

Quagga/SMF Routing Management Design

Alan Maguire (alan dot maguire at sun dot com) and Paul Jakma (paul dot jakma at sun dot com)

quagga-iteam at sun dot com

Solaris Network Approachability, Sun Microsystems, Inc.

Revision 1.2

15 November 2005

1. Overview

This project has two related aims:

- replace SFWzebra routing protocol suite with the quagga routing protocol suite (www.quagga.net) in the Solaris Nevada SFW gate (commonly referred to as the freeware gate, where webmin etc live)
- update routing management to be SMF-aware, and convert Solaris routing daemons in.routed (1M), in.ripngd (1M), the neighbour discovery daemon in.ndpd (1M) and in.rdisc (1M) - router discovery - to SMF services

These goals are related since quagga for Solaris includes an SMF manifest for it's routing services. At present, routing management, administered via routeadm(1M) provides no way of interacting with SMF services. We aim to address that here, and convert the above named Solaris daemons to SMF services.

2. Problem Statement

Currently, the routing protocol suite under /usr/sfw (SFWzebra) is based on GNUzebra. Quagga is also based on a fork of GNUzebra, but has a more active developer community and enhanced feature/protocol support. As such, it makes sense to move to quagga. However accommodating SMF-based routing daemons would be difficult with the current routing management administration tool, routeadm (1M). As a result, we also convert routeadm to support SMF services as part of this project, and modularize routing management itself into a set of SMF services. This will require changes to Solaris Nevada, specifically to the routeadm (1M) administration tool, and inclusion of SMF routing services to manage routing, their manifests and method scripts.

3. Goals and Non-Goals

3.1 Goals

Integrate quagga "as-is" - no Solaris-specific tweaks that do not exist in the community version of quagga will be permitted.

Support automated upgrade for SFWzebra to quagga where possible.

Utilization of the SMF framework to solve routing management problems where possible.

Match syntax with other Solaris “*adm” tools where possible for ease-of-use. In particular, we focus on inetadm (1M) here, since it's goals – administration of inetd services – are similar to ours – administration of routing services.

Backwards compatibility. Legacy routing configurations which specify in.routed/in.ripngd or zebra routing daemons need to be upgraded to refer to the equivalent services – in the latter case, quagga services. All documented routeadm options will also continue to be valid.

Equivalent routing management defaults. Although the default configuration (in.routed as ipv4 routing daemon, in.ripngd as default ipv6 routing daemon, ipv4[6]-routing[forwarding] disabled) will be expressed in a slightly different manner (due to conversion of routing daemons to SMF), we intend to support the exact same default behaviour. See below for discussion of in.routed, in.ripngd and in.ndpd service conversions.

Upgrade support. At present, routeadm allows setting of routing parameters during upgrade – this is achieved using the -R option (used to specify a nonstandard root directory) in conjunction with other options. We need to ensure that the new SMF options added will work during upgrade, when the SMF repository is unavailable. We also need to ensure that legacy specification of in.routed/in.ripngd/zebra daemons are converted to invocations of the appropriate SMF services, and that zebra configuration files are migrated to quagga.

3.2 Non-goals

Delivery of SFWzebra. Since quagga provides a superset of SFWzebra functionality, and SFWzebra configuration can be safely migrated to quagga (and is done so in an automated manner), this should not present any problems to customers. We will replace SFWzebra with SFWquagga, so the SFWzebra binaries will no longer ship.

4. Concepts

Here we assume a familiarity with SMF (<http://www.sun.com/bigadmin/content/selfheal/>) and with routing management (<http://docs.sun.com/app/docs/doc/816-4554/6maoq01n1?a=view> and <http://docs.sun.com/app/docs/doc/816-4554/6maoq01n1?a=view>). First we will describe routing management as it operates presently. Then we will describe the changes that we suggest would both place routing management itself under SMF and allow routing management to manage SMF services, such as those quagga provides. Finally we will discuss how quagga would fit into this scheme.

4.1 Routing management in Solaris 10

Global routing parameters are configured via routeadm (1M). This utility allows specification of whether ip routing/forwarding is to be enabled for ipv4 and ipv6, and supports specification of a specific routing daemon (and method of stopping same) for ipv4 and ipv6. Default settings are as follows:

ipv4 forwarding:	disabled
ipv6 forwarding:	disabled
ipv4 routing:	disabled
ipv4 routing daemon:	"/usr/sbin/in.routed"
ipv4 routing daemon arguments:	""
ipv4 routing stop command:	"kill -TERM `cat /var/tmp/in.routed.pid`"
ipv6 routing daemon:	"/usr/lib/inet/in.ripngd"
ipv6 routing daemon args:	"-s"
ipv6 routing stop command:	"kill -TERM `cat /var/tmp/in.ripngd.pid`"

routeadm allows setting of forwarding/routing state via the "-e" and "-d" options, and reversion to default setting via "-r". For example,

```
# routeadm -r ipv4-routing
# routeadm -e ipv4-forwarding
# routeadm -d ipv6-routing
```

respectively revert ipv4-routing settings to their default, enable ipv4 forwarding flags and disables ipv6 routing.

These routing/forwarding parameters are stored in the project-private file `/etc/inet/routing.conf`.

Configured settings are applied to the running system using the `"-u"` flag, so the above settings would not take effect until `"routeadm -u"` is run. Applying the settings causes `routeadm` to set forwarding flags as appropriate, and execute the specified routing daemon. For `ipv6`, `routeadm` also starts `in.ndpd`.

During system boot, `routeadm` is invoked twice.

It is first called by the `net-loopback` method (run by `network/loopback:default` service). `routeadm` is called with `"-u"` and `"-F"` options, which translates to "apply current configuration, setting forwarding options only". No routing daemons are run at this time, due to it being too early - we just want to set ip forwarding flags.

A subsequent call is made to `routeadm` in the `net-init` method, (run by the `network/initial:default` service). `routeadm` is called with the `"-b"` and `"-u"` options, and either `"-e ipv4-routing"` or `"-d ipv4-routing"` is specified, depending on whether default routers are specified (the former being set if none are). These options essentially say: set default `ipv4-routing` option to enabled/disabled (`-b` specifies we update defaults, not current values) and apply changes. Note that if the current `ipv4-routing` setting is "default", this will have the effect of enabling/disabling `ipv4` routing.

The `routeadm` utility supports setting of routing parameters during upgrade. `routeadm` provides a `"-R"` option, which sets an alternate root directory (as is needed during upgrade). With this set, `routeadm` will allow setting of routing parameters, and these will be stored in `<alternate_root>/etc/inet/routing.conf`, e.g.

```
# routeadm -R <alternate_root> -e ipv4-routing
```

run during upgrade would ensure that post-upgrade, `ipv4-routing` would be enabled.

4.2 Routing Management and SMF

The problem we wish to address then is to find a way to accommodate SMF routing daemon services, such as those quagga provides, in routeadm.

We propose this could be best achieved by factoring routing management into a number of services that routeadm will interact with - one for each of the options ipv4-routing, ipv6-routing, ipv4-forwarding and ipv6-forwarding. We suggest the following FMRI for the services

ROUTEADM OPTION	FMRI	ABBREVIATION
ipv4-routing	svc:/network/routing/ipv4-routing:default	ipv4-routing
ipv6-routing	svc:/network/routing/ipv6-routing:default	ipv6-routing
ipv4-forwarding	svc:/network/routing/ipv4-forwarding:default	ipv4-forwarding
ipv6-forwarding	svc:/network/routing/ipv6-forwarding:default	ipv6-forwarding

Associating SMF services with each of these options provides a number of advantages:

- we can now enable/disable these options through SMF, e.g.

```
# svcadm disable ipv4-routing
```

- SMF routing daemons can now use dependencies on these services to ensure they only run when the appropriate service is enabled, e.g. converted in.routed would depend on ipv4-routing being enabled, while in.ripngd would depend on both ipv6-routing and ipv6-forwarding being enabled
- we can move much of the code from routeadm into the start/stop methods of the new services. This is a much more modular design – e.g. It becomes the responsibility of ipv4-forwarding's “start” method to switch on ip-forwarding flags. The “update” operation for routeadm (routeadm -u) then merely needs to enable/disable the appropriate services as required, e.g.

```
# routeadm -e ipv4-routing -u
```

would cause the ipv4-routing service to be enabled/restarted (if it is already enabled).

Note that the ipv4-routing and ipv6-routing start/stop methods would have to support starting and stopping of routing daemons that cannot be converted to SMF invocations by the routing-manager start method (see below). Again, this code would move from routeadm into the appropriate service's start/stop methods.

In addition, we need to support enabling/disabling/modification of SMF service properties via `routeadm`. We model our syntax here on `inetadm`, so the user can do the following:

```
# routeadm -e route
# routeadm -d ripng
# routeadm -m ripng routing-daemon-options="-s"
# routeadm -l route
```

```
NAME=VALUE
routing-daemon-options=""
...
```

These in turn enable `in.routed`'s service, disable `in.ripngd`'s service, modify `in.ripngd`'s commandline options, and list routing-related options for `in.routed`'s service. Note that all these operations, unlike the changing of forwarding/routing state would occur immediately, without need of an “update” operation. This is reasonable, since even if we enable the `in.routed` service, say, its dependency on `ipv4-routing` being enabled will prevent the daemon from running until `ipv4-routing` is enabled.

A key question here is: how would `routeadm` recognize routing services? We propose that a routing daemon service should provide a “routing” property group in its manifest such as:

```
<property_group name='routing' type='application' >
  <propval name='routing-daemon-args' type='astring' value='true' />
  ...
</property_group>
```

All properties to be available to `routeadm` for modification/listing should be in this property group, and the presence of the property group itself will identify the service as a routing daemon. Calling `routeadm` with “-e”, “-d”, “-m” or “-l” options, and specifying a FMRI which does not contain such a property group will trigger an error.

We also suggest enhancing the information `routeadm` provides when run with no parameters to the following:

```

# routeadm
      Configuration      Current      Current
      Option             Configuration  System State
-----
      IPv4 forwarding    disabled    disabled(disabled)
      IPv4 routing       enabled     enabled(online)
      IPv6 forwarding    disabled    disabled(disabled)
      IPv6 routing       disabled    disabled(disabled)

Non-SMF routing daemons:
      IPv4 routing daemon  ""
      IPv4 routing daemon args ""
      IPv4 routing daemon stop ""
      IPv6 routing daemon  ""
      IPv6 routing daemon args ""
      IPv6 routing daemon stop ""

ENABLED      STATE      FMRI
enabled      online     svc:/network/routing/route:default
disabled     disabled  svc:/network/routing/ripng:default
...          ...       ...

```

However, these SMF-related options present a problem. As we noted above, routeadm provides options to allow it to be run during upgrade. At present, the SMF repository is unavailable during upgrade, so to work around this, upgrade scripts add commands to `/var/svc/profile/upgrade`, which will be run on first reboot post-upgrade. Accordingly, we suggest that during upgrade, routing operations involving SMF services could be appended to be `/var/svc/profile/upgrade`, to be run on first reboot post-upgrade. To test if an operation should be added to the upgrade file, routeadm could detect if the “-R” option has been specified *and* if the SMF repository is unavailable. If so, rather than running SMF routing operations, they would be appended to `<nonstandard_root>/var/svc/profile/upgrade`.

On first reboot post-upgrade, these would be picked up and executed. This workaround could be removed in the future, when the SMF repository becomes available during upgrade.

We propose that a routing-manager service (`svc:/network/routing/routing-manager`), enabled by default, would be useful to carry out several operations:

- attempt to convert specification of in.routed/in.ripngd via ipv4(6)-routing-daemon options to specification of their SMF services
- Attempt to convert specification of SFWzebra routing daemons to specification of the quagga SMF services, and run the quagga upgrade tool to migrate config from zebra -> quagga
- carry out tests for default routers, and set ipv4-routing default accordingly (currently carried out by network-initial service (see above).
- run “routeadm -u” to apply any changes made

As noted above, during boot, routeadm is invoked twice. The first time, it is called with the undocumented “-F” flag, which specifies “apply forwarding options only”. We can remove this flag now, and replace it with dependencies – if we make the ipv4-routing and ipv6-routing services depend on the network/initial service, we ensure they do not run routing daemons too early. The second call to “routeadm -u” (and the code that determines what the ipv4-routing revert value is) in the network/initial script, can be moved to the routing-manager start method.

In addition, as mentioned, we would suggest converting in.routed, in.ripngd and in.ndpd to SMF, and suggest the following FMRIS:

DAEMON	FMRI	ABBREVIATION
in.routed	svc:/network/routing/route:default	route
in.ripngd	svc:/network/routing/ripng:default	ripng
in.ndpd	svc:/network/routing/ndp:default	ndp
in.rdisc	svc:/network/routing/rdisc:default	rdisc

The “route” and “rdisc” services would specify a “require_all” dependency on the ipv4-routing service, preventing them from running if ipv4-routing was not online. Similarly, “ripng” would depend on the ipv6-routing service and the ipv6-forwarding service. The “ndp” service would not depend on either (it is currently started by routeadm whether ipv6-routing is enabled or disabled – the only criterion is if there are ipv6 interfaces configured). It should however specify a dependency on network/initial to

prevent it from running too early, and its start method should simply return without running the daemon if no v6 interfaces are configured. Each service would specify a “routing” property group, and this could include a “routing-daemon-args” property. This would make the arguments used when running the daemon available to routeadm's “-l” and “-m” commands for viewing and modification respectively. The “route”, “ripng” and “ndp” services should have their instances enabled in their manifests. The daemons would still not run until ipv4(6)-routing was enabled of course, due to dependencies, but this would fit with current default behaviour. If the administrator enables ipv4-routing, in.routed runs. If the administrator enables ipv6-routing and ipv6-forwarding, in.ripngd runs. As long as a v6 interface is configured, in.ndpd runs.

A key aim of this design is the possibility of routing management through SMF tools. For example

```
# routeadm -e route
# routeadm -e ipv4-routing -u
```

is equivalent to

```
# svcadm enable -r route
```

(recursive enable of in.routed notes that it depends on the ipv4-routing service, so it enables this also).

In addition svcs (1M) “-x” option would be useful for indicating why routing services are offline, e.g.:

```
# svcs -x ripng
svc:/network/routing/ripng:default (in.ripngd routing daemon service)
  State: offline since Sun Sep 25 21:34:15 2005
Reason: Service svc:/network/routing/ipv6-forwarding:default is disabled
  See: http://sun.com/msg/SMF-8000-GE
  See: in.ripngd(1)
Impact: This service is not running.
```

4.3 Integration of quagga to sfw consolidation

As mentioned above, we intend replacing the SFWzebra package in the SFW consolidation with SFWquagga, consisting of the quagga routing daemons, configuration files and SMF manifest.

We do not intend to integrate the “vtysh” component of Quagga, since there are issues with it's use of libreadline, so quagga will be compiled with a switch reflecting this. The same was true of zebra.

At present, quagga defines a number of SMF instances in it's manifest (see <http://cvs.quagga.net/cgi-bin/viewcvs.cgi/quagga/solaris/quagga.xml.in?rev=1.4&contenttype=text/vnd.viewcvs-markup>). These are:

<u>DAEMON</u>	<u>FMRI</u>	<u>ABBREVIATION</u>
zebra	svc:/network/routing/quagga:zebra	quagga:zebra
ripd	svc:/network/routing/quagga:ripd	quagga:ripd
bgpd	svc:/network/routing/quagga:bgpd	quagga:bgpd
ospfd	svc:/network/routing/quagga:ospfd	quagga:ospfd
ripngd	svc:/network/routing/quagga:ripngd	quagga:ripngd
ospf6d	svc:/network/routing/quagga:ospf6d	quagga:ospf6d

All instances depend on quagga:zebra (other than itself of course). A few changes will be needed to fit with the above-suggested routing management model.

quagga:zebra will have to specify an require_any dependency on ipv4-routing/ipv6-routing. This is because zebra is required if either/both v4/v6 routing is enabled. In addition, v4 daemon instances (ripd, bgpd, ospfd) should specify a dependency on ipv4-routing, while v6 daemon instances (ripngd, ospf6d) should specify a dependency on ipv6-routing. A “routing” property group also needs to be added at the service level so the above daemons are identified to routadm as routing daemons.

As mentioned above, the routing-manager start method will automatically run on first reboot post-upgrade, and migrate SFWzebra configuration to quagga, and enable routing daemons as required. Quagga will include a script to migrate configuration files from zebra -> quagga. If a deprecated option is encountered, the configuration migration will cause an error, and the appropriate quagga services will be marked as maintenance state by routing management until the issue is resolved by the administrator.

5. Draft routeadm(1M) manpage

NAME

routeadm— IP forwarding and routing configuration

SYNOPSIS

routeadm [-p]

routeadm [-R root-dir] [-e option...|fmri...] [-d option...|fmri...] [-l fmri...] [-m fmri name=value...] [-r option...] [-s var=value]

routeadm

routeadm [-u]

DESCRIPTION

The routeadm command is used to administer system-wide configuration for IP forwarding and routing. IP forwarding is the passing of IP packets from one network to another; IP routing is the use of a routing protocol to determine routes.

routeadm is used to enable or disable each function independently, overriding any system default setting for each function. IP forwarding and routing functions are also represented as services within SMF, and can be administered via svcadm also, with FMRI `svc:/network/routing/ipv4(6)-forwarding`, `svc:/network/routing/ipv4(6)-routing`. See below for examples. The states of these services will match those configured by routeadm when the last update was run.

In addition, routeadm is used to interact with SMF-based routing daemon services, supporting enable, disable, property modification and listing for such services. Routing daemon services are identified by the presence of a “routing” application property group, which contains the properties that routeadm can list or change. If an FMRI for a service without such a property group is specified, an error is issued and the operation is not carried out. SMF routing daemon services should specify dependencies on global IP forwarding and IP routing services as necessary, in order that the global state of IP forwarding or routing dictates the running of those daemons.

If a routing daemon has not been converted to SMF, the `ipv4(6)-routing-daemon`, `ipv4(6)-routing-daemon-args`, and `ipv4(6)-routing-stop-cmd` variables can be used to specify the appropriate daemon for `ipv4(6)` routing.

OPTIONS

The following command-line options are supported:

-p

Print the configuration in parseable format.

-R root-dir

Specify an alternate root directory where routeadm applies changes. This can be useful from within JumpStart scripts, where the root directory of the system being modified is mounted elsewhere.

-e option...|fmri...

Enable the specified option or SMF routing daemon service. Enable of routing daemon services occurs immediately.

-d option...|fmri...

Disable the specified option or SMF routing daemon service. Disable of routing daemon services occurs immediately.

-l fmri...

List all properties in the “routing” application property group for SMF routing daemon service.

-m fmri name=value...

Modify property value of property name to value in “routing” application property group for SMF routing daemon service.

-r option...

Revert the specified option to the system default. The system defaults are specified in the description of each option.

-u

Apply the currently configured options to the running system. Enable or disable IP forwarding and IP routing services, and/or launch or kill routing daemons, if any are specified. It does not alter the state of the system for those settings that have been set to default. This option is meant to be used by administrators who do not want to reboot to apply their changes.

-s var=value

Specify string values for specific variables in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the program exits. The following variables can be specified:

ipv4-routing-daemon=<full_path_to_routing_daemon>

Specifies the routing daemon to be started when ipv4-routing is enabled. The routing daemon specified must be an executable binary or shell-script. Default: ""

ipv4-routing-daemon-args=<args>

Specifies the startup arguments to be passed to the ipv4-routing-daemon when ipv4-routing is enabled. Default: no arguments

ipv4-routing-stop-cmd=<command>

Specifies the command to be executed to stop the routing daemon when ipv4-routing is disabled. <command> may be an executable binary or shell-script, or a string that can be parsed by system(3C). Default: ""

ipv6-routing-daemon=<full_path_to_routing_daemon>

Specifies the routing daemon to be started when ipv6-routing is enabled. The routing daemon specified must be an executable binary or shell-script. Default: ""

ipv6-routing-daemon-args=<args>

Specifies the startup arguments to be passed to the ipv6-routing-daemon when ipv6-routing is enabled. Default: ""

ipv6-routing-stop-cmd=<command>

Specifies the command to be executed to stop the routing daemon when ipv6-routing is disabled. <command> can be an executable binary or shell-script, or a string that can be parsed by system(3C). Default: ""

Multiple -e, -d, and -r options can be specified on the command line. Changes made by -e, -d, and -r are persistent but, unless referring to an SMF routing service, are not applied to the running system unless routeadm is called later with the -u option.

Use the following options as arguments to the -e, -d, and -r options (shown above as option...).

ipv4-forwarding

Controls the global forwarding configuration for all IPv4 interfaces. The system default is disabled. If enabled, IP will forward IPv4 packets to and from interfaces when appropriate. If disabled, IP will not forward IPv4 packets to and from interfaces when appropriate. This SMF service associated with this configuration variable is svc:/network/routing/ipv4-forwarding. This service will be enabled or disabled as appropriate when routeadm is called with the -u option, or of course svcadm can be used. Services that require ipv4-forwarding to be enabled should call out a dependency on this service.

ipv4-routing

Determines whether or not an IPv4 routing daemon is run. The system default is enabled unless the `/etc/defaultrouter` file exists, in which case the default is disabled. The SMF service associated with this configuration variable is `svc:/network/routing/ipv4-routing`. This service will be enabled or disabled as appropriate when `routeadm` is called with the `-u` option, or of course `svcadm` can be used. Services that require `ipv4-routing` to be enabled should call out a dependency on this service. The SMF routing daemon service for `in.routed` is enabled by default, but depends on this service, and as a result `in.routed` will not run unless the `ipv4-routing` service is enabled.

ipv6-forwarding

Controls the global forwarding configuration for all IPv6 interfaces. The system default is disabled. If enabled, IP will forward IPv6 packets to and from interfaces when appropriate. If disabled, IP will not forward IPv6 packets to and from interfaces when appropriate. The SMF service associated with this configuration variable is `svc:/network/routing/ipv6-forwarding`. This service will be enabled or disabled as appropriate when `routeadm` is called with the `-u` option, or of course `svcadm` can be used. Services that require `ipv6-forwarding` to be enabled should call out a dependency on this service. The SMF routing daemon service for `in.ripngd` is enabled by default, but depends on this service (along with `ipv6-routing`), so as a result `in.ripngd` will not run unless both are enabled.

ipv6-routing

Determines whether or not an IPv6 routing daemon is run. The system default is disabled. The routing daemon for IPv6 is `/usr/lib/inet/in.ripngd`. If not set, the system boot scripts' current default logic determines whether or not to run `in.ripngd`. Note that even if this option is enabled, an IPv6 routing daemon will run only if `ipv6-forwarding` is enabled. The SMF service associated with this configuration variable is `svc:/network/routing/ipv6-routing`. This service will be enabled or disabled as appropriate when `routeadm` is called with the `-u` option, or of course `svcadm` can be used. Services that require `ipv4-routing` to be enabled should call out a dependency on this service. The SMF routing daemon service for `in.ripngd` also depends on this service, as a result `in.ripngd` will not run unless it is enabled.

The forwarding and routing settings are related but not mutually dependent. For example, a router will typically forward IP packets and use a routing protocol, but nothing would prevent an administrator from configuring a router that forwards packets and does not use a routing protocol. In that case, the administrator would enable forwarding, disable routing, and populate the router's routing table with static routes.

The forwarding settings are global settings. Each interface also has an `IFF_ROUTER` forwarding flag that determines whether packets can be forwarded to or from a particular interface. That flag can be independently controlled by means of `ifconfig(1M)`'s `router` option. When the global forwarding setting is changed (that is, `-u` is issued to change the value from enabled to disabled or vice-versa), all interface flags in the system are changed simultaneously to reflect the new global policy. Interfaces configured by means of DHCP automatically have their interface-specific `IFF_ROUTER` flag cleared.

When a new interface is plumbed by means of `ifconfig(1M)`, the value of the interface-specific forwarding flag is set according to the current global forwarding value. Thus, the forwarding value forms the “default” for all new interfaces.

EXIT STATUS

The following exit values are returned:

0

Successful completion.

!=0

An error occurred while obtaining or modifying the system configuration.

EXAMPLES

Example 1 Enabling IPv4 Forwarding

IPv4 forwarding is disabled by default. The following command enables IPv4 forwarding:

```
example# routeadm -e ipv4-forwarding
```

Example 2 Apply Configured Settings to the Running System

In the previous example, a system setting was changed, but will not take effect until the next reboot unless a command such as the following is used:

```
example# routeadm -u
```

An alternative to the above two steps is to simply enable the equivalent SMF service:

```
example# svcadm enable svc:/network/routing/ipv4-forwarding
```

or, using the abbreviated FMRI:

```
example# svcadm enable ipv4-forwarding
```

Example 3 Making a Setting Revert to its Default

To make the setting changed in the first example revert to its default, enter the following:

```
example# routeadm -r ipv4-forwarding example# routeadm -u
```

Example 4 Starting in.routed with the -q Flag

The following sequence of commands starts in.routed with the -q flag:

```
example# routeadm -m route routing-daemon-args="-q"
```

This sets the "routing-daemon-args" property of the "routing" application property group for service svc:/network/routing/route to "-q".

Example 5 Disabling in.routed, in.ripngd services, specifying quagga routing services instead:

```
example# routeadm -d route ripng -e quagga:ripngd quagga:ripd quagga:zebra
```

or via SMF:

```
example# svcadm disable route example# svcadm disable ripng example# svcadm enable  
quagga:ripngd example# svcadm enable quagga:ripd example# svcadm enable quagga:zebra
```

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

Interface Stability

Stable

SEE ALSO

ifconfig(1M), in.routed(1M), in.ripngd(1M), quagga(1M), gateways(4), svcadm(1M), smf(5), attributes(5)