single export target statement.

Defaults

By default, no **proto ripng** export source exists.

Context

export source statement

Examples

Example 1

The following example exports all RIPng routes to BGP peers in AS 201.

```
export proto bgp as 201 {
    proto ripng {
        all;
    };
};
```

Example 2

The following example exports into IS-IS external reachability all RIPng routes, except that those tagged with 12345 are exported with a higher cost.

```
export proto isis level 2 {
    proto ripng tag 12345 metric 5000 {
        all;
    };
    proto ripng {
        all;
```

```
        };
    };
```

See Also

"Chapter 31 Route Importation" on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630

export proto ospfnssa on page 633

export proto rip on page 635

proto aggregate on page 643

proto bgp on page 646

proto direct on page 649

proto isis on page 652

proto kernel on page 655

proto ospf on page 657

proto ospfase on page 660

proto static on page 669

proto static

Name

proto static - filter on statically configured routes

Syntax

```
proto static [ interface interface_list ] restrict ;
proto static [ interface interface_list ]
    [ metric metric ] [ noagg ]
    { [ route_filter
      [ fromribs riblist ] [ restrict | ( metric metric ) ] ; ] } ;
```

Parameters

interface_list - the list of interfaces on which the next hop for the matching route must reside in order for the matching route to be exported to the associated export target

metric - a metric with range appropriate to the protocol described in the associated export target

route_filter - route filter as described in the route filter document. See “Chapter 28 Route Filtering” on page 129 for more information.

noagg - specifies that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target

restrict - can be specified on the **proto** statement or any of its *route_filters*. It indicates that any matching routes are not to be exported.

fromribs *riblist* **-** specifies the RIBs from which routes matching this route_filter will be exported. See “**fromribs**” on page 641 for more information.

Description

The **proto static** export source statement is used to match routes which have been statically configured for exportation via one of the export target statements, described in this chapter.

If the optional interface parameter is used, only routes associated with those interfaces matching the *interface_list* will be tested against the remaining filters and policy. Since static routes are not learned, per se, the match against interface may seem slightly different than that of other *export_source_statements* (though it technically is not). The static routes must have next hops that reside on a matched, directly connected interface in order for them to match this filter.

In the **restrict** version of this *export_source_statement*, any static routes that match the optional interface parameter (or all static routes in the absence of said parameter) will not be exported via the associated *export_target_statement*.

The **noagg** matching filter is used to specify that any route matching the other filters has the additional caveat that it must not be contributing to an aggregate in order for it to be passed on to the export target.

As with all export source statements, the optional *metric* parameter must be a valid metric for the associated protocol in the *export_target_statement*. Setting a metric value in an export source overrides any metric set on the export target, thus allowing for additional scope.

If any *route_filters* are provided, then any routes matching these filters which also match the **proto static** statement parameters will be passed on to the export target within which this **proto static** is included.

Any number of unique **proto static** export sources can be used within the context of a single export target statement.

Defaults

By default, no **proto static** export source exists.

Context

export source statement

Examples

Example 1

The following example exports into OSPF as ASE all static routes configured on this router.

```
export proto ospfase {
    proto static {
        all;
    };
};
```

Example 2

The following example exports into RIP all statically configured routes, except those with next hops on the interface eth0.

```
export proto rip {
    proto static interface eth0 restrict;
    proto static {
        all;
    };
};
```

See Also

“Chapter 31 Route Importation” on page 137 in *Configuring GateD*

export proto bgp on page 623

export proto isis on page 627

export proto ospfase on page 630