

# INTERNAL POLICIES AND POLICY EVALUATION FOR NG MVPNS

---

## Table of Contents

Executive Summary .....	3
Introduction .....	3
NG MVPN Overview .....	4
Default Internal Policies.....	4
Policy: __vrf-import-vpna-internal__ .....	5
Policy: __vrf-mvpn-import-cmcast-vpna-internal__.....	5
Policy: __vrf-export-vpna-internal__ .....	6
Policy: __vrf-mvpn-export-inet-vpna-internal__.....	6
Policy: __vrf-mvpn- import-cmcast-leafAD-global-internal__ .....	7
MVPN-Specific Route Targets and Policies .....	7
Policy: __vrf-mvpn-import-target-vpna-internal__.....	8
Policy: __vrf-mvpn-export-target-vpna-internal__ .....	8
Internal Policy Evaluation Order .....	9
Conclusion .....	11
Acronyms .....	11
References .....	12
About Juniper Networks.....	12

## Executive Summary

This white paper describes internal policies created by the routing system (routing protocols daemon) in order to support next-generation multicast BGP-MPLS IP VPNs (also referred as *NG MVPNs*). The protocols and procedures for creating NG MVPNs are described in draft 2547bis-mcast. The control plane is set up similar to the IP unicast BGP-MPLS VPNs where the BGP protocol is used for distributing multicast control information (also referred as *mvpn routes*). These routes are exchanged by PE routers using a new BGP address family as specified in draft 2547bis-mcast-bgp. This paper describes the policies created by the routing system in support of NG MVPNs.

## Introduction

VPN routing and forwarding tables (`routing-instance` or `vrf`) are populated based on VRF-specific import and export policies. You can configure these policies explicitly via `vrf-import` and `vrf-export` statements, or implicitly by configuring a `vrf-target` statement. When a `vrf-target` statement is configured, the system automatically creates these two internal policies.

```
__vrf-import-<routing-instance-name>-internal__
__vrf-export-<routing-instance-name>-internal__
```

Internal policies always use this underscore notation (`__`) at the beginning and end of policy names. At the end, internal policies always use the keyword `internal`. As the name implies, the first policy is applied when importing routes into a VRF table, and the latter is applied when routes are exported from the VRF table to remote provider edge (PE) routers. The purpose of the `vrf-target` statement is to simplify VRF configuration by automatically creating the required VRF import and export policies.

To support NG MVPN control plane operations, the system creates five new internal policies. They work in conjunction with the VRF import and export policies.

```
__vrf-mvpn-import-cmcast-<routing-instance-name>-internal__
__vrf-mvpn-export-target-<routing-instance-name>-internal__
__vrf-mvpn-import-target-<routing-instance-name>-internal__
__vrf-mvpn-export-inet-<routing-instance-name>-internal__
__vrf-mvpn-import-cmcast-leafAD-global-internal__
```

**Example:** This example displays seven internal policies created for a VRF table named `vpna`.

```
user@PE1> show policy
Configured policies:
__vrf-mvpn-import-cmcast-vpna-internal__
__vrf-mvpn-export-inet-vpna-internal__
__vrf-mvpn-import-target-vpna-internal__
__vrf-mvpn-export-target-vpna-internal__
__vrf-export-vpna-internal__
__vrf-import-vpna-internal__
```

Created by default when [edit routing-instances vpna protocols mvpn] is configured.

Created only if mvpn import and export route targets are configured under [edit routing-instances vpna protocols mvpn route-target] hierarchy.

Automatically created when a vrf-target statement is configured.

Since there is now more than one internal import and export policy, it is important to understand which internal policy is applied to which routing tables, what type of routes, and in which order.

## NG MVPN Overview

The extended `route-target` community configured via the `vrf-target` statement is called the *unicast route target* and is applied to unicast VPN routes. You can use the same community for `mvpn` routes as the multicast route target. This is the default behavior if no special multicast `route-target` community is configured. You can also configure a multicast `route-target` community. The routing system handles the creation of appropriate internal policies based on unicast and multicast route targets configured.

When a `routing-instance` is configured for VPN service, a PE router creates two tables. The first one is the `bgp.l3vpn.0` table, which holds all the VPN unicast routes learned from remote PE routers. The second one is the VRF-specific `<routing-instance-name>.inet.0` table, which holds local unicast VPN routes as well as VPN routes imported from the `bgp.l3vpn.0` table. With the implementation of NG MVPNs, two new tables are created: `bgp.mvpn.0` and `<routing-instance-name>.mvpn.0`. The `bgp.mvpn.0` table holds all the `mvpn` routes received from remote PE routers. The `<routing-instance-name>.mvpn.0` table holds local `mvpn` routes and those `mvpn` routes that are imported from the `bgp.mvpn.0` table.

Within the `<routing-instance-name>.mvpn.0` table, there could be up to seven types of `mvpn` routes. However, `mvpn` routes can be divided into two categories based on their role in the control plane: non-C-multicast and C-multicast `mvpn` routes.

Non-C-multicast `mvpn` routes contain autodiscovery routes, as well as those `mvpn` routes that are used in more complex MVPN operations, such as setting up `selective` data tunnels. Autodiscovery `mvpn` routes are mainly used for communicating MVPN membership information between PE routers. MVPN route types are described in drafts 2547bis-mcast and 2547bis-mcast-bgp. Types 1, 2, 3, 4, and 5 are considered as the non-C-multicast `mvpn` routes.

C-multicast `mvpn` routes are used for propagating VPN customer multicast routing information, that is, PIM `join` messages received from local receivers. A PE router translates local C-PIM `join` messages into BGP C-multicast `mvpn` routes and advertises them to remote PE routers towards the sources. There are two types of C-multicast `mvpn` routes. If the PE-CE multicast protocol is PIM-SM in ASM mode, then the PE router creates a Type 6 (shared tree `join`) C-multicast `mvpn` route. If the PE-CE multicast protocol is PIM-SM in SSM mode, then the PE router creates Type 7 (source tree `join`) C-multicast `mvpn` route. Type 6 and 7 are considered as C-multicast `mvpn` routes.

Policies applied to C-multicast `mvpn` routes differ from the policies applied to non-C-multicast `mvpn` routes. The difference comes from the fact that C-multicast routes represent MVPN receivers and they only need to be propagated to those PE routers where the sources are located. On the other hand, non-C-multicast `mvpn` routes are used for more generic control plane operations and are distributed, in general, across all the PE routers.

## Default Internal Policies

When you configure a PE router to support NG MVPNs, five internal policies are created.

**Example:** A typical MVPN `routing-instance` configuration is as follows.

```
user@PE1# show routing-instances
vpna {
  instance-type vrf;
  interface so-0/2/1.0;
  interface lo0.1;
  route-distinguisher 1:100;
  provider-tunnel {
    pim-asm {
      group-address 239.1.1.2;
    }
  }
  vrf-target target:100:200;
  protocols {
    ospf {
      export inject_bgp_routes;
      area 0.0.0.0 {
```

```

        interface lo0.1;
        interface so-0/2/1.0;
    }
}
pim {
    interface all;
}
mvpn;
}
}

```

In this example, the `vrf-target target:100:200` statement results in the creation of the following two policies.

```

__vrf-export-vpna-internal__
__vrf-import-vpna-internal__

```

Similarly, the `protocols mvpn` statement results in the creation of the following three policies.

```

__vrf-mvpn-import-cmcast-vpna-internal__
__vrf-mvpn-export-inet-vpna-internal__
__vrf-mvpn-import-cmcast-leafAD-global-internal__

```

### Policy: `__vrf-import-vpna-internal__`

This policy is applied to `bgp.13vpn.0` and `bgp.mvpn.0` tables. It compares the `route-target` community of all the VPN routes in the `bgp.13vpn.0` table against the unicast `route-target` community configured for `vpna` (`target:100:200`). Similarly, it compares the `route-target` community of all non-C-multicast `mvpn` routes in the `bgp.mvpn.0` table (Type 1 through Type 5) against the unicast `route-target` community configured for `vpna` (`target:100:200`). If there is a match, VPN routes from the `bgp.13vpn.0` table are installed in the `vpna.inet.0` table while `mvpn` routes from the `bgp.mvpn.0` table are installed in the `vpna.mvpn.0` table.

#### Example:

```

user@PE1> show policy __vrf-import-vpna-internal__
Policy __vrf-import-vpna-internal__:
    Term unnamed:
        from community __vrf-community-vpna-common-internal__ [target:100:200]
        then accept
    Term unnamed:
        then reject

```

The values in this example are as follows.

```

__vrf-import-vpna-internal__ = internal policy name
__vrf-community-vpna-common-internal__ = name of unicast route-target community
target:100:200 = value assigned to this community

```

### Policy: `__vrf-mvpn-import-cmcast-vpna-internal__`

This policy is applied to the `bgp.mvpn.0` table. It evaluates the `route-target` community of the C-multicast `mvpn` routes against a special `route-target` community. The value of this special community is entirely different than either the unicast or the multicast route targets configured. The routing system keeps track of the value of the import `route-target` that needs to be applied to C-multicast `mvpn` routes. If there is a match, the C-multicast `mvpn` routes are installed in the `vpna.mvpn.0` table.

.....

Example: Only those C-multicast `mvpn` routes from the `bgp.mvpn.0` table with a `route-target` community of `target:10.255.170.104:3` are imported into the `vpna.mvpn.0` table.

```
user@PE1>show policy __vrf-mvpn-import-cmcast-vpna-internal__
Policy __vrf-mvpn-import-cmcast-vpna-internal__:
  Term unnamed:
    from community __vrf-mvpn-community-rt_import-target-vpna-internal__
  [target:10.255.170.104:3]
    then accept
  Term unnamed:
    then reject
```

The values in this example are as follows.

`__vrf-mvpn-import-cmcast-vpna-internal__` = internal policy name  
`__vrf-mvpn-community-rt_import-target-vpna-internal__` = name of import `route-target` community  
`target:10.255.170.104:3` = value assigned to this community

.....

### Policy: `__vrf-export-vpna-internal__`

This policy is applied to `vpna.inet.0` and `vpna.mvpn.0` tables. Applied to `vpna.inet.0` table, it attaches the unicast `route-target` community to all unicast `vpn` routes advertised to remote PE routers. Applied to the `vpna.mvpn.0` table, it attaches the unicast `route-target` community to all non-C-multicast `mvpn` routes. C-multicast `mvpn` routes in the `vpna.mvpn.0` table are not affected by this policy.

.....

Example:

```
user@PE1>show policy __vrf-export-vpna-internal__
Policy __vrf-export-vpna-internal__:
  Term unnamed:
    then community + __vrf-community-vpna-common-internal__ [target:100:200]    accept
```

The values in this example are as follows.

`__vrf-export-vpna-internal__` = internal policy name  
`__vrf-community-vpna-common-internal__` = name of export `route-target` community  
`target:100:200` = value assigned to this community

.....

### Policy: `__vrf-mvpn-export-inet-vpna-internal__`

This policy is applied to all routes in the `vpna.inet.0` table. It attaches two new extended communities called *source AS* (`src-as`) and *VFR route import* (`rt-import`) to all the unicast VPN routes advertised to remote PE routers. These communities are also created internally by the routing system.

.....

Example: This internal policy adds `src-as` and `rt-import` communities.

```
user@PE1>show policy __vrf-mvpn-export-inet-vpna-internal__
Policy __vrf-mvpn-export-inet-vpna-internal__:
  Term unnamed:
    then community + __vrf-mvpn-community-rt_import-vpna-internal__ [rt-
import:10.255.170.104:3 ] community + __vrf-mvpn-community-src_as-vpna-internal__ [src-
as:65000:0] accept
```

The values in this example are as follows.

`__vrf-mvpn-export-inet-vpna-internal__` = internal policy name  
`__vrf-mvpn-community-rt_import-vpna-internal__` = name of `rt-import` community  
`rt-import:10.255.170.104:3` = value assigned to this `rt-import` community  
`__vrf-mvpn-community-src_as-vpna-internal__` = name of `src-as` community  
`src-as:65000:0` = value assigned to this `src-as` community

.....

**Policy: `__vrf-mvpn- import-cmcast-leafAD-global-internal__`**

This policy is applied to Type 4 routes in the `bgp.mvpn.0` table. It evaluates the `route-target` community of the Type 4 `mvpn` routes against a special `route-target` community that is different than unicast, multicast, and C-multicast communities. It is used for setting up selective tunnels. Note that the routing system applies this policy in addition to the unicast and multicast `vrf-import` policies. Type 4 routes must pass all import policies, including `__vrf-mvpn- import-cmcast-leafAD-global-internal`.

**MVPN-Specific Route Targets and Policies**

In JUNOS Software, you can configure multicast `route-target` communities that are different than the unicast import and export `route-target` communities. Multicast import and export route targets are applied to non-C-multicast `mvpn` routes only. These route targets are configured under the `[edit routing-instances <routing-instance-name> protocols mvpn route-target]` hierarchy.

- To attach an MVPN-specific `route-target` community to non-C-multicast routes, configure `export-target` statement under this hierarchy.
- To import non-C-multicast `mvpn` routes carrying an MVPN-specific `route-target` community configure `import-target` statement under this hierarchy.

An MVPN-specific `route-target` configuration is intended for noncongruent network topologies, meaning unicast and multicast traffic following different data paths. It is recommended to avoid configuring MVPN-specific route targets if the unicast and multicast topologies are congruent. This is due to the complexity and the interaction of the internal policies involved.

**Example:** A typical MVPN `routing-instance` configuration with an MVPN-specific `route-target` is as follows.

```
user@PE1# show routing-instances
vpna {
  instance-type vrf;
  interface so-0/2/1.0;
  interface lo0.1;
  route-distinguisher 1:100;
  provider-tunnel {
    pim-asm {
      group-address 239.1.1.2;
    }
  }
  vrf-target target:100:200;
  protocols {
    ospf {
      export inject_bgp_routes;
      area 0.0.0.0 {
        interface lo0.1;
        interface so-0/2/1.0;
      }
    }
    pim {
      interface all {
        mode sparse;
        priority 100;
        version 2;
      }
    }
    mvpn {
      route-target {
        import-target {
          target target:1:2;
        }
      }
    }
  }
}
```

```

    }
    export-target {
        target target:1:2;
    }
}
    }
}
}

```

This configuration results in the creation of the following internal policies.

```

__vrf-mvpn-import-target-vpna-internal__
__vrf-mvpn-export-target-vpna-internal__

```

### Policy: **\_\_vrf-mvpn-import-target-vpna-internal\_\_**

This policy is applied to the `bgp.mvpn.0` table. It compares the `route-target` community of non-C-multicast `mvpn` routes against the multicast `route-target` community configured via this policy. If there is a match, the `mvpn` routes are imported into the `vpna.mvpn.0` table.

#### Example:

```

user@PE1>show policy __vrf-mvpn-import-target-vpna-internal__
Policy __vrf-mvpn-import-target-vpna-internal__:
  Term unnamed:
    from community __vrf-mvpn-community-import-vpna-internal__ [target:1:2]
    then accept
  Term unnamed:
    then reject

```

The values in this example are as follows.

```

__vrf-mvpn-import-target-vpna-internal__ = internal policy name
__vrf-mvpn-community-import-vpna-internal__ = name of import multicast route-target community
target:1:2 = value assigned to this community

```

### Policy: **\_\_vrf-mvpn-export-target-vpna-internal\_\_**

This policy is applied to the `vpna.mvpn.0` table. It attaches the multicast `route-target` community configured in this policy to all non-C-multicast `mvpn` routes.

#### Example:

```

user@PE1>show policy __vrf-mvpn-export-target-vpna-internal__
Policy __vrf-mvpn-export-target-vpna-internal__:
  Term unnamed:
    then community + __vrf-mvpn-community-export-vpna-internal__ [target:1:2] accept

```

The values in this example are as follows.

```

__vrf-mvpn-export-target-vpna-internal__ = internal policy name
__vrf-mvpn-community-export-vpna-internal__ = name of export multicast route-target community
target:1:2 = value assigned to this community.

```

## Internal Policy Evaluation Order

In JUNOS Software, you can configure non-C-multicast `mvpn` routes to be evaluated against both the unicast and multicast `route-target` communities. Similarly, you can configure non-C-multicast `mvpn` routes to be advertised with both route targets. This is done by configuring the unicast statement under `[edit routing-instances <routing-instance-name> protocols mvpn route-target import-target target]` and `[<routing-instance-name> protocols mvpn route-target export-target target]` hierarchies.

**Example:** A typical MVPN `routing-instance` configuration with MVPN-specific route targets that also allows unicast route targets to be included in non-C-multicast `mvpn` route import and export operations is as follows.

```
user@PE1# show routing-instances
vpna {
    instance-type vrf;
    interface so-0/2/1.0;
    interface lo0.1;
    route-distinguisher 1:100;
    provider-tunnel {
        pim-asm {
            group-address 239.1.1.2;
        }
    }
    vrf-target target:100:200;
    protocols {
        ospf {
            export inject_bgp_routes;
            area 0.0.0.0 {
                interface lo0.1;
                interface so-0/2/1.0;
            }
        }
        pim {
            interface all {
                mode sparse;
                priority 100;
                version 2;
            }
        }
        mvpn {
route-target {
            import-target {
                unicast;
                target target:1:2;
            }
            export-target {
                unicast;
                target target:1:2;
            }
        }
    }
}
```

On the import side, the configuration of the `unicast` statement results in both the internal and user-configured unicast and multicast import policies to be evaluated using an OR operation. In other words, assuming only the internal policies exist in the system, non-C-multicast `mvpn` routes are imported into the `<routing-instance-name>.mvpn.0` table if they pass either of the following two policies.

```
__vrf-import-<routing-instance-name>-internal__
OR
__vrf-mvpn-import-<routing-instance-name>-internal__
```

On the export side, the unicast statement results in both the internal and user configured unicast and multicast export policies to be applied using an AND operation. In other words, assuming there are only internal policies in the system, the non-C-multicast mvpn routes are exported based on the following two policies.

```
__vrf-export-<routing-instance-name>-internal__
AND
__vrf-mvpn-export-<routing-instance-name>-internal__
```

Note that these policies are evaluated as a whole, not just the policy terms that have the route-target community. Therefore, care must be taken when configuring unicast keyword because the results of the OR/AND operation of the policies depend on the entire content of the policies. If a user-configured VPN policy has a term that has a reject action, it might have an adverse affect on the importing or exporting of non-C-multicast mvpn routes.

**Example:** Multicast import policy with unicast keyword.

```
user@PE1# show routing-instances vpna vrf-target
target:100:200;

[edit]
user@PE1# show routing-instances vpna protocols mvpn
route-target {
  import-target {
    unicast;
    target target:1:2;
  }
}

[edit]
```

In this example, the following two internal import policies are created and applied to non-C-multicast mvpn routes in the `bgp.mvpn.0` table. Those routes that have either the unicast (`target:100:200`) or the multicast (`target:1:2`) route-target community are installed in the `vpna.mvpn.0` table.

```
user@PE1> show policy __vrf-import-vpna-internal__
Policy __vrf-import-vpna-internal__:
  Term unnamed:
    from community __vrf-community-vpna-common-internal__ [target:100:200 ]
    then accept
  Term unnamed:
    then reject

user@PE1> show policy __vrf-mvpn-import-target-vpna-internal__
Policy __vrf-mvpn-import-target-vpna-internal__:
  Term unnamed:
    from community __vrf-mvpn-community-import-vpna-internal__ [target:1:2 ]
    then accept
  Term unnamed:
    then reject
```

**Example:** Multicast export policy with unicast keyword.

```
user@PE1# show routing-instances vpna vrf-target
target:100:200;
```

```
[edit]
user@PE1# show routing-instances vpna protocols mvpn
route-target {
  export-target {
    unicast;
    target target:1:2;
  }
}

[edit]
```

In this example, the following two internal export policies are created and applied to non-C-multicast `mvpn` routes in the `vpna.mvpn.0` table. These routes are exported with both `route-target` communities.

```
user@PE1> show policy __vrf-export-vpna-internal__
Policy __vrf-export-vpna-internal__:
  Term unnamed:
    then community + __vrf-community-vpna-common-internal__ [target:100:200 ] accept

user@PE1> show policy __vrf-mvpn-export-target-vpna-internal__
Policy __vrf-mvpn-export-target-vpna-internal__:
  Term unnamed:
    then community + __vrf-mvpn-community-export-vpna-internal__ [target:1:2 ] accept
```

## Conclusion

Special attention must be paid to internal policies when NG MVPN service is configured on a PE router. The default values are set to ensure MVPN routing works with minimal user configuration. However, it is quite likely that there would be several VRF unicast import and export policies that are already configured on a PE router before NG MVPN service is turned on. As this new service is introduced, knowing which internal policies come into play and how they interact with existing unicast policies is essential for quick resolution of problems.

## Acronyms

AS	autonomous system
CE	customer edge
C-PIM	customer PIM
MVPN	multicast VPN
MCAST	multicast
NG MVPN	next-generation multicast VPN
NLRI	network layer reachability information
PE	provider edge
PIM	Protocol Independent Multicast
PIM-SM	PIM sparse mode
SSM	source-specific multicast
VPN	virtual private network
VRF	VPN routing and forwarding table

## References

BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs.  
<http://ietfreport.isoc.org/idref/draft-ietf-l3vpn-2547bis-mcast-bgp/>  
Multicast in MPLS/BGP IP VPNs.  
<http://tools.ietf.org/html/draft-ietf-l3vpn-2547bis-mcast-08>

## About Juniper Networks

Juniper Networks, Inc. is the leader in high-performance networking. Juniper offers a high-performance network infrastructure that creates a responsive and trusted environment for accelerating the deployment of services and applications over a single network. This fuels high-performance businesses. Additional information can be found at [www.juniper.net](http://www.juniper.net).

---

### Corporate and Sales Headquarters

Juniper Networks, Inc.  
1194 North Mathilda Avenue  
Sunnyvale, CA 94089 USA  
Phone: 888.JUNIPER  
(888.586.4737)  
or 408.745.2000  
Fax: 408.745.2100

### APAC Headquarters

Juniper Networks (Hong Kong)  
26/F, Cityplaza One  
1111 King's Road  
Taikoo Shing, Hong Kong  
Phone: 852.2332.3636  
Fax: 852.2574.7803

### EMEA Headquarters

Juniper Networks Ireland  
Airside Business Park  
Swords, County Dublin,  
Ireland  
Phone: 35.31.8903.600  
Fax: 35.31.8903.601

Copyright 2009 Juniper Networks, Inc. All rights reserved. Juniper Networks, the Juniper Networks logo, JUNOS, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. JUNOSe is a trademark of Juniper Networks, Inc. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

To purchase Juniper Networks solutions, please contact your Juniper Networks representative at 1-866-298-6428 or authorized reseller.

