

Virtual Routing and Forwarding

- Feature Summary and Revision History, on page 1
- Revision History, on page 1
- Feature Description, on page 1
- Configuring VRF, on page 4
- VLAN Segmentation Using VRF, on page 6
- Monitoring and Troubleshooting, on page 8

Feature Summary and Revision History

Revision History

Table 1: Revision History

Revision Details	Release
UPF supports up to 200 VRFs for private APN/DNN.	2023.02.0
UPF supports up to 129 VRFs for private APN/DNN.	2022.04.0
Support is added for the following functionality: • Overlapping IP Pools • Removal of mandatory VRF ordering between SMF and UPF	2021.01.0
First introduced.	2020.02.0

Feature Description

Virtual Routing and Forwarding (VRF) is a technology that allows multiple instances of a routing table to coexist within the same router at the same time. As the routing instances are independent, VRF uses the same or overlapping IP addresses without conflicting with each other.

Note In 2023.02.0 and later releases, UPF supports up to 200 VRFs for private APN or DNN.

In releases prior to 2023.02.0, UPF supported up to 129 VRFs for private APN or DNN.

In UPF, this feature enables association of IP address pools with VRF. The chunks from this pool are allocated to the UPFs that are configured to use these pools. VRF-associated pools in UPF can be either Static or Private type.

When UPF comes up for registration, the chunks in the PRIVATE VRF pool are allocated similar to the normal private pools. For a Static VRF pool, SMF does chunk allocation to UPF during configuration. An Sx-Route-Update message is sent for pre-allocated static chunks during UPF registration.

Overlapping IP Pool

Overlapping pools share and use an IP address range. Overlapping pools can either be of Static or Private type. Public pools cannot be configured as overlapping pools. Each overlapping pool is part of a different VRF (routing domain) and pool-group. Since an APN can use only one pool-group, overlapping pools are part of different APNs.

Without this functionality, overlapping pools are configured at SMF. However, chunks from two overlapping pools cannot be sent to the same UPF. That is, the UPF cannot handle chunks from two different overlapping pools. Same number of UPFs and overlapping pools are required for sharing the same IP address range.

With this functionality, UPF handles chunks from two different overlapping pools. So, a single UPF can handle any number of overlapping pools sharing the same IP range.

The functionality of overlapping pools in the same UPF includes:

- When a chunk from a particular pool is installed on UPF, its corresponding vrf-name is sent along with the chunk.
- The UPFs are VRF-aware of the chunks and install chunks on the corresponding VRFs. The chunk database is populated under VRFs.
- During call allocation, release, recovery, or any communication toward VPNMgr, the corresponding SessMgr at UPF includes vrf-id. This enables VPNMgr to select the correct chunk for that IP under the provided vrf-id for processing.

UE IP VRF is a custom IE that encapsulates the VRF name of N4 SESSION ESTABLISHMENT REQUEST message.

UE IP VRF Information Element

The following is the IE format of the private UE IP VRF.

Table 2: UE IP VRF Format

	Bits							
Octets	8	7	6	5	4	3	2	1
1 to 2	Type = 242 (decimal)							
3 to 4	Length = n							

5	Spare	Identical VRF flag	IPv6 VRF Valid	IPv4 VRF Valid
m to m+1	VRF-1 Name Length = p			
m+1 to m+1+p	VRF-1 Name			
n to n+1	VRF-2 Name Length = q			
m+1 to m+1+q	VRF-2 Name			

The following table shows the possible values of the "UE IP VRF" fields.

Cases	UE IP VRF	Value (binary)		
		Bit 3	Bit 2	Bit 1
1	None of the IPv4 and IPv6 UE IP addresses are associated to VRF.	0	0	0
2	Only IPv4 UE IP address is associated to a VRF.	0	0	1
3	Only IPv6 UE IP address is associated to VRF.	0	1	0
4	Both IPv4 and IPv6 UE IP addresses are associated to different VRFs.	0	1	1
5	Both IPv4 and IPv6 UE IP addresses are associated to a common VRF.	1	1	1

VRF Name as Identifier

The communication between SMF and UPF, related to VRF, was done through vrf-id. This required the operator to have all VRFs configured in both SMF and UPF, and also in the same order.

With this feature, vrf-name is used as identifier in all the communication between SMF and UPF related to VRFs. This feature eliminates the configuration of all VRFs in UPF. Operator can configure VRFs in different order at SMF and UPF, and can identify the VRF with the same vrf-name in both the nodes.

Limitations and Restrictions

The following are the known limitations and restrictions in UPF:

- UPF supports only VRF-based overlapping pools. UPF does not support overlapping pools such as NH-based and VLAN-based pools.
- UPF does not permit PDN Type IPv4v6-based call on static IP pools with multiple UPFs in the same UPF group.
- UPF does not support dynamic update of VRF.

Configuring VRF

Use the following steps to configure VRF support in UPF.

At SMF:

1. Create the APN or DNN profile.
2. Create overlapping IP pools and associate the respective APN or DNN and VRF at context-level.
3. Associate APN or DNN to the UPF profile.

The following is an example of the SMF configuration:

```
profile dnn intershat1
.
.
.
   upf apn mpls1.com
exit
profile dnn intershat2
.
.
.
   upf apn mpls2.com
exit
profile network-element upf upf1
.
.
.
   dnn-list [ intershat1 intershat2 ]
exit
profile network-element upf upf2
.
.
.
   dnn-list [ intershat1 intershat2 ]
exit
ipam
   source local
   address-pool pool-intershat1
      vrf-name mpls-vrf-1@isp
      tags
         dnn intershat1
      exit
      ipv4
         address-range 209.165.201.25 255.255.255.224
      exit
   exit
   address-pool pool-intershat2
      vrf-name mpls-vrf-2@isp
      tags
         dnn intershat2
      exit
      ipv4
         address-range 209.165.201.25 255.255.255.224
      exit
   exit
exit
```

At UPF:

It is recommended to configure VRF in UPF before a chunk is pushed from SMF. Else, it leads to failure of the complete IP pool transaction (including chunks that donot belong to the VRF). SMF retries the attempt after some time.

The following is an example of the UPF configurations:

UPF 1:

```
config
  context EPC2
    sx-service sx
      instance-type userplane
      bind ipv4-address 209.165.201.11 ipv6-address bbbb:aaaa::4
    exit
    user-plane-service up
      associate gtpu-service pgw-gtpu pgw-ingress
      associate gtpu-service sgw-ingress-gtpu sgw-ingress
      associate gtpu-service sgw-engress-gtpu sgw-egress
      associate gtpu-service saegw-sxu cp-tunnel
      associate sx-service sx
      associate fast-path service
      associate control-plane-group g1
    exit

  context isp
    ip vrf mpls-vrf-1
    #exit
    ip vrf mpls-vrf-2
    #exit
    ip vrf mpls-vrf-1
        route-distinguisher 61601 11100001
        route-target export 61601 11100001
        route-target import 61606 11100001
        route-target import 65200 11100001
    #exit
    address-family ipv4 vrf mpls-vrf-1
        redistribute connected
        redistribute static
    #exit
    address-family ipv6 vrf mpls-vrf-1
        redistribute connected
        redistribute static
    #exit
    ip vrf mpls-vrf-2
        route-distinguisher 61601 11100002
        route-target export 61601 11100002
        route-target import 61606 11100002
        route-target import 65200 11100002
    #exit
    address-family ipv4 vrf mpls-vrf-2
        redistribute connected
        redistribute static
    #exit
    address-family ipv6 vrf mpls-vrf-2
        redistribute connected
        redistribute static
    #exit
    #exit
apn mpls1.com
      pdp-type ipv4 ipv6
      bearer-control-mode mixed
      selection-mode sent-by-ms
      ip context-name isp
    exit
```

```
exit
control-plane-group g1
    peer-node-id ipv4-address 209.165.201.15
  #exit
  user-plane-group default
```

UPF 2:

```
config
  context EPC2
    sx-service sx
      instance-type userplane
      bind ipv4-address 209.165.201.12 ipv6-address bbbb:aaaa::5
    exit
    user-plane-service up
      associate gtpu-service pgw-gtpu pgw-ingress
      associate gtpu-service sgw-ingress-gtpu sgw-ingress
      associate gtpu-service sgw-engress-gtpu sgw-egress
      associate gtpu-service saegw-sxu cp-tunnel
      associate sx-service sx
      associate fast-path service
      associate control-plane-group g1
    exit
exit

  context isp
    ip vrf mpls-vrf-1
    #exit
    ip vrf mpls-vrf-2
    #exit
    apn mpls2.com
      pdp-type ipv4 ipv6
      bearer-control-mode mixed
      selection-mode sent-by-ms
      ip context-name isp
    exit
exit

control-plane-group g1
    peer-node-id ipv4-address 209.165.201.15
  #exit
  user-plane-group default
```

VLAN Segmentation Using VRF

Table 3: Feature History

Feature Name	Release Information	Description
Enabling VLAN Segmentation of Traffic Using Virtual Routing and Forwarding	2024.01	UPF allows VLAN segmentation using VRF. A VRF is configured for each VLAN and each VRF has a default route through which the uplink packets are sent. **Default Setting**: Disabled-Configuration required to enable.

Feature Description

UPF supports segmenting the enterprise traffic using Virtual LAN (VLAN). VLANs enable a single physical LAN segment to be further segmented into smaller virtual networks so that the groups of ports are isolated from one another, as if they were on physically different segments. In VLAN segmentation, the UEs belonging to different classes are assigned to different VLANs for traffic isolation.

VLAN Segmentation can be achieved using two methods:

1. Next Hop Forwarding
2. Virtual Routing and Forwarding

UPF supports segmenting VLAN traffic by configuring VRF. It is a layer-3 virtualization that consists of configuring a VRF with a defaut route per VLAN. The VRF default route is used for sending packets.

VLAN segmentation is enabled by logically mapping the VLAN tags with the IP pools at the SMF. The UPF is informed about the mapping configuration based on which UPF forwards the uplink packets on the N6 interface through the default route configured for a VRF.

Enabling VLAN Segmentation Using VRF

To enable VLAN segmentation using VRF on UPF, use following configuration:

```
config
  port ethernet slot/port
    no shutdown
    vlan vlan_tag_ID
    no shutdown
    bind interface interface_name context_name
    end
    context context_name
      ip vrf vrf_name
      { ip | ipv6 } address address subnetmask [ secondary ]
      end
```

NOTES:

- **ip vrf** *vrf_name*—Creates a VRF in the router and assigns a VRF-ID.

Configuration Example

Following is a sample configuration for enabling VLAN segmentation using VRF:

```
config
    port ethernet 1/10
       no shutdown
       vlan 400
         no shutdown
         bind interface N6_interface1 ISP
       #exit
       vlan 401
         no shutdown
         bind interface N6_interface2 ISP
       #exit
 context ISP
```

```
   ip vrf vrf1
   #exit
   ip vrf vrf2
   #exit
interface N6_interface1
     ip vrf forwarding vrf1
     ip address 209.165.200.225 209.165.200.254
     ip address 209.165.201.1 209.165.200.254 secondary
     ipv6 address 2001:DB8::1/32 secondary
interface N6_interface2
     ip vrf forwarding vrf2
     ip address 209.165.202.129 209.165.200.254
     ip address 209.165.201.30 209.165.200.254 secondary
     ipv6 address 2001:DB8:1::1/32 secondary
```

Monitoring and Troubleshooting

This section provides information regarding the CLI commands available for monitoring and troubleshooting the feature.

Show Commands and Outputs

This section provides information regarding show commands and their outputs in support of this feature.

show ip chunks

The output of this CLI command displays all chunks in that context.

With the Overlapping IP Pools functionality, the **show ip chunks vrf** *vrf_name* CLI command displays only chunks under that VRF.

- chunk-id
- chunk-size
- vrf-name
- start-addr
- end-addr
- used-addrs
- Peer Address

Following is a sample output:

```
==========================================
VRF Name: MPN00001
==========================================================================================================
|------------------------------|----------|------------|--------------|--------------|------------|
|          Peer Address          | chunk-id | chunk-size |   start-addr  |    end-addr   |
 used-addrs |
|------------------------------|----------|------------|--------------|--------------|------------|
|                    192.10.25.23|1074790405|        8192|    36.40.128.0|  36.40.159.255|
           0|
|                    192.10.25.23|1074790406|        8192|    36.40.160.0|  36.40.191.255|
```

```
            0|
|-------------------------------|----------|-----------|--------------|--------------|-----------|
```

show ipv6 chunks

The output of this CLI command displays all chunks in that context.

With the Overlapping IP Pools functionality, the output of the **show ipv6 chunks vrf** *vrf_name* CLI command displays only chunks under that VRF.

- chunk-id
- chunk-size
- vrf-name
- start-prefix
- end-prefix
- used-prefixes
- Peer Address

show ip bgp vpnv4

The output of this CLI command displays all the VPN routing information.

With the Overlapping IP Pools functionality, the **show ip bgp vpnv4 vrf** *vrf_name* CLI command displays the information under that VRF.

Following is a sample output:

```
VPNv4 Routing Table:
BGP table version is 1, local router ID is 172.31.35.36
Status Codes: s suppressed, d damped, h history, * valid, > best, i - internal, S stale, m
 Multipath
Origin Codes: i - IGP, e - EGP, ? - incomplete

Network              Next Hop             Metric LocPrf Weight Path
*>   0.0.0.0/0            172.31.35.29         0             0      65200 ?
*>   2.2.2.101/32         172.31.35.29         0             0      65200 ?
*>   2.2.3.2/32           0.0.0.0              0             32768  ?
*>   36.40.0.0/19         0.0.0.0              0             32768  ?
*>   36.40.32.0/19        0.0.0.0              0             32768  ?
*>   36.40.192.0/19       0.0.0.0              0             32768  ?

Total number of prefixes 6
```

show ip bgp vpnv6

The output of this CLI command displays all the VPN routing information.

With the Overlapping IP Pools functionality, the **show ip bgp vpnv6 vrf** *vrf_name* CLI command displays the information under that VRF.

Following is a sample output:

```
VPNv6 Routing Table:
BGP table version is 1, local router ID is 172.31.35.36
```

```
Status Codes: s suppressed, d damped, h history, * valid, > best, i - internal, S stale, m
 Multipath
Origin Codes: i - IGP, e - EGP, ? - incomplete

 Network              Next Hop             Metric LocPrf Weight Path
*>   0.0.0.0/0            172.31.35.29         0             0      65200 ?
*>   2036:adb0:40         172.31.35.29/51         0             32768  ?
*>   2036:adb0:40:2000    172.31.35.29/51           0             32768  ?
*>   2036:adb0:40:c000    172.31.35.29/51            0             32768  ?

Total number of prefixes 4
```

show mpls ilm

The output of this CLI command displays the MPLS ILM table with FEC information.

With the Overlapping IP Pools functionality, the **show mpls ilm** *fec verbose* and **show mpls ilm** *fec summary* CLI commands display the information under the VRF.

The following is a sample output for the **show mpls ilm** *fec summary* CLI command to display the count.

```
Total ILM entries: 406
```

The following is a sample output for the **show mpls ilm** *fec Verbose* CLI command to display a detailed MPLS ILM table.

```
In-segment entry with in label: 832, id: 99, in label-space: 0, row status: Active
Owner: BGP, # of pops: 1 XC Index:708
FEC: 36.40.0.0/19
FEC: 36.40.32.0/19
FEC: 36.40.192.0/19

In-segment entry with in label: 833, id: 100, in label-space: 0, row status: Active
Owner: BGP, # of pops: 1 XC Index:709
FEC: 2037:adb0:201::/51
FEC: 2037:adb0:201::/51
FEC: 2037:adb0:201::/51
```