Chapter 14
Kernel Interface

background

Name

`background` - controls the background processing of routes

Syntax

```
background [ limit number ] [ priority ( flash | higher | lower ) ] ;
```

Parameters

`limit` *`number`* - **`limit`** specifies the number of routes that may be processed during one batch. *`number`* must be in the range 0 to 4,294,967,295. Note that *`number`* is an unsigned integer. Specifying -1 results in the maximum 32-bit unsigned number, which is mentioned above.

`priority ( flash | higher| lower )`

`priority` specifies the priority of the processing of batches of kernel updates in relationship to the flash update processing. The default is **`lower`**, which means that flash updates are processed first. To process kernel updates at the same priority as flash updates, specify **`flash`**. To process kernel updates at a higher priority, use **`higher`**.

Description

Because only interface routes are normally installed during a flash update, the remaining routes are processed in batches in the background, that is, when no routing protocol traffic is being received.

Defaults

Process up to 120 routes at a time, processing flash updates first.

```
kernel { background limit 120 priority lower ; } ;
```

Context

`kernel` statement

Examples

Example 1

Process 100 routes at a time.

```
kernel { background limit 100; } ;
```

Example 2

Process kernel updates first.

```
kernel { background priority higher ; } ;
```

See Also

"Chapter 18 Kernel Interface" on page 95 of *Configuring GateD*

`flash` on page 329

`kernel` on page 327

flash

Name

`flash` - used to control the number and type of routes processed during a flash update

Syntax

```
flash [ limit number ] [ type ( interface | interior | all ) ] ;
```

Parameters

`limit` *`number`* - **`limit`** specifies the maximum *`number`* of routes that may be processed during one **`flash`** update. The default is 20. A value of -1 will cause all pending route changes of the specified type to be processed during the flash update. *`number`* is essentially unlimited. It must be in the range 0 to 4,294,267,295. Note that *`number`* is unsigned. Specifying **`flash limit`** *`-1`* **`all`** causes all routes to be installed during the flash update; this mimics the behavior of prior versions of GateD that did not support this kernel option.

`type ( interface | interior | all )` - **`type`** specifies the type of routes that will be processed during a **`flash`** update (see "Description" below). **`interior`** specifies that interior routes will be installed. **`all`** specifies the inclusion of exterior routes as well. The default is interface, which specifies that only interface routes will be installed during a flash update.

Description

When routes change as a result of kernel or protocol module activity, the process of notifying the GateD protocol module is called a flash update. The kernel forwarding table interface is the first to be notified. The flash process is concerned with three types of routes:

- interface routes: routes defined by an interface statement
- interior routes: routes within the domain
- exterior routes: routes exterior to the domain

A flash update results from protocol activity. The intent of this mechanism is to limit the number of routes installed during a flash update -- suspending the current protocol module until the flash completes. Typically, only up to 20 interface routes are normally installed during a flash update. The remaining routes are processed in batches in the background, that is, when no routing protocol traffic is being processed. The **`flash`** option is used to control the number and type of routes processed during a flash update.

Defaults

```
kernel { flash limit 20 type interface; };
```

Context

`kernel` statement

Examples

`flash` update up to 40 routes at a time. All types of routes are updated:

```
kernel { flash 40 type all ; } ;
```

See Also

`background` on page 327

“Chapter 18 Kernel Interface” on page 95 of *Configuring GateD*

`kernel` on page 327

kernel

Name

kernel - controls how GateD interfaces with the kernel

Syntax

```
kernel {
    [ options
        [ nochange ]
        [ noflushatexit ]
    ; ]
    [ remnantholdtime time ; ]
    [ routes number ; ]
    [ flash
        [ limit number ]
        [ type ( interface | interior | all ) ]
    ; ]
    [ background
        [ limit number ]
        [ priority ( flash | higher | lower ) ]
    ; ]
    [ traceoptions
          [ tracefile [ replace ]
          [ size tracesize [ k | m ] files tracefiles ] ] [ nostamp ]
          [ trace_global_options | trace_protocol_options |
            trace_protocol_packets ]
          [ except ( trace_global_options | trace_protocol_options |
            trace_protocol_packets ) ]
       ; ]
} ;
```

Parameters

options

remnantholdtime

routes *number*

flash

background

traceoptions

Description

Although the kernel interface is not technically a routing protocol, it has many characteristics of one, and GateD handles it similarly. The routes GateD chooses to install in the kernel forwarding table are those that will actually be used by the kernel to forward packets.

The add, delete, and change operations that GateD must use to update the typical kernel forwarding table take a non-trivial amount of time. The time used does not present a problem for older routing protocols (such as RIP), which are not particularly time-critical and do not easily handle large numbers of routes anyway. The newer routing protocols (such as OSPF and BGP) have stricter timing requirements and are often used to process many more routes. The speed of the kernel interface becomes critical when these protocols are used.

To prevent GateD from locking up for significant periods of time while installing large numbers of routes (up to a minute or more has been observed on real networks), the processing of these routes is now done in batches. The size of these batches can be controlled by the tuning parameters described in the **`routes`**, **`flash`**, and **`background`** sections, but normally the default parameters will provide the proper functionality.

During normal shutdown processing, GateD deletes all the routes it has installed in the kernel forwarding table, except for those static routes marked with **`retain`**. Optionally, GateD can leave all routes in the kernel forwarding table by not deleting any routes, using **`noflushatexit`**. This option is useful on systems with large numbers of routes because it eliminates the need to re-install the routes when GateD restarts, which can greatly reduce the time it takes to recover from a restart.

Options **`nochange`** and **`noflushatexit`** are disabled by default.

Defaults

```
kernel {
    flash limit 20 type interface;
    routes -1;       # no limit on kernel-installed routes
    background limit 120 priority lower;
    remnantholdtime  180;
    traceoptions none;
} ;
```

Context

global

Examples

```
kernel {
    options nochange;
    flash limit 40 type all;
    routes 5000;
    background limit 500 priority flash;
    remnantholdtime 7:0;
} ;
```

See Also

`background` on page 327

"Chapter 18 Kernel Interface" on page 95 of *Configuring GateD*

`flash` on page 329

options

Name

`options` - specifies the kernel statement options

Syntax

`options ( nochange | noflushatexit ) ;`

Parameters

`nochange` - On systems supporting the routing socket, **`nochange`** ensures that change operations will not be performed (only deletes and adds will). **`nochange`** is useful on early versions of the routing socket code where the change operation was broken.

`noflushatexit` - During normal shutdown processing, GateD deletes all routes that do not have a retain indication from the kernel forwarding table. **`noflushatexit`** prevents route deletions at shutdown.

During a GateD shutdown/restart sequence, it may be desirable to keep the routes that existed at the time of the GateD shutdown in the kernel forwarding table. This enables the router to continue forwarding packets while GateD is being restarted.

After a restart, the protocol modules are given a short amount of time (currently 3 minutes) to determine their routes. After 3 minutes, all residual routes not re-established by the protocol modules are flushed.

There are four conditions under which GateD does not flush a route:

- interface routes
- static routes with **`retain`** keyword
- **`noflushatexit`**
- routes with static bit set

`noflushatexit` is handy on systems with thousands of routes. Upon startup, GateD will notice which routes are in the kernel forwarding table and not add them back.

Description

`options` specifies the kernel statement options.

Defaults

`nochange` and **`noflushatexit`** are disabled.

Context

`kernel` statement

Examples

Prevent FIB change calls.

```
kernel { options nochange; } ;
```

See Also

"Chapter 18 Kernel Interface" on page 95 of *Configuring GateD*

`kernel` on page 327

`remnantholdtime` on page 336

remnantholdtime

Name

`remnantholdtime` - sets the default remnant hold time

Syntax

`remnantholdtime` *`time`*

Parameters

`time` - The remnant hold time must be between 0 and 900 seconds. It can be expressed as *`seconds`* or *`minutes:seconds`*.

`seconds` - time in seconds

`minutes`:*`seconds`* - time in minutes and seconds, separated by a colon

Description

When GateD starts up, it reads the kernel forwarding table and installs corresponding routes into GateD's routing table. These routes, with the exclusion of interface routes and routes configured via the UNIX route command, are called remnants. Remnant routes are timed out after the specified interval, or as soon as a more attractive route is learned. This method allows forwarding to occur while the routing protocols start learning routes. Setting **`remnantholdtime`** to 0 disables this timer, causing remnant routes to be kept indefinitely.

Defaults

Delete remnant routes after 180 seconds:

```
kernel {remnantholdtime  180; };
```

Context

`kernel` statement

Examples

Set the remnant hold time to 5 minutes.

```
    kernel { remnantholdtime 5:0 ; } ;
```

See Also

"Chapter 18 Kernel Interface" on page 95 of *Configuring GateD*

`kernel` on page 327

routes

Name

`routes` - limits the number of routes in the kernel's forwarding table

Syntax

`routes` *`number`*

Parameters

`number` - an integer specifying the maximum number of routes that GateD will place in the kernel's routing table. *`number`* is essentially unlimited. It must be in the range 0 to 4,294,967,295. Note that *`number`* is unsigned. Specifying -1 results in the maximum 32-bit unsigned number, which is mentioned above.

Description

On some systems, kernel memory is at a premium. With **`routes`**, a limit can be placed on the maximum number of routes GateD will install in the kernel. This discussion is concerned with three types of routes:

- interface routes: routes defined by an **`interface`** statement (includes UNIX 'ifconfig' and 'route' generated routes)
- interior routes: routes within the domain
- exterior routes: routes exterior to the domain

Normally, GateD adds, changes, or deletes routes in interface/interior/exterior order. That is, GateD queues interface routes first, followed by interior routes, followed by exterior routes, and then processes the queue from the beginning. When the route limit is reached, GateD must ensure that interface/interior/exterior route preferences are followed. This is accomplished by first deleting kernel-based routes and then turning queued changes into adds. Finally, the list of active routes in the RIB is processed in interface/internal/external order, until the route limit is reached.

Defaults

There is no limit on kernel-installed routes.

```
kernel { routes -1; } ;
```

Context

`kernel` statement

Examples

Limit the number of kernel routes to 1000.

```
kernel { routes 1000; } ;
```

See Also

"Chapter 18 Kernel Interface" on page 95 of *Configuring GateD*

`kernel` on page 327

traceoptions

Name

`traceoptions` - enable tracing for kernel-related activities

Syntax

```
traceoptions none;

traceoptions
       [ tracefile [ replace ]
       [ size tracesize [ k | m ] files tracefiles ] ] [ nostamp ]
       [ trace_global_options | trace_protocol_options |
         trace_protocol_packets ]
       [ except ( trace_global_options | trace_protocol_options |
         trace_protocol_packets ) ];
```

Parameters

`none` - Do not trace any kernel packets.

`replace` - specifies to start tracing by truncating an existing file. The default is to append to an existing file.

`size` *`tracesize`* **`[ k | m ]`** **`files`** *`tracefiles`* - Limits the maximum size of the trace file to the specified *`tracesize`* (minimum 10k). When the trace file reaches the specified size, the file is renamed to file.0, then file.1, then file.2, up to the maximum number of files. (The minimum specification is 2.)

`k` - specifies the file size in kilobytes

`m` - specifies the file size in megabytes

`files` *`tracefiles`* - specifies the maximum number of files

`nostamp` - specifies that a timestamp should not be prepended to all trace lines

`except` - The listed options that follow are not traced.

`trace_global_options` - global trace options as defined in “Chapter 4 Trace Statements” on page 15 of Configuring GateD

`trace_protocol_options` - one or more of the following options:

Although the kernel interface isn’t technically a routing protocol, in many cases it is handled as one. The following two *`trace_options`* can be entered from the command line because the code that uses them is executed before the trace file is parsed.

> **`symbols`** - Trace symbols, which are read from the kernel, by **`nlist( )`** or similar interface. The only useful way to specify this level of tracing is via the **`-t`** option on the command line, because the symbols are read from the kernel before parsing the configuration file.
>
> **`iflist`** - Trace **`iflist`**, the interface list **`scan. iflist`** is useful when entered from the command line, because the first interface list scan is performed before the configuration file is parsed.

The following *`trace_protocol_options`* may only be specified in the configuration file. They are not valid from the command line.

`remnants` - trace remnants, which specify routes read from the kernel when GateD starts

`request` - trace requests that specify to add, delete, or change routes in the kernel forwarding table

The following general option and packet *`trace_protocol_options`* apply only on systems that use the routing socket to exchange routing information with the kernel. They do not apply on systems that use the old *BSD 4.3* **`ioctl( )`** interface to the kernel.

`info` - Trace info messages, which are messages received from the routing socket, such as TCP lossage, routing lookup failure, and route resolution requests. GateD does not currently process these messages, but logs the information if requested.

`trace_protocol_packets` - **`[ detail ] [ send | receive ] ( packets | routes | redirect | interface | other )`** as defined below:

`detail` - Use detailed packet tracing.
`send` - Trace only kernel packets sent.
`receive` - Trace only kernel packets received.
`packets` - Trace all kernel packet types.
`redirect` - Trace only kernel redirect packets.
`routes` - Trace routes that are exchanged with the kernel, including add, delete, or change messages and add, delete, or change messages received from other processes.
`redirect` - Trace redirect messages, which are received from the kernel.
`interface` - Trace interface status messages that are received from the kernel. These are supported only on systems with networking code derived from *BSD 4.4*
`other` - Trace other messages that are received from the kernel, including those mentioned in the info type above. This option is currently not being used and is reserved for future use.

Description

Trace statements control tracing options. The trace options specified here are used to enable tracing of kernel-related event processing. The kernel events that can be traced include routing socket messages and kernel forwarding table updates.

Defaults

The parameters described above are disabled by default.

Context

`kernel` statement

Examples

Traces routes exchanged with the kernel.

```
kernel { traceoptions routes ; } ;
```

See Also

"Chapter 18 Kernel Interface" on page 95 of *Configuring GateD*

`kernel` on page 327