10 Route Importing, Exporting, and Filtering

During normal operation of GateD, all routes encountered by a protocol are learned and stored in the routing table (assuming other restrictions are not in place). By adding import and export statements, you can control what routes are learned or advertised from the GateD routing table.

Import statements control what routes advertised by other routers in the network are added to the GateD routing table. Export statements control what routes currently in the GateD routing table are advertised to other routers in the network.

♦ Important Note ♦

When constructing import and export statements, it is important to remember that once a statement has been set, any route not matching the criteria of the statement will be marked as unusable by the routing table. For example, if an import statement is created that limits what type of OSPF routes should be imported, all routes that do not match the specified type will be rejected. When using the import and export statement, thorough knowledge of the network environment is required to prevent the rejection of legitimate routes.

Both import and export statements employ route filters. In statement examples, route filter options are abbreviated with the following syntax:

route_filter

For details on the complete statement for constructing route filters, see *Route Filters* on page 10-2.

Route Filters

Route filters allow you to specify criteria that an incoming or outgoing route must match before it is subjected to the operation of the statement that contains the filter (for example, importing and exporting).

When no route filtering is specified, all routes from the specified source will match that statement. If any filters are specified, only routes that match the specified filters will be imported or exported

The route filter syntax is as follows:

network **[exact | refines | between** *number* **and** *number* **] ;**
network **mask** *mask* **[exact | refines | between** *number* **and** *number* **] ;**
network **masklen** *number* **[exact | refines | between** *number* **and** *number* **] ;**
all ;
default ;
host *host ;*

The *network*, *network* **mask** *mask*, and *network* **masklen** *number* refer to three ways of distinguishing the network mask by using a natural mask for the network address (255.255.0.0), a network address with a specifically set mask (1.1.1.0 **mask** 255.255.255.0), or a network with a specified mask length (1.1.1.0 **masklen** 25). Only one statement is required, but all can be specified.

Parameter descriptions for the *route_filter* variations are as follows:

exact	This parameter specifies that the mask of the destination must match the supplied mask exactly. This is used to match a network, but no subnets or hosts of that network.
refines	This parameter specifies that the mask of the destination must be more specific (i.e., longer) than the filter mask. This is used to match subnets or hosts of a network, but not the network.
between ... and ...	This parameter specifies that the mask of the destination must be as or more specific (i.e., as long as or longer) than the lower limit (the first number parameter) and no more specific (i.e., as long as or shorter) than the upper limit (the second number parameter). These values should be expressed as mask length numbers rather than IP address. See *Route Filter Example* on page 10-3 below for an example.
all	This parameter specifies the entry matches anything. It is equivalent to 0.0.0.0 **mask** 0.0.0.0
default	This parameter specifies the default route. To match, the address must be the default address and the mask must be all zeros. This is equivalent to 0.0.0.0 **mask** 0.0.0.0 **exact**.
host *host*	This parameter specifies a specific host. To match, the address must exactly match the specified host and the network mask must be a host mask (i.e., all ones). This is equivalent to *host* **mask** 255.255.255.255 **exact**

Route Filter Example

The following is an example of a route filter.

```
192.5.6.0 between 16 and 24 ;
```

In this statement:

- 192.5.6.0 is the network address that the filter looks for.
- The **between 16 and 24** modifier determines that any route specifying network 192.5.6.0 must have a mask length between **16** (225.225.0.0) and **24** (225.225.225.0) in order to satisfy the criteria of the route filter.
- Any route that does not match these two criteria is not entered into the routing table.

Route Importation

Importation of routes from routing protocols and the installation of the routes in GateD's routing database can be controlled by the import statement. The format of an import statement varies depending on the source protocol.

Specifying Preferences

In all import statements, one of two keywords is used to control how routes compete with other protocols:

preference *preference*	Specifies the preference value used when comparing this route to other routes from other protocols. The route with the lowest preference available becomes the active route, is installed in the forwarding table, and is eligible to be exported to other protocols. The default preferences are configured by the individual protocols.
restrict	Indicates that these routes are not to be considered as contributors to the routing table. The specified protocol may be any of the protocols supported by GateD.

In some cases, the routes filtered by import statements are not installed in the routing table. In other cases, the routes are installed with a negative preference. This prevents them from becoming active so they will not be installed in the forwarding table or exported to other protocols.

Importing Routes from BGP

The statement format used for importing routes from BGP is shown below. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

```
import proto bgp autonomoussystem autonomous_system [ aspath-opt ]
[ preference preference ] {
        route_filter [ restrict | ( preference preference ) ] ;
} ;

import proto bgp autonomoussystem autonomous_system [ aspath-opt ] restrict ;
```

It is possible for several BGP import clauses to match a given update. If more than one clause matches, the first matching clause will be used; all later matching clauses will be ignored. For this reason, it is generally desirable to order import clauses from most specific (gateway) to least specific (protocol).

BGP stores any routes that are rejected with a negative preference. A negative preference prevents a route from becoming active and exported to other protocols.

The following section defines the terms shown in the import statement.

import proto bgp autonomoussystem *autonomous_system*

This command specifies the AS number from which routes are accepted. If used with the **preference** specifier, then a route filter (with its own **restrict** or **preference** modifier) can be added. See *Route Filters* on page 10-2 for more specific information on route filters.

aspath-opt

Within a BGP group statement, the *aspath-opt* statement is used to generate community attributes. See Chapter 8 of this manual for more information on community attributes.

Importing Routes from BGP Using AS Path Regular Expressions

BGP supports controlling propagation by the use of an AS path regular expressions. An AS path is a list of autonomous systems that routing information has passed through to get to this router, and an indicator of the origin of the routing information. This information can be used to prefer one path to a destination network over another.

The following shows the statements used for importing routes from BGP using AS path regular expressions. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

import proto bgp aspath *aspath_regexp* **origin any | ([igp] [egp] [incomplete])**
[*aspath-opt* **] [preference** *preference***} {**
route_filter **[restrict] [(preference** *preference* **)] ;**
} ;

import proto bgp aspath *aspath_regexp* **origin any | ([igp] [egp] [incomplete])**
[*aspath-opt* **] [restrict };**

The following describes the components of the above statement.

import proto bgp aspath *aspath_regexp*

An import statement specifies the *aspath_regexp* used as the criteria for the statement. The *aspath_regexp* is the valid range of AS numbers (1 through 65534) with the addition of wild card expressions and operators. Available expressions and operators are:

.	Any valid member of the alphabet
*****	Matches zero or more of the preceding element or expression
+	Matches one or more of the preceding element or expression
?	Matches zero or one occurrence of the preceding element or expression.
" " (and)	Any sequence of elements or expressions separated by a space.
"|" (or)	Any sequence of elements or expressions separated by the **|** symbol.
{m,n}	The expression {m,n} is used to provide a limited set of repeated AS numbers in the regular expression where **m** is the minimum amount of elements and **n** is the maximum amount of elements. In addition to the {m,n}, **m** alone can be used to define repetitions of the AS number. Specifying **m** alone means to match exactly that number of repetitions of the preceding AS number.

Examples of this syntax can be seen in *Examples of AS Path Regular Expressions* on page 10-6 below.

aspath-opt

Within a BGP group statement, the *aspath-opt* statement is used to generate community attributes. See Chapter 8 of this manual for more information on community attributes.

origin any | ([igp] [egp] [incomplete])

Each autonomous system through which a route passes prepends its AS number to the beginning of the AS path. The origin information details the completeness of AS path information. An origin of **igp** indicates the route was learned from an interior routing protocol and is most likely complete. An origin of **egp** indicates the route was learned from an exterior routing protocol that does not support AS paths and the path is most likely not complete. When path information definitely is not complete, an origin of **incomplete** is used. The **origin** parameters (**igp**, **egp**, and **incomplete**) can be used singly or in combination with each other.

Examples of AS Path Regular Expressions

The following are some examples of regular expressions.

(808{2,})	Match two or more AS numbers that are identical to 808.
(808{1,2} .*)	Match all AS paths that start with one or two 808s and followed by any number of AS numbers (or none). This is equivalent to (808 .*).
(808{2,4} .+)	Match only if the AS path has between two and four of the given AS number followed by any number of AS numbers, but at least one more. This is equivalent to (808{2} .+).
(808{2})	Match exactly two 808s on the AS path.

AS path regular expressions fit into the BGP import statement as shown:

```
import proto bgp aspath (808{1}.+) origin egp restrict ;
```

In this statement:

- BGP routes from AS 808 that originated in an exterior routing protocol are excluded from being imported.
- The first AS number in the AS path must be 808.

Importing Routes from RIP

The following is the statement used for importing a route from RIP. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

```
import proto rip [( interface interface_list ) | (gateway gateway_list )]
[ preference preference ] {
        route_filter [ restrict | ( preference preference ) ] ;
} ;

import proto rip [( interface interface_list ) | (gateway gateway_list )] [ restrict ] ;
```

The components of the RIP import statement are described below.

import proto rip

This statement imports all routes using RIP or RIP version 2.

interface *interface_list*

This token and modifier set a specific interface from which RIP routes are allowed to be imported.

gateway *gateway_list*

This token and modifier set a specific gateway from which RIP routes are allowed to be imported.

route_filter

When specifying routes to be imported, a route filter can be configured. See *Route Filters* on page 10-2 for more specific information on route filters.

Example

The following is an example of an import statement for RIP:

```
import proto rip preference 50 {
        192.5.6.0 between 28 and 30 restrict;
} ;
```

In this statement:

- All routes using RIP are imported with preference of 50.
- A filter is configured that restricts the importation of any route from a network with address 192.5.6.0 and a mask length that falls between 28 and 30 .

Importing Routes from OSPF

The following shows the statement for importing routes from OSPF. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

```
import proto ospfase [ tag ospf_tag ] [ preference preference ] {
        route_filter [ restrict | ( preference preference ) ] ;
} ;

import proto ospfase [ tag ospf_tag ] restrict ;
```

Due to the nature of OSPF, only the importation of external autonomous system (ASE) routes may be controlled. OSPF intra- and inter-area routes are always imported into the GateD routing table with a preference of 10.

It is only possible to restrict the importation of OSPF ASE routes when the router is functioning as an AS border router. This is accomplished by specifying an export **ospfase** clause. See *Route Exportation* on page 10-9 for more information on exporting routes.

Routes that are rejected by import policies are stored in the table with a negative preference.

The following section describes the variables in the statement shown above.

import proto ospfase

This statement specifies that all OSPF ASE routes are imported into the routing table.

tag *ospf_tag*

This token specifies that only OSPF ASE routes that match the specified tag will be imported into the routing table. For information on setting OSPF tags, see Chapter 7 of this manual.

route_filter

When specifying routes to be imported, a route filter can be configured. See *Route Filters* on page 10-2 for more specific information on route filters.

Example of Import Statement for OSPF

The following is an example of an import statement for OSPF routes:

```
import proto ospfase restrict ;
```

In the above statement, all OSPF external routes are restricted and not installed in the routing table.

Route Exportation

The import statement controls which routes 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 import and export statements is that route importation is controlled by source information, but route exportation is controlled by both source *and* destination information.

The first section of a given export statement specifies the destination of the routing information that is to be controlled (defined by the **export** token). The middle portion controls what route sources should be considered for route exportation (defined by the expression *export_list*). The last portion is a route filter used to select individual routes.

Specifying Metrics

The most specific route metric is the one applied to the route at the export level. The values that may be specified for a metric depend on the destination protocol that is referenced by this export statement.

metric *metric*

Specifies the metric to be used when exporting to the specified destination.

restrict

Specifies that no route matching the statement should be exported. If this is specified on the destination portion of the export statement, it means that nothing at all should be exported to this destination. If specified on the source portion, it specifies that nothing from this source should be exported to this destination. If specified as part of a route filter, it specifies that the routes matching that filter should not be exported.

Exporting to BGP

The following is the statement used for exporting a route to BGP. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

export proto bgp as *autonomous_system* **[** *aspath-opt* **] [metric** *metric* **] {**
export_list **;**
} ;

export proto bgp as *autonomous_system* **restrict ;**

Exportation to BGP is controlled by autonomous system. BGP metrics are a range from 0 to 65535, with 0 being the most attractive.

If no export policy is specified, only routes to attached interfaces will be exported. If any policy is specified the defaults are overridden. It is necessary to explicitly specify everything that should be exported.

The following section defines the terms used in the above statements.

export proto bgp as *autonomous_system*

This statement allows you to export to BGP using the autonomous system number as a criteria.

aspath-opt

Within a BGP group statement, the *aspath-opt* statement is used to generate community attributes. See Chapter 8 of this manual for more information on community attributes.

export_list

The export list is a definition specifying the source of the routing information to be exported. The following is the statement syntax for configuring what BGP routes are exported. Two variations are shown, one for each of the preference options (**metric** and **restrict**).

proto bgp autonomoussystem *autonomous_system* **[metric** *metric* **] {**
route_filter **[restrict | (metric** *metric* **)] ;**
} ;
proto bgp autonomoussystem *autonomous_system* **restrict ;**

This syntax is used in place of the *export_list* token.

Example

The following is an example of an export statement for advertising routes to BGP.

```
export proto bgp as 10 metric 50 {
        proto bgp autonomoussystem 15 ;
} ;
```

In the example above, all routes from autonomous system 15 are exported to autonomous system 10 with a metric of 50.

Exporting to RIP

The following is the statement used for exporting a route to RIP. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

```
export proto rip [ ( interface interface_list ) | (gateway gateway_list ) ]
[ metric metric ] {
        export_list ;
} ;

export proto rip [ ( interface interface_list ) | (gateway gateway_list ) ] restrict ;
```

Exportation to RIP is controlled by protocol, interface, or gateway. If more than one is specified, they are processed from most general (protocol) to most specific (gateway).

It is not possible to set metrics for exporting RIP routes into RIP. Attempts to do this are silently ignored.

If no export policy is specified, RIP and interface routes are exported into RIP. If any policy is specified, the defaults are overridden. Like import statements, it is necessary to explicitly specify everything that should be exported.

When exporting routes from other protocols, it is important to specify a metric on the export statement or in the route filters. Unless this is done, the value specified in the default metric is used. The default metric is **16** (unreachable).

To announce routes that specify a next hop of the loopback interface through RIP (i.e., static and internally generated default routes), it is necessary to specify the metric at some level in the export clause. Just setting a default metric for RIP is not sufficient. This is a safeguard to verify that the announcement is intended.

The following describes the components of an export statement for RIP.

export proto rip

This statement exports routes using RIP or RIP version 2.

interface *interface_list*

This token and modifier set a specific interface from which RIP routes are allowed to be exported.

gateway *gateway_list*

This token and modifier set a specific gateway from which RIP routes are allowed to be exported.

export_list

The export list is a definition specifying the source of the routing information to be exported. The following is the statement syntax for configuring what RIP routes are exported. Two variations are shown, one for each of the preference options (**metric** and **restrict**).

```
proto rip [ ( interface interface_list ) | (gateway gateway_list ) ] [ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
} ;

proto rip [ ( interface interface_list ) | (gateway gateway_list ) ] restrict ;
```

route_filter

When specifying routes to be exported, a route filter can be configured. See *Route Filters* on page 10-2 for more specific information on route filters.

Example

The following is an example of an export statement for RIP routes:

```
export proto rip gateway 210.1.3.6 {
        proto direct all ;
        proto static all ;
} ;
```

In this statement all direct and static routes are exported to gateway 210.1.3.6.

Exporting to OSPF

The following is the statement used for exporting a route to OSPF. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

```
export proto ospfase [ type 1 | 2 ] [ tag ospf_tag ] [ metric metric ] {
        export_list ;
} ;
export proto ospfase [ type 1 | 2 ] [ tag ospf_tag ] restrict ;
```

It is not possible to export routes from the GateD routing table into OSPF. It is only possible to export from the GateD routing table into OSPF ASE routes. It is also not possible to control the propagation of OSPF routes within the OSPF protocol.

There are two types of OSPF ASE routes, type 1 and type 2. The default type is specified by the defaults subclause command in the OSPF statement. This may be overridden by a specification on the export statement. For more information on OSPF route types, see Chapter 7 of this manual.

OSPF ASE routes also have the provision to carry a tag. This is an arbitrary 32-bit number that can be used on OSPF routers to filter routing information. The default tag is specified by the ospf defaults command in the OSPF statement. This may be overridden by a tag specified on the export statement. For more information on OSPF route types, see Chapter 7 of this manual.

The following defines the variables in this export statement.

export proto ospfase

This command allows you to export all ASE routes to OSPF.

type 1 | 2

This modifier allows you to choose either type 1 or 2 ASE routes. See Chapter 7 for more information on OSPF route types.

tag *ospf_tag*

This command allows you to specify an OSPF tag. RIP and OSPF route tags can also be specified in the protocol statement. See Chapter 6 (RIP) and Chapter 7 (OSPF) for more details.

export_list

The export list is a definition specifying the source of the routing information to be exported. The following is the statement syntax for configuring what OSPF routes are exported. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**).

```
proto ospf | ospfase [ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
} ;
proto ospf | ospfase restrict ;
```

Both OSPF and OSPF ASE routes can be exported into other protocols.

Example

The following is an example of an export statement for OSPF:

```
export proto ospfase type 1 tag 156 restrict ;
```

In this example, all ASE routes of type 1 and having an OSPF tag of **156** are restricted from being exported.

Exporting Routes from Non-Routing Protocols

Non-routing protocols create routes without reference to IGP or EGP protocols (for example, RIP and BGP). Routes created using non-routing protocols can be exported, and have their own statement syntax. Non-routing protocols are covered in Chapters 9 and 11 of this manual.

Non-routing protocols can be exported based on two criteria: interface and protocol. These can be used together in the same configuration file.

Non-routing with Interface

The following is the statement for exporting non-routing protocol routes based on the source interface. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**):

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

The following is a description of the non-routing protocols that can be exported.

direct

Routes to directly attached interfaces.

static

Static routes specified in a static clause.

kernel

Routes learned from the kernel are installed in the GateD routing table with a protocol of **kernel**. These routes may be exported by referencing this protocol. For information on the kernel statement, see Chapter 9 of this manual.

Non-routing by Protocol

The following is the statement for exporting a route based on the non-routing protocol. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**):

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

These protocols may only be exported by protocol. The following section describes the terms used in the above statement.

default

Refers to routes created by the **gendefault** option in the global statement of the configuration file.

aggregate

Refers to routes synthesized from other routes when the **aggregate** and **generate** statements are used. See Chapter 11 for more information on route aggregation.

Example

The following are examples of export statements for non-routing protocols.

```
export proto ospfase {
        proto direct interface 182.5.6.7 metric 10 {all; } ;
        proto aggregate restrict ;
};
```

In this example, direct routes corresponding to interface 182.5.6.7 are exported to OSPF with with a metric of 10, while aggregated routes are restricted from being exported.

Exporting by AS Path

All BGP routes are assigned an AS path when they are added to the routing table. Route exportation can be determined based on AS path information. Below is the statement for exporting routes based on AS path. Two variations are shown, one for each of the preference options (**metric** and **restrict**):

```
proto proto | all aspath aspath_regexp origin any | ( [ igp ] [egp ] [ incomplete ] )
[ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
} ;

proto proto | all aspath aspath_regexp origin any | ( [ igp ] [egp ] [ incomplete ] ) restrict ;
```

The command **proto** *proto* | **all** is used to determine what route protocol types (e.g., BGP, RIP, default) specified as a destination are examined for possible exportation. Using **all** means every protocol type is eligible.

For information on the *aspath_regexp* syntax and usages, see *Importing Routes from BGP Using AS Path Regular Expressions* on page 10-5.

Example

The following is an example of an export statement using the AS path:

```
export proto ospf {
        proto all aspath (808{1}.+) origin igp metric 10 {all; };
} ;
```

In this example, all protocols that originated from an IGP source and have AS 808 as the first term in the AS path will be exported to OSPF as ASE routes, with an assigned metric of 10.

Exporting by Route Tag

Both OSPF and RIP version 2 currently support tags. Tags are specified in the configuration statements of the protocol in question. All other protocols always have a tag of zero. Below is the statement for exporting routes based on route tags. Two variations are shown, one for each of the preference options (*route_filter* and **restrict**):

```
proto proto | all tag tag [ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
} ;
proto proto | all tag tag restrict ;
```

The command **proto** *proto* | **all** is used to determine what route protocol types (in this case, RIP and OSPF) specified as a source are examined for possible exportation. Using **all** means both protocol types are eligible.

Example

The following is an example of an export statement based on route tags.

```
export proto ospfase {
        proto rip tag 156 restrict ;
        proto rip { all ; };
};
```

In this example, all RIP routes except those with a route tag of 156 are exported to OSPF as ASE routes.