

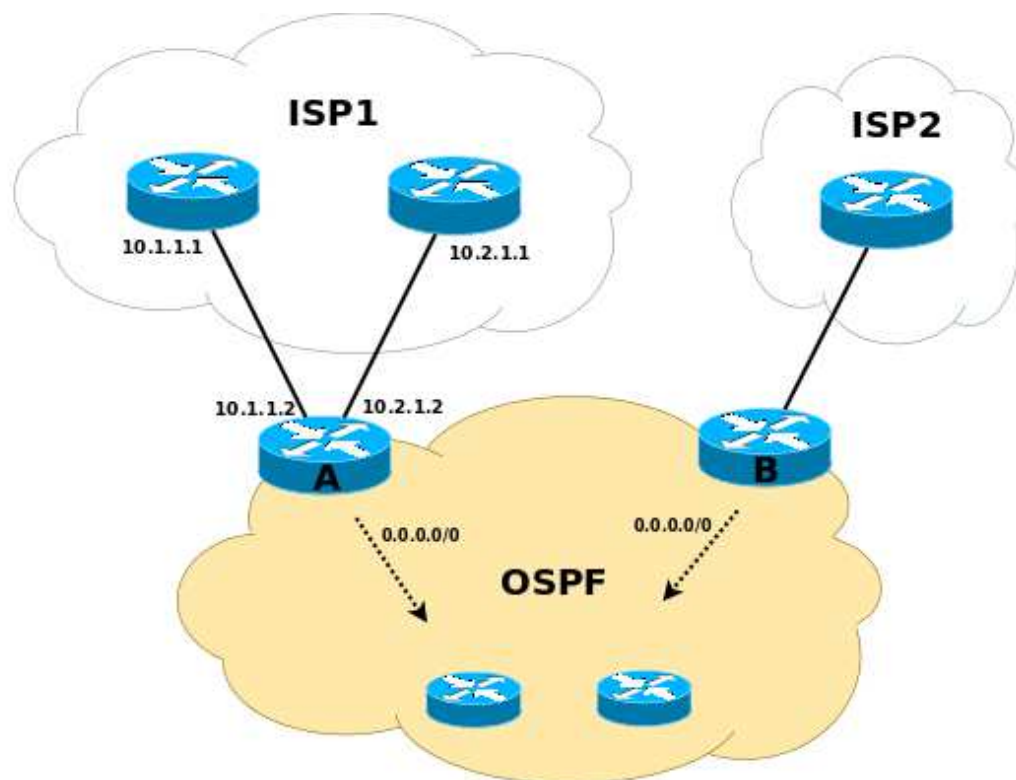
\1

Capture and remember

OSPF default route in JUNOS

Posted by waldner on 4 December 2009, 10:33 am

A common situation is where you have some internet-facing routers running BGP with some ISP, probably receiving full routes, and you want to send a default route to the internal routers, so they will use the "nearest" router (in terms of metric) to send traffic destined outside the AS. So the boundary routers run OSPF with the internal routers in area 0. The topology is something like this:



So in this example you want A and B to inject a default route into OSPF **area 0**. With Cisco, this is all configured from within the routing protocol, using some variation of the **default-information originate** statement. With Juniper this is different. Juniper has **policies**, which are configured separately from the routing protocol itself. You create a policy that does what you want, give it a name, and finally reference it from the routing protocol section (OSPF in this case).

The process is roughly as follows:

- Create a route to 0.0.0.0/0
- Create a policy that "accepts" it
- Use that policy as an *export policy* in OSPF configuration.

It's interesting to note that what you need to do with JUNOS is effectively an explicit **redistribution** of the 0.0.0.0/0 route into OSPF. With Cisco, you don't have to perform an explicit redistribution, but the **default-information originate** will generate the default as external LSA, so in practice it is equivalent to a redistribution. In all cases, the other routers see the route as external.

There are some variations in what can be done with regards to the first two steps above, so let's have a look at some possibilities.

Always generate the default

To always generate the default route, one way is to proceed as follows:

Create a route to 0.0.0.0/0

```
[edit]
root@A# set routing-options static route 0.0.0.0/0 discard

[edit]
root@A# set routing-options static route 0.0.0.0/0 no-install

[edit]
root@A# show routing-options
static {
    route 0.0.0.0/0 {
        discard;
        no-install;
    }
}
```

We don't want the router to use the default route to actually forward traffic, so the **no-install** keyword prevents the route from being installed in the *forwarding* table; it will still be visible with "show route" though, and more important for our purposes, *"even if you configure a route so it is not installed in the forwarding table, the route is still eligible to be exported from the routing table to other protocols."* (from the official documentation)

The **discard** keyword will silently drop packets without notice; if you want to send back an ICMP error message, you can use **reject** rather than **discard**. Both **discard** and **reject** are special cases of next-hops that a static route can have (the regular case is to have a real IP address as next-hop; we'll see an example of that below).

Create a policy

We now create a policy that simply accepts the static route:

```
[edit]
root@A# set policy-options policy-statement ospf-default from protocol static

[edit]
root@A# set policy-options policy-statement ospf-default from route-filter 0.0.0.0/0 exact

[edit]
root@A# set policy-options policy-statement ospf-default then accept

[edit]
root@A# show policy-options policy-statement ospf-default
from {
    protocol static;
    route-filter 0.0.0.0/0 exact;
}
then accept;
```

You can of course add some action in the "then" part, for example if you want to change the metric or metric type to influence exit point decisions by the internal routers.

Apply the policy in OSPF

So far, we have just defined some stuff, but nothing of that is actually used yet. To actually have the router start advertising the route, we need to use the policy created above as OSPF **export policy**:

```
[edit]
root@A# set protocols ospf export ospf-default
```

```
[edit]
root@A# show protocols ospf
export ospf-default;
area 0.0.0.0 {
    interface ge-0/0/0.7;
    interface ge-0/0/2.7;
    interface lo0.0 {
        passive;
    }
    interface ge-0/0/3.1 {
        passive;
    }
}
```

After you commit, router A should now be injecting the default route into the OSPF domain. Repeat the above steps on all the boundary routers that you want to use as exit points for traffic destined to the Internet (in our example, only router A and B).

Conditionally generate the default

What we have done so far *unconditionally* generates a default route; in other words, it's more or less equivalent to Cisco's **default-information originate always**. Since we have more than one boundary router, that might not always be useful, since if one of them loses its upstream BGP peering, you want internal routers to stop sending traffic to it otherwise it will be blackholed.

One way around this problem is to have the generation of the default on boundary routers be *conditional*, that is, only happen if some conditions are met. Those conditions can be, for example, the presence of a specific route or set of routes (received from upstream) in the routing table, or the availability of the BGP upstream peer.

Depending on neighbor reachability

```
[edit]
root@A# set routing-options static route 0.0.0.0/0 next-hop 10.1.1.1
```

```
[edit]
root@A# set routing-options static route 0.0.0.0/0 no-install
```

```
[edit]
root@A# show routing-options
static {
    route 0.0.0.0/0 {
        next-hop 10.1.1.1;
        no-install;
    }
}
```

The trick here is to set the next-hop to the address of a directly connected neighbor. This makes the 0.0.0.0/0 route active as long as that connected address is reachable. When it is not reachable, the route will not be active and all the policies that redistribute it will stop doing that. The net result is that the default route is injected into OSPF only if the connected neighbor is reachable.

The other parts of the configuration are as before.

Depending on the presence of specific routes

What if the directly connected neighbor is reachable, but the BGP session is down? In that case, with the previous method router A would still inject a default route into OSPF, which again may not be what we want. So we can make the generation of the default route conditional upon the existence of BGP routes in the routing table received from that neighbor.

To do this, we can use an **aggregate** or **generated** route instead of a static one.

Whereas a static route is active if the next hop is reachable, which in the case of the special **discard** and **reject** next hops means that the static route is always active, for **aggregate** and **generated** routes their being active depends on the presence of so-called **contributing** routes. A **contributing route** is, simply put, a more specific route than the aggregate or generated route (and sharing the same prefix, of course). So for example if 192.168.0.0/16 is the aggregate route, 192.168.4.0/24 can be a contributing route of the aggregate. If there is at least an active contributing route, then the aggregate or generated route is active as well.

The difference between an aggregate and a generated route is that an aggregate route, even when it's active, can only have **discard** or **reject** next hops, so it will never forward any traffic; a generated route, on the other hand, can have a real next hop, which is taken from the "preferred" contributing route (how that "preferred" contributor is determined is not important here). If a generated route is active with an actual next-hop, it behaves roughly like a static route. If a generated route is active but has a **discard** next-hop, then it acts mostly like an aggregated route. Note that a generated route cannot have a **reject** next-hop, whereas aggregated can.

For our purposes, we want the default route to not be used to forward any traffic (ie much like having a route pointing to **null0** on Cisco), so we can use either an aggregate route, or a generated route with a **discard** next-hop.

Of course, since the aggregate route is 0.0.0.0/0, then *any* route will be a contributing route and make the aggregate active, which is not what we want. The good news is that it is possible to set up a policy to decide which contributing routes are actually counted towards making the aggregate active. So for our scenario we can choose specifically the routes that we receive from the upstream BGP peer(s), and make the default route active based on the presence of those routes. Here's an example using a generated route with **discard** as next-hop:

```
[edit]
root@A# set policy-options policy-statement filter-contributors term 1 from neighbor 10.1.1.1

[edit]
root@A# set policy-options policy-statement filter-contributors term 1 from next-hop 10.1.1.1

[edit]
root@A# set policy-options policy-statement filter-contributors term 1 then accept

[edit]
root@A# set policy-options policy-statement filter-contributors term 2 from neighbor 10.2.1

[edit]
root@A# set policy-options policy-statement filter-contributors term 2 from next-hop 10.2.1

[edit]
root@A# set policy-options policy-statement filter-contributors term 2 then accept

[edit]
root@A# set policy-options policy-statement filter-contributors term 3 then reject

[edit]
root@A# show policy-options policy-statement filter-contributors
term 1 {
    from {
        neighbor 10.1.1.1;
        next-hop 10.1.1.1;
    }
    then accept;
```

```

}
term 2 {
    from {
        neighbor 10.2.1.1;
        next-hop 10.2.1.1;
    }
    then accept;
}
term 3 {
    then reject;
}

[edit]
root@A# set routing-options generate route 0.0.0.0/0 policy filter-contributors

[edit]
root@A# set routing-options generate route 0.0.0.0/0 discard

[edit]
root@A# show routing-options
generate {
    route 0.0.0.0/0 {
        policy filter-contributors;
        discard;
    }
}

[edit]
root@A# set policy-options policy-statement ospf-default from route-filter 0.0.0.0/0 exact

[edit]
root@A# set policy-options policy-statement ospf-default then accept

[edit]
root@A# show policy-options policy-statement ospf-default
from {
    route-filter 0.0.0.0/0 exact;
}
then accept;

```

Then the policy **ospf-default** is used as export policy in the OSPF configuration as before. So what we get with this method is to tie the injection of the default route into the OSPF domain to the actual reception of routes from the BGP peers. As long as either peer 10.1.1.1 or peer 10.2.1.1 are up and are sending us routes, the default route is generated. When they're not up, injection of the default route stops as none of the allowed contributors to the generated route is matched. A similar configuration should be performed on router B, using its BGP peer's IP in the filter-contributors policy.

Conclusion

The generation of a default route in JUNOS is performed differently from how it's usually done in Cisco. The methods described here are the results of my research on the matter, but any comment or correction is more than welcome because, while the Juniper documentation is excellent, there does not seem to be a lot of information on the subject (at least not as much as is available for Cisco).

Be Sociable, Share!



Tweet

2

Like

2

0

Share

Filed under [networking](#), [tips](#), [worksforme](#) Tagged [juniper](#), [junos](#), [networking](#), [ospf](#), [routing](#)
 | [Permalink](#)

7 Comments

1. *scurvy* says:

June 14, 2010 at 19:41

This is a great article! Thanks for writing it up.

2. *Lan* says:

July 20, 2010 at 09:52

This is very useful..It helps me a lot. Thanks for sharing.

3. *Hank* says:

July 24, 2010 at 08:19

Great write up. Thank you

4. *Pragnesh* says:

September 29, 2010 at 21:15

A fantastic article. Really grateful for publishing this. Thanks a lot

5. *Sohaib* says:

November 24, 2010 at 23:32

Thanks for making the effort to do this. Makes life so much easier for the rest of us dumb folks :)

6. *apm* says:

April 19, 2011 at 15:06

Interesting writing.

Good Job.

Thanks.

7. *indifo* says:

June 23, 2011 at 01:32

Thanks so much for sharing this. I have been trying to grasp this concept but it only finally clicked when I came across your article