Chapter 32
Route Exportation

32.1 Route Exportation Overview

The **import** statement controls which routes that are received from other systems are used by GateD, and the **export** statement controls which routes are advertised by GateD to other systems. Like the **import** statement, the syntax of the **export** statement varies slightly per protocol. The syntax of the **export** statement is similar to the syntax of the **import** statement, and the meanings of many of the parameters are identical. The main difference between the two is that while route importation is just controlled by source information, route exportation is controlled by both destination and source.

The outer portion of a given **export** statement, the **export** target statement, specifies the destination of the routing information you are controlling. The middle portion, the **export** source statement, restricts the sources of importation that you wish to consider. The innermost portion is a route filter used to select particular routes.

The **export** command specifies the policy for dynamically advertising routes from GateD to other systems. Any number of **export** commands can be given, provided that each **export** target statement is unique.

The **export** target statement specifies by which protocol the route will be advertised and how it will be advertised. It can limit to which neighbors of a given protocol the routes are sent and modify a route's attributes before it is sent. Obviously, the route attributes that exist or can be set are dependant on the outgoing protocol. (For example, you cannot set an AS Path for an OSPF route.) Only one **export** target statement can be given per **export** command.

The union of all **export** source statements represents a filter that determines the routes that will be exported. The filter will match in order and is succeeded by an implied restriction, so anything not matched by one of the **export** source statements will not be advertised. As a result, the following two statements are identical:

```
export proto rip restrict;
export proto rip { };
```

The same statements about order and restriction apply to multiple instances of the **export** command as a whole. So, if no **export** commands are given, then default behavior will ensue; however, if even one **export** statement is given, then no other exporting will occur.

32.1.1 Export Inheritance

The following parameters can be specified at multiple places within a given **export** statement:

- **restrict**
- **metric**

In the case of **restrict**, placing it on an **export target** statement restricts exportation of all routes matching the **export target** statement. Including it on an **export source** statement restricts exportation of all routes matching the **export source** statement. Including it on a *route_filter* restricts exportation of all routes that match that route filter.

In the case of export **metric**, the most specific instance of a **metric** value is assigned to a route. Thus, any **metric** value specified on an **export target** statement is used only if neither the **export source** statement nor the *route_filter* that matches a route specifies a value. The value of the most specific matching *route_filter* is used. If neither the **export target** statement, the matching **export source** statement, nor the matching *route_filter* specifies a metric value, then the default metric value for the protocol is used.

32.2 Export Syntax

export target statements are one or more of the following:

```
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
    } ;

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
    };

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
    };

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

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

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

export_source_statements are one or more of the following:

```
    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 ) ] ; ] } ;

    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 ) ] ; ] } ;

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 ) ] ; ] } ;

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

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

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

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

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

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

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

32.3 Export Defaults

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

32.4 Export Examples

32.4.1 Exporting to BGP

Example 1

This example configures BGP to export all active BGP routes learned from AS 202 and three static routes and any of their static subnets to BGP peers in AS 201.

Note: The default policy of exporting all 'direct' routes is no longer in effect if export policy is given.

```
export proto bgp autonomoussystem 201 {
    proto bgp autonomoussystem 202 {
        all;
    };
    proto static {
```

```
            223.1.0/24;
            223.2.0/24;
            223.3.0/24;
        };
    };
```

Example 2

This example exports all IPv4 and IPv6 static routes and all routes learned from AS 202, to AS 201 (assuming MPBGP is enabled).

```
    export proto bgp as 201 {
        proto bgp as 202 {
            all;
        };
        proto static {
            all ;
        } ;
    } ;
```

32.4.2 Exporting to RIP

This example configures all static and direct routes to be exported into the RIP protocol with tag 2112.

Note: The default policy of exporting all 'direct' routes is no longer in effect if export policy is given.

```
    export proto rip tag 2112 {
        proto static {
            all;
        };
        proto direct {
            all;
        };
    };
```

32.4.3 Exporting to RIPng

This example configures all static and direct IPv6 routes to be exported into the RIPng protocol with tag 2112.

Note: The default policy of exporting all 'direct' routes is no longer in effect if export policy is given.

```
    export proto ripng tag 2112 {
        proto static {
            all;
```

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

32.4.4 Exporting to OSPF ASE and NSSA

Example 1

This example configures all static routes to be exported into OSPF ASE (Type-5 LSAs).

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

Example 2

A similar configuration can be used to export all static routes into OSPF NSSA (Type-7 LSAs).

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

32.4.5 Exporting to ISIS

This example configures all IPv4 and IPv6 static routes to be exported into ISIS external reachability. The routes shall be originated in Level 2 LSPs only.

```
    export proto isis level 2 {
        proto static {
            all;
        };
    };
```

32.4.6 Exporting RIP Routes

This example configures certain RIP routes to be exported into OSPF ASE using a metric of 1 in the ASE LSAs. The 192.168.10/24 route shall be exported using a metric of 2.

```
    export proto ospfase metric 1 {
        proto rip {
            192.168.10/24 metric 2;
```

```
            192.168.11/24 restrict;
            192.168.12/24;
        };
    };
```

32.4.7 Exporting OSPF Routes

This example configures all OSPF routes to be exported into RIP.

Note: The OSPF routes that will match are OSPF internal routes, not ASE-based routes.

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

32.4.8 Exporting Routes from Non-routing Protocols

The 'aggregate' protocol is not a routing protocol but a classification of routes configured by the user. The following example exports all aggregate routes (see the aggregate clause) into OSPF ASE.

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

32.4.9 Exporting by AS Path

This example uses a simple AS Path regular expression to export routes with an AS Path of 204, 203 into RIP, placing the tag 2112 in the RIP tag field.

```
export proto rip tag 2112 {
    proto bgp aspath “(204 203)” origin any {
        all;
    };
};
```

32.4.10 Exporting by Route Tag

Where supported by the routing protocol, export policy may use route tags. The following example exports all RIP routes with tag 2112 into OSPF ASE.

```
export proto ospfase {
    proto rip tag 2112 {
        all;
    };
```

```
};
```