L3VPN Route-target and route-distinguisher Part I:

When configuring an L3VPN, you need to include both a **route-distinguisher** and a **route-target**. Due to the similar format of these two values, it is hard to understand at first why they are in fact very different, and why both are required.

This purpose of this article is to explain what these two parameters are for, and how they are used together to allow multiple L3VPNs, with potentially overlapping IP address ranges, to co-exist.

Part II of this article will show example configurations of the use of the **route-distinguisher** and **route-target**.

The challenge:

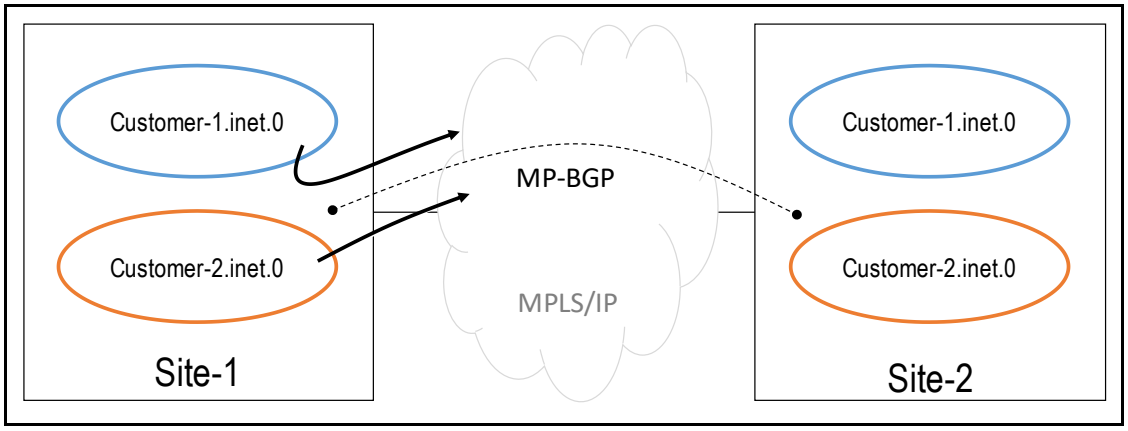
In JUNOS, the first step to create an L3VPN is to configure a routing-instance of type vrf, with a given **`instance-name`**. As a result of configuring this routing-instance (once all the configuration steps have been completed, and the configuration committed), the router creates a separate routing table named **`instance-name.inet.0`**.

For example, if you create an instance called **`SLI-VPN`**, the router creates a routing table named **`SLI-VPN.inet.0`**

You can configure more than one routing-instance on the same router. For example, you can configure **`routing-instance customer-1`**, and **`routing-instance customer-2`**. The router will create two new routing tables: **`customer1.inet.0`** and **`customer-2.inet.0`**. Each routing table will contain independent sets of routing entries.

Now, part of the idea of creating an L3VPN is to be able to share routes across multiple sites, while keeping independent routing for the different instances. In other words, you want to be able to create the same routing-instances on two different routers, and exchange routes between them.

As an example, consider the following scenario where you have created the two routing instances described above, on two different sites (Site-1 and Site-2), which are connected across an MPLS/IP network.



In order to share routes between **`customer-1.inet.0`** on Site-1 and **`customer-1.inet.0`** on Site-2, and between **`customer-2.inet.0`** on Site-1 and **`customer-2.inet.0`** on Site-2, you configure BGP, in fact, MP-BGP (Multiprotocol-BGP) which will be explained later.

Now consider what would happen when routes advertised by Site-1 are received by Site-2. How does the receiving router know which routes belongs to **`customer-1.inet.0`** and which routes belong to **`customer-2.inet.0`** ?

The answer: by looking at the routes **route-target,** and comparing it with the locally configured **route-target** policies.

Route Target:

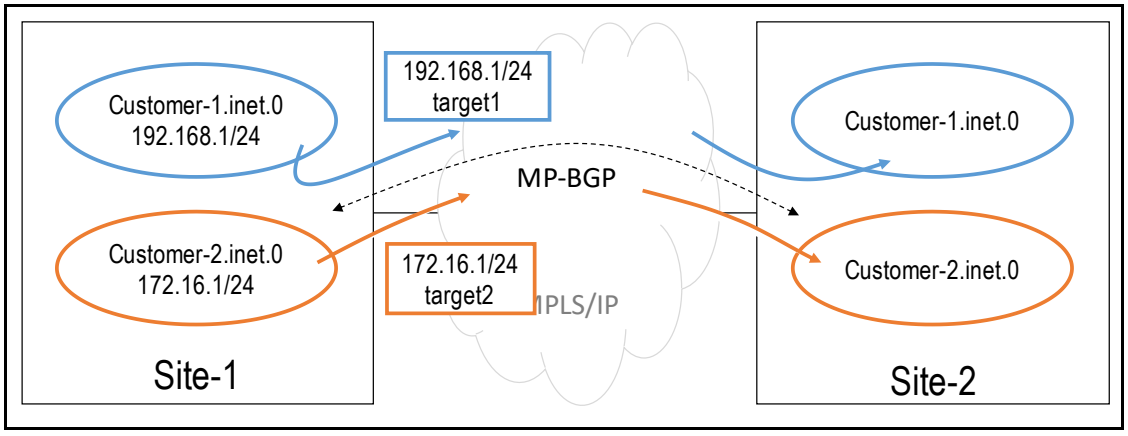
A **route-target** is a community attribute that is added to a route, to identify which routing-table(s), the route should be installed on. You can think of a **route-target** as a value that is attached to a route, and associates it with a given VPN or routing-instance; sometimes with more than one.

route-targets values can be used to:

- Install routes in a specific L3VPN routing table
- Install routes in 2 or more L3VPN routing tables, for example, for customers who are partners and required connectivity.
- Build hub-and-spoke L3VPN topologies.

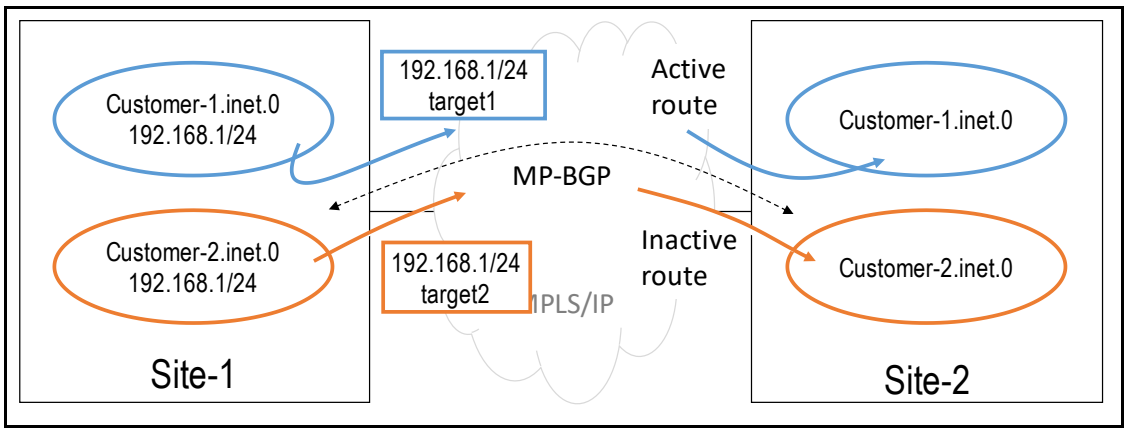
This document does not cover all these cases, but for the purposes on understanding the use of the route-target vs. route-distinguisher, it is important to understand that the **route-target** is a label that can be used to install routes into one or more routing-tables, as needed.

Continuing with our previous example, now the routes from customer 1 and customer 2 could be placed in the correct routing tables by looking at the route targets.



NOTE: Keep in mind, that this is just trying to show the **route-target** function. You won't be able to configure an L3VPN without adding the **route-distinguisher**.

Now consider what would happen if the customers were using overlapping address ranges, typically from the private address space, as shown in the diagram below. When the receiving router on Site-2 receives these two routes for the same prefix, it considers them to be the same route, so only one becomes active.



At this point, you are probably thinking: "wait, but these two routes are not the same, they have different route-targets". They do have different **route-targets**, but for the BGP decision process they are the same. The **route-target** is not part of the process, as described next.

BGP Communities and the decision process.

Details on the JUNOS BGP decision process can be found in the following document.

https://www.juniper.net/documentation/en_US/JUNOS13.3/topics/reference/general/routing-ptotocols-address-representation.html

When multiple BGP routes for the same prefix are received, attributes such as local-preference, AS-path, origin, MED, and IGP next-hop, are compared in a certain order until a route is selected. Communities are not one of the attributes compared.

1) Communities are simply tags attached to routes sent to BGP peers. Routes belonging to multiple customers, and/or multiple sites, or peers could be tagged with the same community.

2) Communities are a powerful tool to mark BGP routes and take certain actions based on that marking, but by themselves they do nothing. They are again not used in the BGP decision process.

   You could apply a policy that reads the value of the community, and modifies some other BGP attribute (local-preference for example), thus affecting the decisions made by the router, but by themselves communities will not make a route be preferred over another route. BGP simply does not look at them when going through the selection process.

3) Communities cannot be used to differentiate routes for the same prefix:

When two BGP routes with the same prefix value arrive from the same peer, and all attributes included in the decision process are the same, the router considers them to be the same route. The community tags on the routes make no difference.

Using our previous example again, if the two customers were using **192.168.1/24** (**prefix1**), only one of them would have his route active in the routing table at the remote site.

In other words, if Site-1 advertised the following two routes to Site-2:

route1 (from customer1) = prefix1 w/ community1 (target1)
route2 (from customer2) = prefix1 w/ community2 (target2)

Only one of the two routes would become active.

Now, if **route1** and **route2** are modified to somehow make them look different, while still advertising **prefix1**, the receiving router would treat them as different routes, and would accept and keep both routes active, thus allowing the two customers to advertise the same prefix without any conflicts.

This is achieved by means of the **route-distinguisher**, as explained in the next section.

VPN-IPv4 Address Family – Route Distinguisher

MP-BGP is an extended BGP that can carry multiple address families or NLRI (Network Layer Reachability Information) types such as IPV4 (inet) and IPv6 (inet6), on the same BGP session. RFC4364 introduced a new BGP address family called VPN-IPv4.

A VPN-IPv4 address is a 12-byte quantity, which consists of an 8-byte Route Distinguisher (RD), and a 4-byte IPv4 address. Basically, an IPv4 address can be converted into an VPN-IPv4 address by appending a RD.

Example: IPv4 = **192.168.1.1** => VPN-IPV4 = **1:1:192.168.1.1** (with RD = **1:1**)

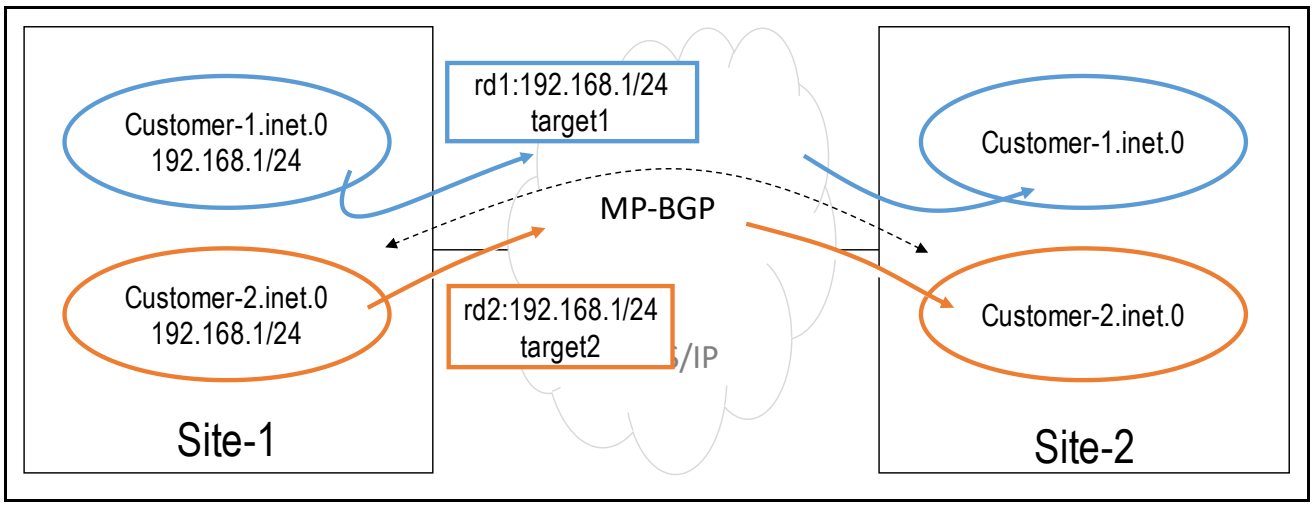
Applying this idea to our example, if **customer1** and **customer2**, the router in Site1 can make the routes for **prefix1** unique for each customer, by converting the prefix into unique VPN-IPv4 prefixes, before the routes are sent to Site-2.

In other words, by adding unique Route Distinguishers to **prefix1**, the routes from **customers1** and **customer2,** become unique and can now be treated by the receiving router as different routes:

route1 (customer1) = RD1:prefix1 w/ community1
route2 (customer2) = RD2:prefix1 w/ community2

RD1:prefix1 and **RD2:prefix1** contain the same IPv4 prefix (**prefix1**), but are unique VPN-IPv4 prefixes, so no selection process between the two is performed.

At the receiving router in Site-2, both routes are now active. The router can now look at the community value on each route to decide which routing table(s) they should be installed on.



Notes:

- Since VPN-IPv4 addresses and IPv4 addresses are different address families, BGP never treats them as comparable addresses.
- VPN-IPV4 prefixes in JUNOS are installed in a separate common routing table called bgp.l3vpn.0. The routes are imported into the appropriate routing table, based on their route-target
- The advertisement of IPV4 NLRI is configured by adding family inet; VPN-IPv4 NLRI is configured by adding family inet-vpn.
- Family inet is enabled by default when you configure BGP, but you need to explicitly enable family inet-vpn. Keep in mind that when you explicitly enable an address family, the default family no longer applies. Thus, if you enable family inet-vpn, you must also explicitly enable family inet, if you still want to carry IPv4 routes.

Route Target and Route Distinguisher together

To summarize the previous concepts:

- The **route-distinguisher** is used to differentiate equal prefixes that belong to different customers or VPNs. The sole purpose of the **route-distinguisher** is to allow the creation of distinct routes for a common IPv4 address prefix. It does not identify the origin of the route or the set of VPNs to which the route belongs to.

- The **route-target** community attribute is used to place the routes in the appropriate routing table(s). The sole purpose of the **route-target** is to identify which set of VPNs the route belongs to. It does not make routes unique as it not an attribute checked by the BGP decision process.

- When you are configuring L3VPNs you must configure both (configuration commit would fail otherwise).

Example using both the **route-target** and the **route-distinguisher**:

Two customers are advertising the same prefix (**10.1.0.0/24**):

customer 1 uses **route-distinguisher 100:1** and **route-target target:1:1**, and
customer 2 uses **route-distinguisher 200:1** and **route-target target:2:1**

The corresponding VPN-IPV4 prefixes (installed in **bgp.l3vpn.0**) are:

100:1:10.1.0.0/24, target:1:1
200:1:10.1.0.0/24, target:2:1

Thus, **route1** is advertising the VPN-IPV4 prefix **100:1:10.1.0.0/24**, while **route2** is advertising **200:1:10.1.0.0/24** => two different routes!

Using policies, the receiving router can be configured to import these routes into one of more L3VPN routing-tables (VRF routing instance).

For example, a policy can associate **target:1:1** with routing-instance ***customer1***, and **target:2:1** with routing-instance ***customer2***, resulting in the following routing-tables entries:

customer1.inet.0
10.1.0.0/24, target:1:1

customer2.inet.0
10.1.0.0/24, target:2:1

If ***customer1*** and ***customer2*** need to share the following routes to allow communication between some of their users:

route3(customer1)= 100:1:11.1.0.0/24, target:1:1,target:2:1
route4(customer2)= 200:1:12.1.0.0/24, target:1:1,target:2:1

NOTE: route3 and route4 are tagged with both **route-targets** **target:1:1**, and **target:2:1.**

applying the appropriate policies, that match on **target:1:1** and **target:2:1**, the corresponding routing tables will look like this:

customer1.inet.0

```
10.1.0.0/24, target:1:1
11.1.0.0/24, target:1:1,target:2:1
12.1.0.0/24, target:1:1,target:2:1
```

customer2.inet.0

```
10.1.0.0/24, target:2:1
11.1.0.0/24, target:1:1,target:2:1
12.1.0.0/24, target:1:1,target:2:1
```

In another example, assume that you need to place some routes from ***customer1*** and some routes from ***customer2***, in a shared routing instance named ***common***. To do this you configure your policies so that these routes are tagged with a common **route-target** (**target:3:1**).

Assume the routes from ***customer1*** and ***customer2*** that you need in the common routing-instance are:

```
Route5(customer1)= 100:1:15.1.0.0/24,target:1:1,target:3:1
Route6(customer2)= 200:1:16.1.0.0/24,target:2:1,target:3:1
```

By applying the proper policies, you can associate route-target target:3:1, with routing-instance ***common***, and the resulting routing tables would look like:

customer1.inet.0

```
10.1.0.0/24, target:1:1
11.1.0.0/24, target:1:1,target:2:1
12.1.0.0/24, target:1:1,target:2:1
15.1.0.0/24, target:1:1,target:3:1
```

customer2.inet.0

```
10.1.0.0/24, target:2:1
11.1.0.0/24, target:1:1,target:2:1
12.1.0.0/24, target:1:1,target:2:1
16.1.0.0/24, target:2:1,target:3:1
```

Common.inet.0

```
15.1.0.0/24,target:1:1,target:3:1
16.1.0.0/24,target:2:1,target:3:1
```