Understanding RIB Groups

Tech Note

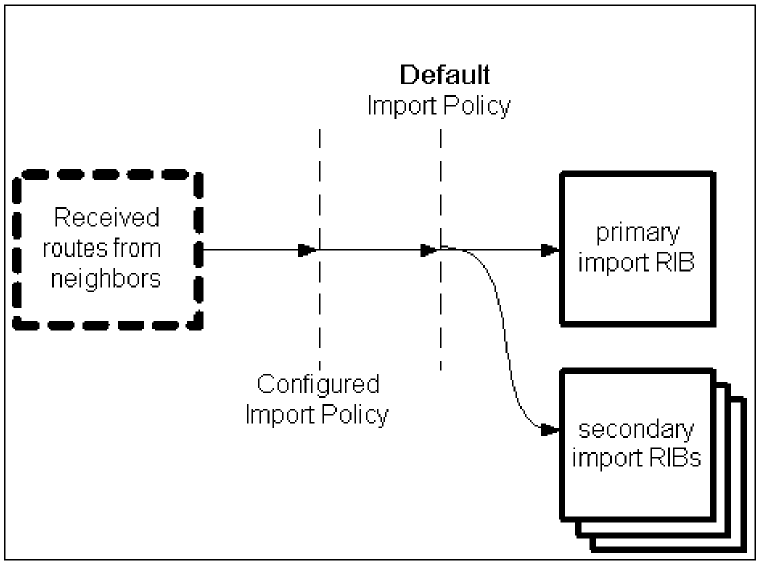
November 2016

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

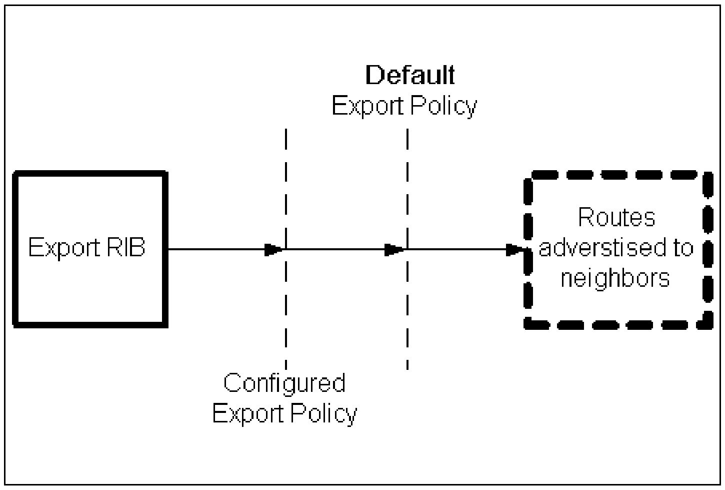
Contents

Introduction to RIB Groups

The purpose of this white paper is to explain the various ways that RIB groups can be used in Junos software. RIB groups can be used to specify the RIB(s) a routing protocol uses when it is importing and exporting routes. RIB groups are protocol specific (i.e. you cannot mix IPv4 and IPv6 RIBs in the same RIB group). Keep in mind that the words *import* and *export* are in relation to the RIB, so a protocol imports routes into a RIB and it exports routes out of a RIB. During the import process the protocol first applies any configured import policies, then it applies its default import policy. The resulting routes are stored in the import RIB of the protocol. With a RIB group you can specify multiple import RIBs. The following diagram depicts the import process:



During the export process the protocol scans the routes in the protocol's export RIB. The protocol first applies any configured export policies to the routes, and then it applies its default export policy. The resulting routes are advertised to neighbors. With a RIB group you can specify only a single export RIB. The following diagram depicts the export process:



There are two steps to configuring RIB groups. The first step is to define a named RIB group. The following configuration defines a RIB group named `example-rg`:

```
routing-options {
    rib-groups example-rg {
        import-rib [ inet.3 inet.2 ];
        export-rib vrfa.inet.0;
    }
}
```

The second step is to apply the RIB group to a protocol. The commands for applying a RIB group to each protocol differ. The following configuration applies the RIB group `example-rg` to the BGP protocol:

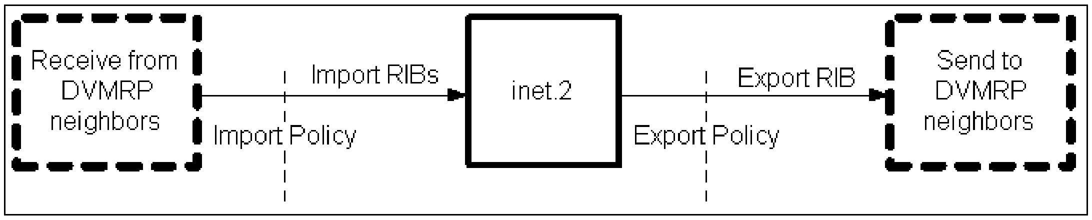
```
protocols {
    bgp {
        family inet {
            unicast {
                rib-group example-rg;
            }
        }
    }
}
```

How Each Routing Protocol Interprets RIB Groups

The introduction above explained RIB groups in generic terms. The most important thing to understand about RIB groups is that they are interpreted differently depending on the protocol they are applied to.

Distance Vector Protocols

The generic discussion fits closely with how RIB groups are interpreted by distance vector protocols. For instance, RIP uses RIB groups in an identical fashion as was described in the previous section. Also, DVMRP has a unicast routing portion that behaves almost identically to RIP. The unicast routing portion of DVMRP interprets RIB groups in the same fashion as RIP. The following diagram depicts how the DVMRP's unicast routing portion uses its RIB group configuration if inet.2 is chosen as the import and export RIB:



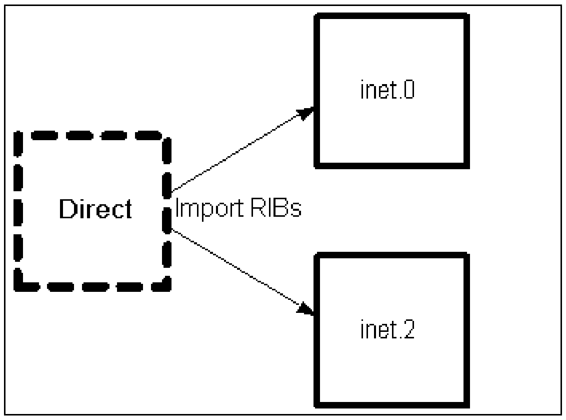
In addition to the unicast routing, DVMRP also has a multicast routing component. The router uses DVMRP's primary import RIB (the first RIB listed in the RIB group configuration) as the RPF RIB for multicast data packets received on interfaces configured for DVMRP multicast routing.

Interface Routes

An even simpler example of using RIB groups is directly connected routes. The direct *protocol* only has an import function. You can use RIB groups to choose which RIBs include the directly connected routes. Note that you must first have directly connected routes in a RIB before non-directly connected routes learned through IGPs can be installed. The following configuration places connected routes into both inet.0 and inet.2:

```
routing-options {
    interface-routes {
        rib-group if-rg;
    }
    rib-groups if-rg {
        import-rib [ inet.0 inet.2 ];
    }
}
```

The following diagram is a logical representation of what happens with this configuration:



If you configure multiple import RIBs for interface routes, then they have the unique property of only affecting rpd's routing tables. The forwarding tables in the kernel, and therefore the PFE, do not see the routes in any RIB other than the *instance*.inet.0 RIB for the instance to which the interface belongs. It works this way because rpd is permitted only very limited rights to manipulate interface routes; interface routes are really owned by dcd and the kernel. You might ask, then, what is the purpose of interface RIB groups? They're used to permit resolution of other routes to directly connected next hops.

So, in the end, you will not be able to ping the router through any VRF other than the interface's own VRF. If this is a problem, a workaround is to configure static receive routes (shown just below) for the local addresses. You will also have to apply an interface-routes import policy to filter out local routes—or don't include an interface routes RIB group—to allow the statics to work.

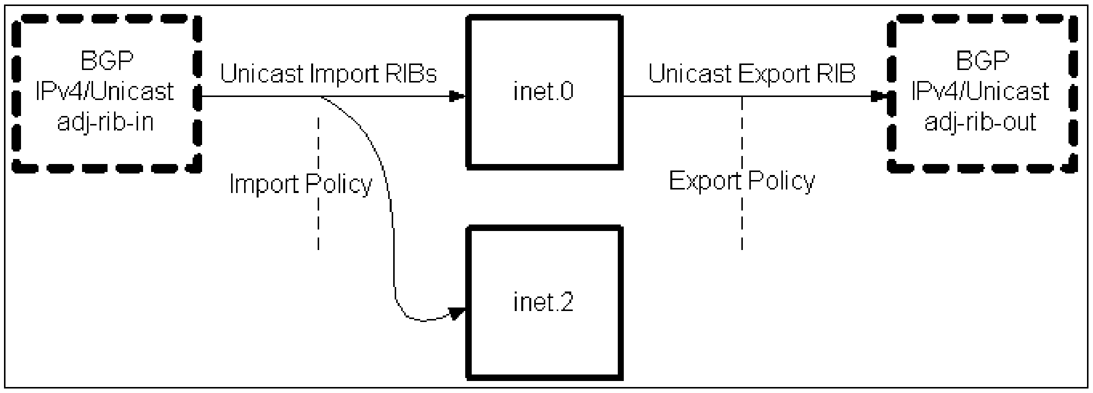
```
routing-options static route 172.16.1.1/32 receive
```

BGP

Being a path vector protocol, BGP treats RIB groups similarly to the distance vector protocols. The following configuration places routes learned from BGP into both inet.0 and inet.2, and instructs BGP to use inet.0 when exporting routes:

```
routing-options {
    rib-groups bgp-rg {
        import-rib [ inet.0 inet.2 ];
        export-rib inet.0;
    }
}
protocols {
    bgp {
        family inet {
            unicast {
                rib-group bgp-rg;
            }
        }
    }
}
```

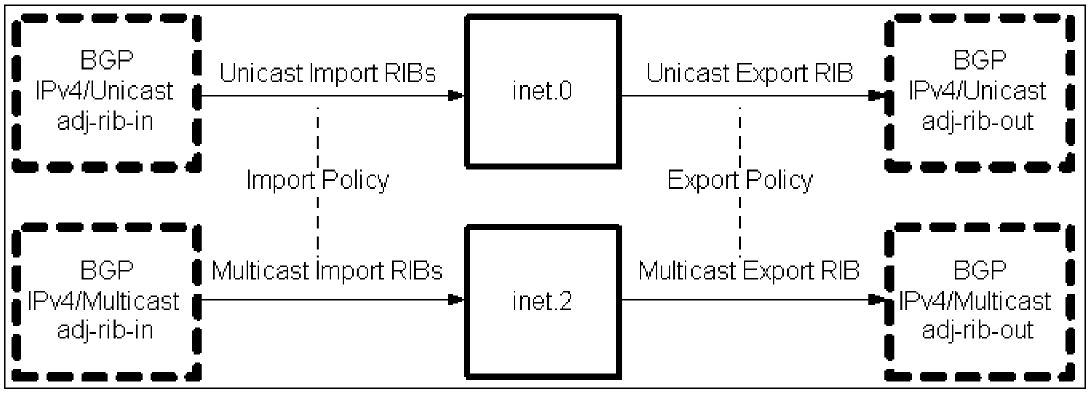
The following diagram is a logical representation of what happens with this configuration:



Additionally, RIB groups are useful when using multiprotocol BGP (MBGP). MBGP has the capability of tagging routes as belonging to different virtual topologies. The following configuration specifies MBGP routes tagged as unicast routes to be imported to and exported from inet.0, while routes tagged as multicast routes are imported to and exported from inet.2.

```
routing-options {
    rib-groups bgp-rg {
        import-rib inet.0;
        export-rib inet.0;
    }
    rib-groups mbgp-rg {
        import-rib inet.2;
        export-rib inet.2;
    }
}
protocols {
    bgp {
        family inet {
            unicast {
                rib-group bgp-rg;
            }
            multicast {
                rib-group mbgp-rg;
            }
        }
    }
}
```

The following diagram represents the results of this configuration:



Link-State Protocols

When it comes to link-state protocols things are quite different. Link-state protocols don't advertise routes like distance and path vector protocols, instead they advertise link states. The link-states are stored in protocol specific link-state databases; they are not stored in a RIB. Because of this fact, the relationship between link-state routing protocols, import/export, and RIB groups is unique.

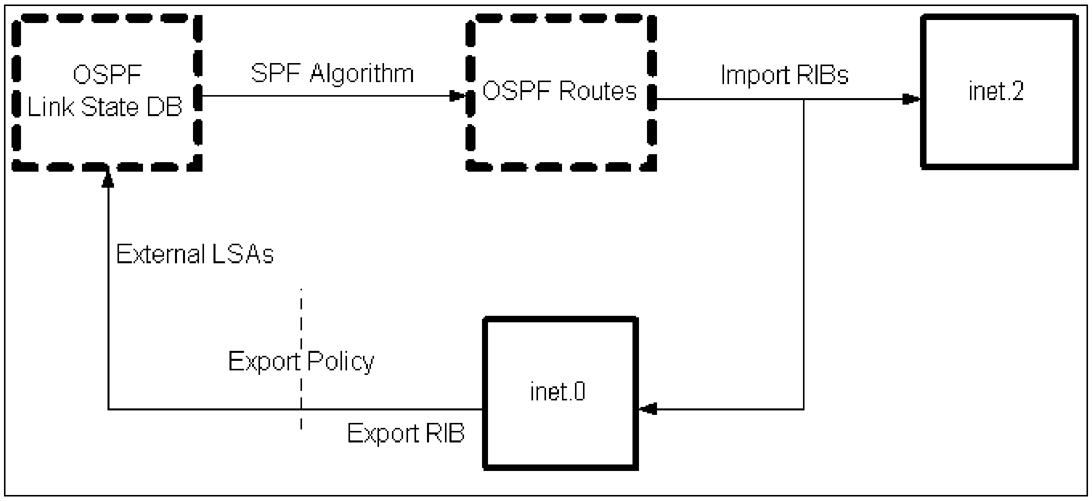
The link-state protocol runs the shortest path first (SPF) algorithm to create a protocol route table. Link-state protocols generally do not have import policies*; all of the routes created when running the SPF algorithm are accepted and placed into the protocols import RIBs. The primary use of a link-state protocol's export policy is to accept and advertise external routes (routes redistributed from other protocols). The protocol scans the export RIB and applies the export policy to all routes. The routes that are accepted are added to the link-state database as external routes. Link-state protocols use RIB groups to determine their import and export RIBs.

*OSPF offers limited import-policy functionality to block external routes (Type 5 and Type 7 LSAs) from being installed in the local device's routing table. However, the policy does not prevent those LSAs from being flooded. The use of OSPF import policies is generally discouraged.

Both IS-IS and OSPF work in similar fashion with respect to RIB groups and importing/exporting routes. The following configuration tells OSPF to import its routes into both inet.0 and inet.2, and to use inet.0 as its export RIB:

```
routing-options {
    rib-groups ospf-rg {
        import-rib [ inet.0 inet.2 ];
        export-rib inet.0;
    }
}
protocols {
    ospf {
        rib-group ospf-rg;
    }
}
```

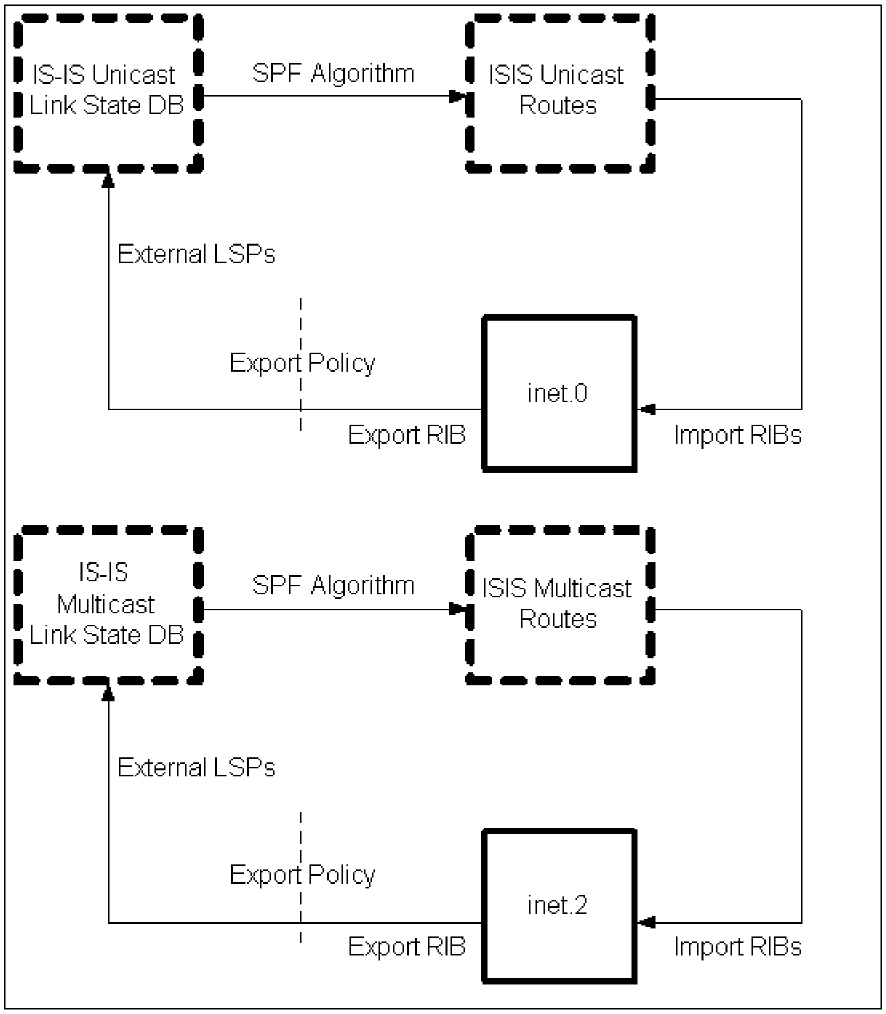
The following diagram depicts the effects of this configuration:



In addition to supporting the scenario shown above, IS-IS supports alternate (parallel) topologies for IPv4 and IPv6 unicast and multicast. The following configuration calculates an alternate IPv4 multicast topology, in addition to the normal IPv4 unicast topology, and adds the corresponding routes to inet.2.

```
protocols {
    isis {
        topologies ipv4-multicast;
    }
}
```

The following diagram shows how the IS-IS interacts with the RIBs when `topologies ipv4-multicast` is configured:



PIM and MSDP

By default, PIM and MSDP use inet.0 for their RPF checks. The Junos OS has another unicast routing table named inet.2 that is reserved for use as a dedicated multicast RPF table. PIM and MSDP use the first import RIB listed in the configuration as the multicast RPF table. Any other RIBs listed in the RIB group are ignored by PIM and MSDP. To have PIM and MSDP use inet.2 as the dedicated multicast RPF table use the following configuration:

```
routing-options {
    rib-groups {
        inet-2 {
            import-rib inet.2;
        }
    }
}
protocols {
    msdp {
        rib-group inet inet-2;
    }
    pim {
        rib-group inet inet-2;
    }
}
```

The PIM and MSDP RIB groups really have nothing to do with importing and exporting of route information. PIM imports routes into and exports routes out of inet.1, and PIM import/export policy affects this route exchange with inet.1. However, the PIM RIB group has nothing to do with this exchange. Instead the PIM import RIB only sets PIM's RPF RIB. To verify that the PIM is now using inet.2 as its RPF table, use the following command:

```
user@host> show multicast rpf
Multicast RPF table: inet.2 , 10 entries
...
```

Likewise, MSDP's RIB group acts in a similar fashion. MSDP imports and exports its control information (called source actives) to and from inet.4. MSDP import/export policies effect this exchange with inet.4. Yet the MSDP RIB group has nothing to do with this exchange. Instead the MSDP import RIB identifies the RIB that MSDP uses for its RPF-peer check.

For More Information

Follow the links below for further information and examples on RIB groups.

General

Understanding Junos OS Routing Tables

Routing Instances Overview

Example: Exporting Specific Routes from One Routing Table Into Another Routing Table

Command Reference: rib-groups

Command Reference: import-rib

Command Reference: export-rib

Distance Vector Protocols

Example: Configuring DVMRP to Announce Unicast Routes

Example: Redistributing Routes Among RIP Instances

Interface and Static Routes

Example: Importing Direct and Static Routes Into a Routing Instance

Filter-Based Forwarding Overview

Example: Configuring Filter-Based Forwarding

Command Reference: interface-routes

OSPF

Understanding OSPF Routing Instances

Example: Configuring Multiple Routing Instances of OSPF

IS-IS

IS-IS Multicast Topologies Overview

Example: Configuring IS-IS Multicast Topology

Example: Configuring IS-IS IPv4 and IPv6 Unicast Topologies

BGP

Configuring Overlapping VPNs Using Routing Table Groups

PIM/MSDP

Example: Configuring a Dedicated PIM RPF Routing Table

Example: Configuring MSDP