User Guide, Site Router

USER GUIDE

1/1553-LZA 104 102/8 Uen C

Copyright

© Ericsson AB 2012. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Contents

1 Introduction

This chapter outlines the scope, structure and intended target groups of this document.

1.1 Scope

This document describes the Site Router functionality (routing, switching and network services) supported by the T12B software release for the SIU 02 (Site Integration Unit). For further information about this unit, refer to the Reference list.

Note: In this document, where not explicitly mentioning IPv6, a description only applies for IPv4.

1.2 Target Group

The intended target groups for this document are:

- Radio Network Engineers.
- System Administrators.
- Field Technicians.
- Installation and Integration personnel.
- Site Engineers.

1.3 Main Changes

For information about the main changes in this Ericsson release, see Reference [1].

2 Capabilities

This chapter gives an overview of the IP transport functionality.

2.1 Physical Layer

The IP transport functionality includes traffic from site equipment with IP over Ethernet (IPoE) interface connected to the site LAN ports (Ethernet/SFP). The E1/T1 interfaces can be used as PPP links in an ML-PPP bundle (1 to 8 links in the bundle), see Reference [7], as Abis Lower connections to GSM RBSs, see Reference [13], or as general-purpose E1/T1 interfaces in CESoPSN scenarios, see Reference [8].

For further information about connectors on the SIU, see the document *SIU 02 Description* (Reference [3]).

2.2 Data Link Layer

The following data link layer features are supported:

- Ethernet (IEEE 802.3 only). A Maximum Frame Size of 2112 bytes is supported.
- DHCP server;
  Allows to server Dynamic Host Configuration Protocol on IPv4 network.
- ML-PPP;
  Multi-Link PPP is a bandwidth-on-demand protocol that can connect multiple links between two systems as needed to provide bandwidth on demand. Maximum Received Reconstructed Units (MRRU's) up to 2084 bytes are supported.
- Bridged VLAN;
  Allows for configuration of an Ethernet domain between two or more Ethernet interfaces. A Maximum Frame Size of 2112 bytes is supported.
- Remote port loopback activation according to the ITU-T G.704 and TDM path quality measurement according to the ITU-T G.826. For more information, see Reference [7].

2.3 Network Layer

The following network layer features are supported:

- IP Routing;

IP Routing is an umbrella term for the set of protocols that determine the path that data follows in order to travel across multiple networks from its source to its destination.

- Static routes;
  Up to 64 static IPv4 routes and 64 static IPv6 routes can be configured (in addition to the automatically generated routes for attached subnets).
- Policy Based Routing (PBR);
  As a complement to destination based routing, the PBR feature enables advanced routing capabilities. With PBR, a fine-grained routing decision can be made by filtering fields in the IP and VLAN header.
- BFD Connectivity Check over IPv4 and IPv6;
  Bidirectional Forwarding Detection (BFD) is a protocol intended to detect faults in the bidirectional path between two forwarding engines. BFD in the SIU is single-hop and can be used to monitor the connectivity to the next hop routers.
- OSPFv2 (Open Shortest Path First version 2);
  The OSPFv2 feature enables adaptive routing capabilities. With OSPFv2, the SIU constructs a map of the connectivity to the network and calculates the next best logical path from it to every possible destination in the network.

• CES PWE;
  Circuit Emulation Service PseudoWire Emulation (CES PWE) introduces a possibility to transport TDM links over IP. For more information, see Reference [8].

• Routed VLAN;
  Allows using a VLAN for IP routing by attaching an IP interface to a specific VLAN.

• Access Control Lists (ACLs);
  ACLs are used to filter ingress and egress traffic according to user-defined rules. For more information, see Reference [4].

• DSCP - PCP mapping.

2.4 QoS

• QoS;
  For example mapping of DSCP to priority queues and bit-rate limiting.

• Ethernet Frame re-mapping;
  PCP-to-Queue and DSCP-to-Queue.

3 Technical Description

This chapter provides a technical description of site router capabilities in the network.

3.1 Ethernet Based Transmission Network

The figure below illustrates an example of nodes that can be included in an Ethernet based transmission network. The picture shows a co-existence scenario with GSM, CDMA, WCDMA and LTE. The dotted lines illustrates nodes that can be positioned in a site.

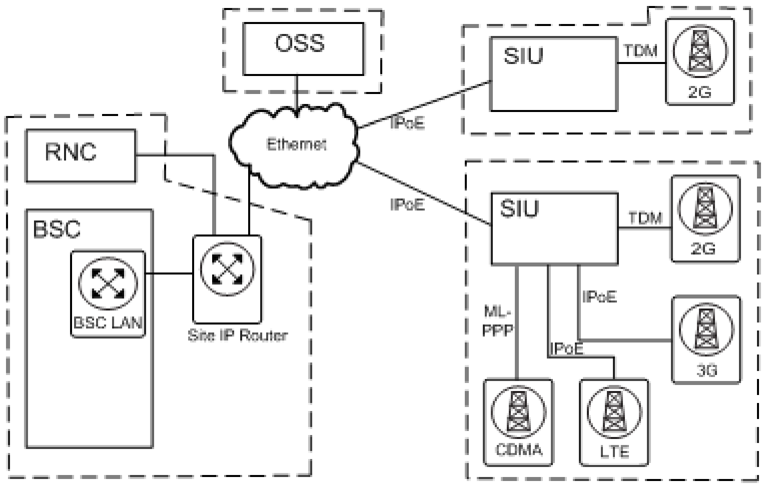

Figure 1 SIU in an IP over Ethernet based transport network

Other Ethernet hosts at the RBS site may also utilize the shared transport through the feature Site LAN. There are no specific requirements on the topology or structure of the Ethernet network but to successfully deliver GSM, WCDMA, LTE and CDMA services, the Ethernet network must meet certain characteristic requirements regarding delay, packet loss, bandwidth and jitter. These requirements are the same as for GSM, WCDMA, LTE and CDMA separately, even though the different technologies affect each other when sharing the same transport network.

As aggregation nodes for the combined GSM, WCDMA, LTE and CDMA traffic, the router at the Site IP Router and the SIU implement the necessary shaping to stay within the bandwidth limits of the Ethernet transport Service Level Agreement (SLA). If sufficient bandwidth is not available, either due to temporary over-use or under-dimensioning, packets are dropped by the routers according to their respective priority.

The proposed mapping of traffic types to DSCP and the configuration of the SIU and router at the Site IP Router, implements a scheme where high priority traffic have precedence over other traffic types, for example, in case of network overload. Ethernet packets may also be dropped in intermediate transport nodes due to for example over-commitment and failures without obeying the DSCP classification. Well functioning DSCP requires that the ratio of such packet drops are at an insignificant level.

On the RBS site, the GSM RBS is connected to the SIU as described in the Packet Abis over IP feature, see Reference [13]. The needed conversion between the protocol used on the Ethernet transport and the protocol on the TDM links towards the GSM RBS are carried out by the SIU. The CDMA, WCDMA and LTE RBSs use Ethernet natively and the SIU acts as an IP router only. No protocol conversion is needed for this traffic.

3.2 Ethernet Bridging

The Ethernet bridging feature provides basic bridging of VLANs based on VLAN IDs (802.1Q). In opposite to a non-bridged configuration where each Ethernet interface has its own IP subnet this feature allows bridging of VLAN tagged Ethernet traffic between different Ethernet interfaces.

A typical use case for bridging is when the SIU is introduced in an existing LTE or WCDMA network without impact on existing IP addresses. In this scenario bridging can be used between the interfaces used for the site connections and the transport network.

With Bridge Virtual Interface (BVI) it is possible to have IP host functionality on an L2 Bridge. The IP host acts as an IP gateway to the routing function. The IP host can also be used for remote connectivity to the SIU, for example by O&M configuration management, and as an IP host for other internal applications. At least 8 instances of BVI are supported. To attach an IP interface to a bridge, set the *depLinkLayer* attribute on the MO **IPInterface** to the MO **Bridge** corresponding to the bridge.

If no IP host is configured on the bridge, the bridged ports are functioning as a standalone switch.

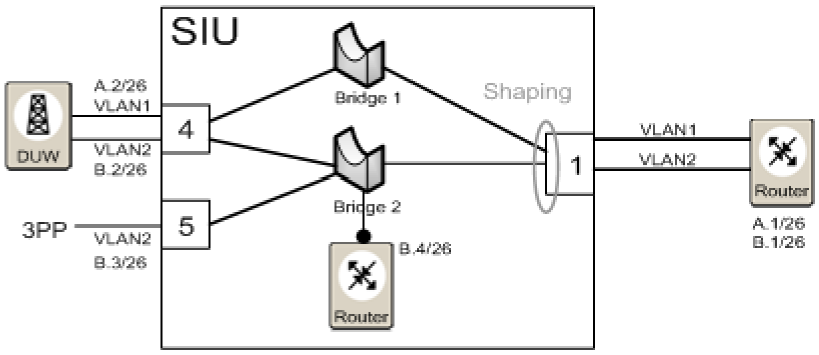

Figure 2 Mix of Bridging and Routing Scenario with BVI

The figure above shows an example where a SIU is added to an existing radio site in order to provide L2 connectivity on the site LAN between the DUW

(Digital Unit WCDMA) and 3PP (3rd Party Product) equipment. Furthermore, the traffic from the DUW and 3PP equipment has to be bridged towards the L2 transport network. The IP address plan of the site is not impacted by the introduction of the SIU.

An IP host is added to the Bridge 2 that uses VLAN2 and an IP address (B.4) in the same subnet with the 3PP and the DUW in VLAN2. The traffic from 3PP or DUW, having as destination the SIU with IP address B4, is not sent to the router and back over the WAN port (port 1) in the SIU. This traffic ends up on the SIU's BVI interface (B.4). In short, all the site equipment can use the same subnet for O&M (VLAN2).

3.3 DHCP Relay

The DHCP relay feature allows hosts connected to the site access ports to get their configuration parameters from a remote DHCP server. Broadcast packets will be forwarded as routable unicast packets to the DHCP server. The DHCP servers are organized in DHCP server groups and a DHCP request is forwarded to all servers included in a group to provide DHCP server redundancy.

Up to eight DHCP server groups can be configured with up to four DHCP servers in each. DHCP relay support is enabled per IP interface by a reference in the MO **IPInterface** to a DHCP server group.

3.4 DHCP Server

The SIU includes DHCP server capabilities to serve hosts on IP network. The functionality can be applied to one or more IP interfaces. The offered IP address range and network subnets are derived from the dependent IP interface configuration. The range is defined as a maximum number of 50 consecutive IP addresses above the primary address of the dependent IP interface. The number of offered IP addresses can be less than 50 as it is additionally limited by the subnet size.

There can be up to four global user defined DHCP options. There is also one automatically added option named routers with the value set to *primaryIP_Address* of the serving IP interface.

A DHCP server is also available on the local console (CONSOLE) port allowing a local maintenance terminal to detect IP settings automatically. This DHCP server is started automatically (together with the SSH and SFTP services) provided that the ports default configured network mask (255.255.255.0) has not been changed.

3.5 IPv4/IPv6 Dual Stack

The IP dual stack feature includes support for both IPv4 and IPv6. It is provided as a means to support IPv6 transport, or a migration mechanism from IPv4 to

IPv6. The IPv6 feature supports transmission of IPv6 packets over Ethernet networks, an IPv6 address architecture, ICMPv6, Neighbor Discovery (ND), and Duplicate Address Detection (DAD). This allows for example LTE to run on IPv6-only, while providing IPv4 transport for WCDMA at the same time.

The current implementation of IPv6 support the following functionality (in addition to IPv4):

- IPv6 interfaces.
- Virtual IPv6 interfaces.
- IPv6 static routes together with BFD over IPv6.
- IPv6 bridging.
- VLAN.
- O&M via CLI of Software Management (SM), Configuration Management (CM), and Performance Management (PM).

Note: In this document, where not explicitly mentioning IPv6, a description only applies for IPv4.

3.5.1 IPv6 Interfaces

An IPv6 interface is created with default, or user specified values. A link-local address is automatically generated according to RFC 4862 (Reference [15]). The link-local address is mainly used for automatic address configuration and neighbor discovery. It is possible to add global addresses to the interface using the contained MO **IPv6Address** and this also triggers a system created MO **IPv6RouteSys** instance per added IPv6 address. Router Advertisement (RA) prefixes can also be added to an IPv6 interface using the contained MO **IPv6AdvPrefix**.

3.5.1.1 Neighbor Discovery

Neighbor Discovery (ND) is an ICMPv6 function that enables a node to identify other hosts and routers on its link. The Neighbor Discovery Protocol (NDP) is specified in RFC 4861 (Reference [16]). NDP defines the following ICMPv6 packet types:

- Router Solicitation;
  Hosts send Router Solicitations (RS) in order to prompt routers to generate Router Advertisement (RA) quickly.
- Router Advertisement;
  The router sends RA messages to a local Ethernet LAN periodically, or in response to a RS.
- Neighbor Solicitation;
  Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target.

Neighbor Solicitations are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbor.

- Neighbor Advertisement;
  A node sends Neighbor Advertisements in response to Neighbor Solicitations and sends unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly.

- Redirect;
  Routers send Redirect packets to inform a host of a better first-hop node on the path to a destination. The SIU will send redirects, but for security reasons it will not respond or act upon received redirect packets.

3.5.1.2 Duplicate Address Detection

When the SIU has configured IPv6 addresses, it must perform Duplicate Address Detection (DAD) to ensure that its configured addresses are unique on the link.

In IPv6, the DAD procedure is defined in RFC 4862 (Reference [15]), and uses the ND procedures defined in RFC 4861 (Reference [16]). The SIU cannot start to use a configured address until the DAD procedure has been successfully executed. Until DAD has succeeded, the address is seen as tentative, in that it can only be used for ND purposes (of which the DAD procedure is part of). The attributes *tentative* and *dadFailed* in MO **IPv6Address** show the state of the DAD procedure as described in the table below:

Table 1 DAD procedure states

Attribute	During DAD	IPv6 address operational	DAD failed
tentative	true	false	true
dadFailed	false	false	true

To perform DAD, the SIU sends out a neighbor solicitation message with its own address as the target address of the solicitation message. The destination address in the IPv6 header of the neighbor solicitation is set to the solicited-node multicast address of the target address with the source address being the unspecified address. If there is another node on the link that uses the same address, one out of two things will happen:

- The duplicate node will receive the SIU's neighbor solicitation message and reply with a neighbor advertisement (sent to the all-nodes multicast address) thus exposing the duplicated address to the SIU.

- The SIU will receive a neighbor solicitation with its address as the target address from a duplicate node that is also in the process of performing DAD.

According to RFC 4862 (Reference [15]), a node performing DAD can consider its tentative address unique if no indications of a duplicate address are observed within RETRANS_TIMER milliseconds after sending DUP_ADDR_DETECT_TRANSMITS number of neighbor solicitations. These parameters are by default set to 1,000 and 1 respectively. Therefore, under default conditions, DAD will take a minimum of 1,000 milliseconds plus additional delay for link transmissions.

Note: RFC 4862 states that a node should delay sending its neighbor solicitation for DAD by a random time interval between 0 and MAX_RTR_SOLICITATION_DELAY seconds if it is the first packet sent from the interface after (re)initialization. In RFC 4861, MAX_RTR_SOLICITATION_DELAY is defined as being 1 second in duration. Therefore, unless the SIU has previously sent a router solicitation, it will incur further delay during its auto configuration process. In the average case (assuming a pure random function) this will be an extra 500ms, and up to 1000ms (1 second) in the worst case.

3.6 Virtual IP Interfaces

Virtual IPv4/IPv6 interfaces (loopback IP addresses) can be used in order to be able to use resilient static routes with IP addresses independent from any physical interface for its IP termination.

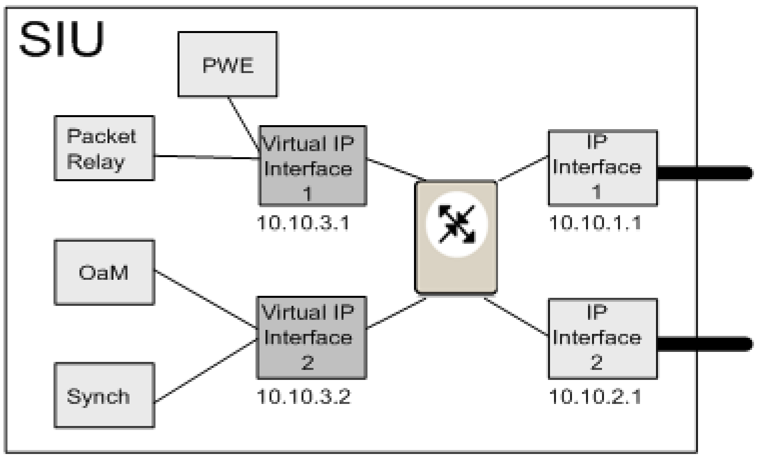

Figure 3 Virtual IP Interfaces (loopback IP addresses)

Each virtual IP interface is configured with a single IP address (network mask /32 for IPv4 and /128 for IPv6) as there is no further routing possible. The other routers/hosts in the network are configured with a static route to the SIU virtual IP interface address via a physical interface address. If there are multiple connections to the transport network, multiple alternate routes are possible.

There are no counters on virtual IP interfaces, but for IPv4 the ACL counters can be used to count the number of packets in and out from each virtual IP interface IP address.

3.7 Static IP Routes

Up to 64 static IPv4 and/or IPv6 routes can be configured (in addition to the automatically generated routes for attached subnets).

Unless a default gateway is defined, each distinct IPv4 subnet, that must be reachable, must have an instance of MO **IpRoute** configured.

3.8 Resilient Static Routes

When using multiple uplink access interfaces, it is essential to have a fast detection when a route looses connectivity, so that impacted traffic can be moved to an alternative route. Resilient static routes is supported for both IPv4 and IPv6 and enable redundant connections utilizing different interfaces. The individual connections can function as backup for each other. Two mechanisms are used together to give resilience:

- Port availability status.
- BFD (Bidirectional Forwarding Detection) connectivity check.

3.8.1 Port Availability Status

Port availability status for resilient handling is applied regardless BFD is enabled or not.

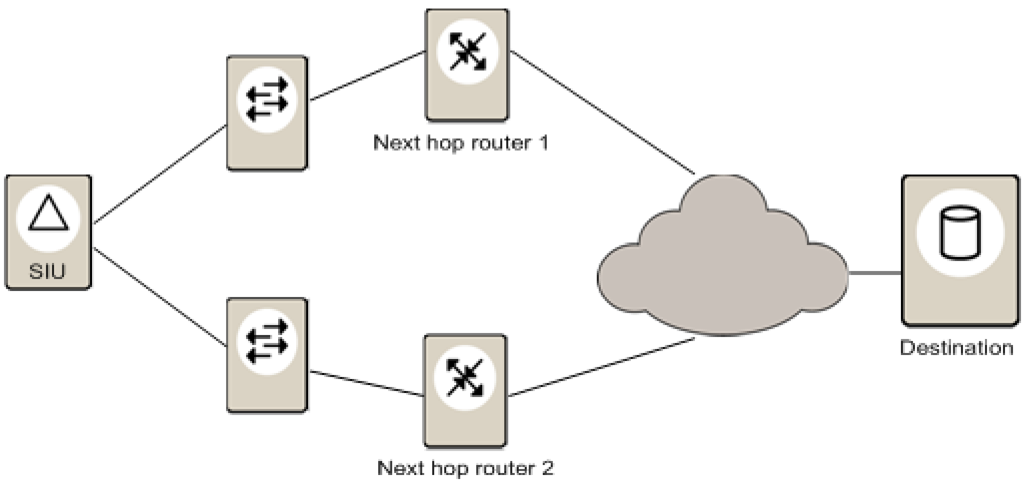

Figure 4 Resilient Static Routes using Port Availability Status Monitoring

In this case, if one Ethernet port is down, all static routes configured on that port will be withdrawn. The alternative route for the same destination (with longer *admDistance* / *metric*) on another port, if any, will take over. When the initial Ethernet port is up again, the route with lower *admDistance* / *metric* will be applied again.

3.8.2 BFD Connectivity Check

It is possible to configure single hop BFD connectivity check to next hop routers, as specified in RFC 5881, Reference [18]. When BFD is enabled by initiating connectivity check for a given IPv4 or IPv6 interface, it creates a BFD session for that IP interface towards a given next hop router. The next hop router(s) are similarly configured to supervise the connectivity. The BFD session has a status that reflects whether connectivity is "up" or "down" towards the next hop router.

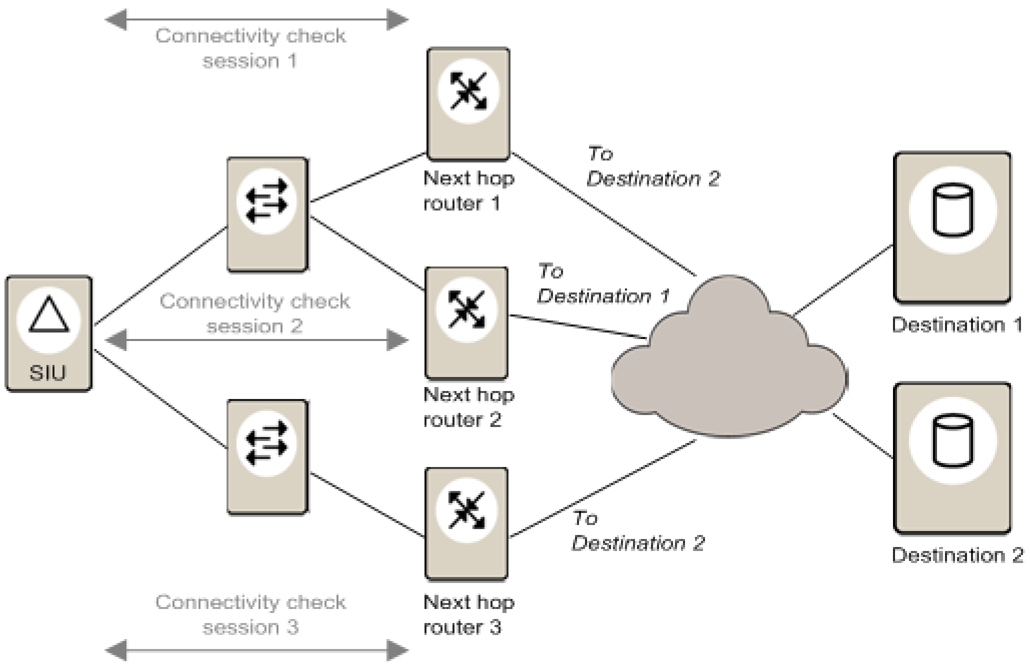

Figure 5 Resilient Static Routes using BFD

When the mutual connectivity check detects loss of connectivity, affected static routes are withdrawn from the data plane Forwarding Information Base (FIB) in both ends. The route is however still configured in each control plane routing table. A hold-down timer on the connectivity check ensures that a flapping interface or link will not cause flapping routes when the connectivity is restored.

BFD is not supported over ML-PPP as ML-PPP has it’s own link failure detection.

3.9 Policy Based Routing

As a complement to IPv4 destination based routing, the Policy Based Routing (PBR) feature enables advanced routing capabilities. Traffic can be routed based on different criteria using policies. These policies are used to specify how traffic is forwarded based on the traffic type and traffic priority. A more fine-grained routing decision can be made, based on filtering of fields in the IP and VLAN header, see Table 2.

Table 2 Filtering Fields in the IP and VLAN Header

IP Header	**VLAN Header**
Destination IP	PCP
Source IP	
Protocol	
Destination port	
Source port	
DSCP	

A routing policy is an ordered list of routing rules, each containing a set of filter criteria. When a routing policy has been applied on an ingress IP interface, all ingress packets on that interface will be matched against the rules in the routing policy. When a matching rule is found, a forwarding decision is made, and the packet will be forwarded according to the forwarding action route associated with the rule. No further matching will take place. Each routing rule has one or two forwarding action routes, where one is the primary and the other is secondary.

To be able to make a failover to an alternative route in case of a failure on the primary route, either due to physical link being down or lack of connectivity to the next hop router, the PBR feature has support for resilience, that is two alternate forwarding routes per policy rule. If both forwarding routes are failing, this policy rule will not be considered for matching and packets matching this rule will revert to destination based routing. For a given policy rule that has been withdrawn, when a route becomes available, the rule will be re-instated and the route will become active. The individual routes used in PBR forwarding actions are behaviorally equivalent with static routes for availability state, connectivity check, hold-down timers, and fail-over/fail-back principles. PBR has support for using existing next-hop monitoring features (for example BFD), see Section 3.8.2 on page 12.

Note: It is possible to configure a rule with filters, which match packets - based on their destination IP address - that are intended for local delivery. Such a rule could jeopardize the functionality, for example if it matches BFD packets, so these will be forwarded instead of locally delivered.

Note: BFD should be used on both the primary and secondary route (if used) in a routing rule to reach optimal fail-over performance. If BFD is not used on the secondary route, an ARP resolution is required on the secondary routes next-hop before traffic can be forwarded.

A routing policy can be applied to a single ingress IP interface or to multiple ingress IP interfaces, covering packets received on that interface or interfaces. A routing policy can be applied to IP packets received from external hosts or to packets originated from an internal host. The latter should be considered a special case where PBR is applied to traffic originating from host applications on the node. In this case the routing policy will be attached to the system itself instead of an IP interface.

For IP traffic originating on the SIU - such as O&M traffic - static routes to the relevant destination subnets must be configured in the routing table. This is true even in the case where a given traffic flow will actually be subject to PBR rather than following the static route configured. If no static route exists to the destination subnet for a given packet, the packet will be dropped.

Each individual filter criteria can be wildcarded independently of other filter criteria. Individual routing rules in a routing policy are evaluated in an assigned priority order.

PBR has precedence over any static route but does not replace destination based routing, that is packets not matching any routing policy rule will be evaluated for destination based routing. ACLs have precedence over PBR.

PBR supports up to 64 active routing policies with a maximum of 32 rules per routing policy. There is a counter for packets matching each routing policy rule. This counter shows the number of times a particular rule in a routing policy has been matched by a packet.

The figure below illustrates an example of forwarding traffic based on IP DSCP value (DSCP is pre-marked and no remarking is done), where this information is used to assist in the forwarding path decision. For example, the voice traffic may be marked with a high priority while the data traffic is best effort.

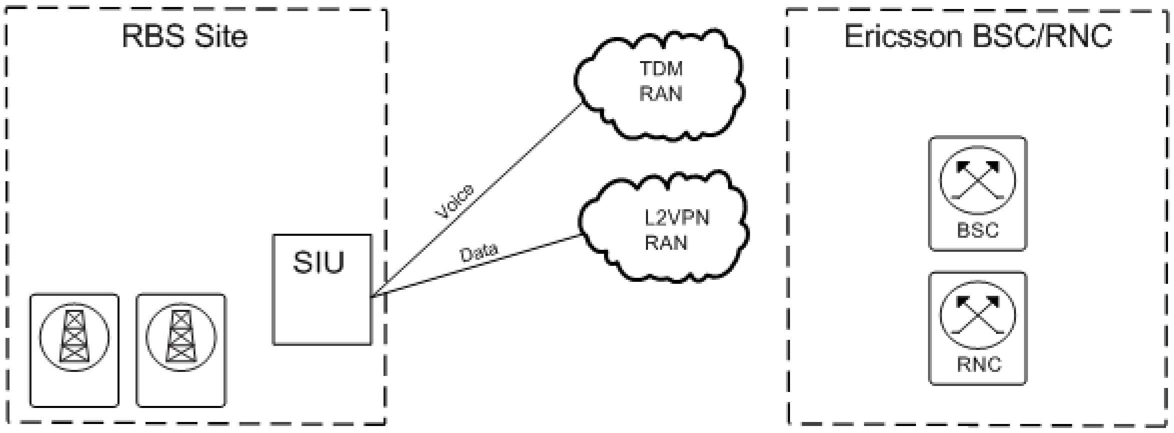

Figure 6 Forwarding decision based on pre-marked IP DSCP

In the figure above:

- Voice traffic is routed via "TDM RAN" (ML-PPP).
- All packet data traffic is routed via "L2VPN RAN".

The figure below illustrates an example of PBR with data path resilience. In this case the traffic is divided by the combination of IP header 5-tuple + DSCP or PCP value. If the primary forwarding data path to the next hop is not available, the traffic is routed via the secondary data path.

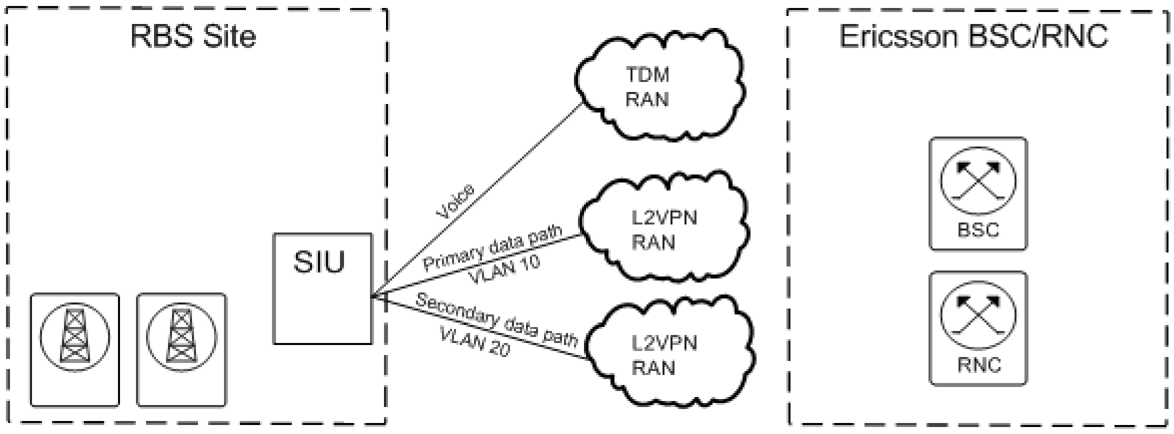

Figure 7 PBR with data path resilience

In the figure above:

- Voice traffic is routed via "TDM RAN" (ML-PPP).
- All packet data traffic is routed via primary or secondary "L2VPN RAN".

3.9.1 Fragmented Packets Processing in PBR

PBR fragmented packets processing is identical to that for unfragmented packets.

Filtering fragmented packets poses a problem when Layer 4 filters are applied. Layer 4 headers are included only in the first fragment of a packet, so for subsequent fragments, layer 4 filters cannot be examined and evaluated.

Layer 4 filters are PBR attributes *criteriaSourcePort* and *criteriaDestinationPort* and these apply when *criteraProtocol* is set to "TCP" or "UDP".

The following behavior applies for fragmented packets, when a Layer 4 filter is configured in a policy rule:

1. For packets with **FO=0**, the processing is identical to non-fragmented packets.
2. For packets with **FO>0**, the processing will result in a "no-match".

This means that the rule will match only the first fragment, not subsequent fragments. The consequence is that subsequent fragments will be subject to destination based routing (unless matched by a lower priority rule.

Note: In order to ensure that fragmented packets are not unintentionally miss-routed, layer 4 filter criteria should not be used in a PBR rule, if traffic potentially could be fragmented by the network.

Table 3 Fragmented Packets Processing in PBR

L3 match	Packet FO	L4 match	Resulting Action
No	Any	Don't care	No match, skip to next entry.
Yes	= 0	Yes	Match
		No	No match, skip to next entry.
	> 0	Does not evaluate	No match, skip to next entry.

3.10 Dynamic Routing (OSPFv2)

OSPFv2 is classified as an Interior Gateway Protocol (IGP). This means that it distributes routing information between routers belonging to a single Autonomous System (AS).

OSPFv2 is a layer 3 protocol, encapsulated directly in IP datagram with protocol number 89 and it is therefore possible to use it over any stack which implements IP. Hence the ML-PPP ports, as well as VLANs and the Ethernet ports of the SIU can be used for handling OSPF.

Every OSPFv2 enabled router constructs a map of the connectivity to the network, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical path from it to every possible destination in the network. The collection of best paths will then form the node's routing table.

OSPFv2 enabled networks specifies the following terms:

- Autonomous System (AS);
  A group of routers exchanging routing information via a common routing protocol below a single technical administration.

- OSPF Area;
  An OSPF area is a set of networks and hosts within an AS that have been administratively grouped together. As the topology of the area is hidden from the rest of the network, routing traffic within the AS can be significantly reduced. Area number 0 is called Backbone Area (or Transmit Area) and must exist in an OSPF network.

3.10.1 OSPFv2 Network Types

OSPF treats point-to-point interfaces and broadcast interfaces differently (neighbor adjacencies, link state information flooding, protocol packet destination addresses). The SIU supports the following network types:

- Point-to-Point (for example Serial T1 Line);
  In this network type, no Designated Router (DR) or Backup Designated Router (BDR) is elected. It is possible to configure OSPF so that it treats LAN interfaces with only two routers connected to them as point-to-point interfaces. The point-to-point extension modifies the Shortest Path First (SPF) calculation so that the correct next hop is calculated. ML-PPP interfaces must be of type point-to-point.

- Broadcast Multi-access (for example Ethernet);
  In this network type, DR and BDR is elected. A DR generates LSA type 2 updates in the area for the broadcast network. If DR failed then BDR is selected.

3.10.2 OSPFv2 Area Types

The SIU supports the following OSPFv2 area types:

- Stub Area;
  In the stub area, external routes are not flooded into and through a stub area in order to reduce the amount of routing information. Routing to AS-external networks in a stub area is done through a default route.

- Not So Stubby Area (NSSA);
  A not so stubby area can import AS external routes and send them to the backbone, but cannot receive AS-external routes from other areas (including Backbone Area).

- Totally Stub Area;
  In the totally stub area, routes other than intra-area are not accepted. Default route is injected by Area Border Routers (ABR) into the area to reach inter- and external-area destinations.

3.10.3 OSPFv2 Packets

OSPFv2 uses Hello packets to discover and maintain neighbor relationships. The Database Description and Link State Request packets are used to form the adjacencies. The following packet types are used to exchange the link state database:

- Hello packet, used to discover neighbors and build adjacencies between them.

- Data Base Description (DBD) packet, used to exchange LSDB catalog (LSA headers).

- Link State Request (LSR), used to request specific link-state records from another router.

- Link State Update (LSU), used to send requested link-state information.

- Link State Acknowledgement (ACK), used to acknowledge other packet types.

There are several types of Link State Advertisement, depending on their role and origination:

- LSA type 1 (Router-LSA), originated by all routers.
  Describes the collected states of the router's interfaces. Flooded throughout a single area only.

- LSA type 2 (Network-LSA), originated for broadcast networks by the DR.
  Contains the list of routers connected to the network. Flooded throughout a single area only.

- LSA type 3 and 4 (Summary-LSA), originated by ABR.
  Describes a route to a destination outside the area, but inside the AS. LSA type 3 describes routes to networks, and LSA type 4 describes routes to AS Boundary Routers.

- LSA type 5 (AS-external-LSA), originated by AS Boundary Routers.
  Describes a route to a destination in another AS. Default routes for the AS can also be described by AS-external-LSA.

- LSA type 7, generated by an NSSA ASBR.
  LSAs of type 5 are not allowed in NSSA areas, so the NSSA ASBR generates a type 7 LSA instead. LSA type 7 remains within the NSSA and is translated to LSA type 5 by the NSSA ABR.

3.10.4 OSPFv2 Authentication

OSPFv2 supports authentication to secure the communication between the routers. The SIU supports MD5 authentication.

The "OSPFv2 Authentication Failure" alarm is raised when an OSPF packet has been received on a non-virtual interface from a router whose authentication key or type is not defined as an authentication key or type within the receiving interface. The alarm is ceased when a successful authentication is done.

All alarms have their own Operating Instructions (OPI) document, see Reference [12] describing the "OSPFv2Authentication Failure" alarm.

3.10.5 OSPFv2 Route Management

OSPFv2 provides the possibility to control what links are preferred for traffic by setting different costs. Cost can be set for OSPFv2 network.

Additionally, the SIU supports administrative distance management for static, inter-area, intra-area and external OSPFv2 routes. The administrative distance specifies how a router determines the preference of route source.

3.10.6 OSPFv2 BFD

OSPFv2 can use the BFD mechanism (see Section 3.8.2 on page 12) for bi-directional connectivity check of the forwarding plane to next-hop routers. BFD is used in combination with OSPFv2 to achieve short fail-over time.

3.10.7 OSPFv2 Status Monitoring

The SIU allows operators to monitor OSPFv2 protocol status and operations. The CLI command `getOSPF` can be used to print LSDB, border routers, interfaces, neighbors and routes.

See Reference [11] for detailed information about the `getOSPF` command.

4 Engineering Guidelines

This chapter shows a selection of configuration examples using CLI commands.

For further information about MO classes, their attributes/values and counters, see Reference [10].

Examples of command lines are used in the procedures, some of which are longer than the page width of this document and require more than one line. Line break, enter, or carriage return characters must never be used when entering a command line.

4.1 Configuring Ethernet Interfaces

Follow the steps below to configure an Ethernet interface:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **EthernetInterface**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,EthernetInterface=0
   ```

3. Set the value of the physical port to either "Gigabit" for an electrical interface or "SFP" for an optical interface.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,EternetInterface=0 port
   Gigabit
   ```

4. Set the value of the physical port that is used. This value must be unique among all instances of MO **EthernetInterface**.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,EthernetInterface=0
   portNumber 3
   ```

5. If needed, set the attribute *mode* to be used. The value "AUTO" means that the mode will be negotiated and the interface speed agreed will be utilized.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,EthernetInterface=0 mode
   AUTO
   ```

6. Check the validity and consistency of the configuration.

Example:
```
OSmon> checkConsistency trans1
```

7. The **`commit`** command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the **`commit`** command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:
   ```
   OSmon> commit trans1  forcedCommit
   ```

8. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

9. End the transaction.

   Example:
   ```
   OSmon> endTransaction trans1
   ```

4.2 Configuring VLANs

When VLANs are used, one MO **IPInterface** and/or **IPv6Interface** instance is associated with each VLAN. The VLANs are contained in one or more MO **VLANGroup** instances that links the VLANs to the MO **EthernetInterface**. An instance of MO **IPInterface** or **IPv6Interface** cannot be associated with a bridged VLAN.

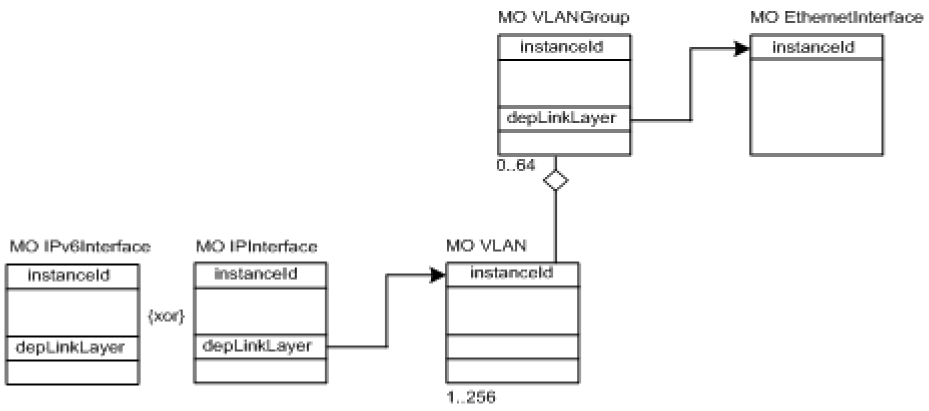

Figure 8 Dependencies for VLANs

Follow the steps below to enable VLANs on an Ethernet interface:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **EthernetInterface** as described in Section 4.1 on page 21.

3. Create one or more instances of MO **VLANGroup**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,VLANGroup=0
   ```

4. Set the reference (use dependency) to the MO **EthernetInterface** created in step 2.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,VLANGroup=0 depLinkLayer
   STN=0,EthernetInterface=0
   ```

5. Create one or more instances of MO **VLAN** below each of the created MO **VLANGroup** instances (created in step 3).

   Example:
   ```
   OSmon> createMO trans1 STN=0,VLANGroup=0,VLAN=0
   ```

6. Set the VLAN tag value to be used on the VLAN. This attribute shall have the value for the VID field of the Ethernet header.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,VLANGroup=0,VLAN=0
   tagValue 2
   ```

   Note: This value cannot be the same on two **VLAN** instances on the same **EthernetInterface**.

7. If needed, set the attribute *tagged* which determines if the egress Ethernet frames will have a VLAN tag or not. The default value is "true" and by setting it to "false" it is possible to allow also non-VLAN-tagged traffic on the same Ethernet interface that has VLANs configured.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,VLANGroup=0,VLAN=0 tagged
   false
   ```

   Note: Only one **VLAN** instance per **EthernetInterface** instance can have this attribute set to "false".

8. If not already created, create an instance of MO **IPInterface** as described in Section 4.6 on page 28 or MO **IPv6Interface** as described in Section 4.7 on page 30 for each VLAN.

9. Set the dependency link layer between the **IPInterface** or **IPv6Interface** and the **VLAN** interface.

   Example A:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0 depLinkLayer
   STN=0,VLANGroup=0,VLAN=0
   ```

Example B:
```
OSmon> setMOAttribute trans1 STN=0,IPv6Interface=0
depLinkLayer STN=0,VLANGroup=0,VLAN=0
```

10. Check the validity and consistency of the configuration.

    Example:
    ```
    OSmon> checkConsistency trans1
    ```

11. The `commit` command is required to activate the configuration.

    Example:
    ```
    OSmon> commit trans1
    ```

    Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

    Example:
    ```
    OSmon> commit trans1  forcedCommit
    ```

12. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

13. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.3 Configuring Ethernet Bridging with IP Host

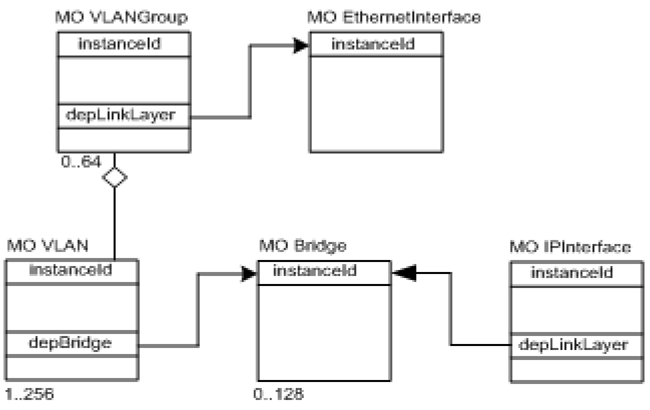

Figure 9   Dependencies for Ethernet Bridging with IP Host

Follow the steps below to configure Ethernet bridging with IP host:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create and configure the needed number of MO **EthernetInterface** instances as described in Section 4.1 on page 21.

3. Create and configure the needed number of MO **VLAN** instances as described in Section 4.2 on page 22 and associate them to the **EthernetInterfaces** created in previous step.

   Note: Different **VLAN** instances must be created for each physical port, even if their VID is identical.

4. Create the needed number of MO **Bridge** instances.

   Example:
   ```
   OSmon> createMO trans1 STN=0,Bridge=0
   ```

5. Connect each **VLAN** to the appropriate **Bridge** instance using the *depBridge* attribute.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,VLANGroup=0,VLAN=0
   depBridge STN=0,Bridge=0
   ```

6. Create and configure an instance of MO **IPInterface** as described in Section 4.6 on page 28 for each Bridge.

7. Set the dependency link layer between **IPInterface** and **Bridge** interface.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0 depLinkLayer
   STN=0,Bridge=0
   ```

8. If QoS is required, create and configure instances of MO **TrafficManager** and MO **QosPolicy** and bind them to VLAN's Ethernet Interfaces, as described in Reference [6].

9. Create and configure IP routes for any destination IP subnets, see Section 4.10 on page 36.

10. Check the validity and consistency of the configuration.

    Example:
    ```
    OSmon> checkConsistency trans1
    ```

11. The `commit` command is required to activate the configuration.

    Example:
    ```
    OSmon> commit trans1
    ```

Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

Example:
```
OSmon> commit trans1 forcedCommit
```

12. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

13. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.4 Configuring DHCP Relay

DHCP relay allows hosts connected to the site access port to get their configuration parameters from a remote DHCP server. Up to eight DHCP server groups with up to four DHCP servers in each can be created.

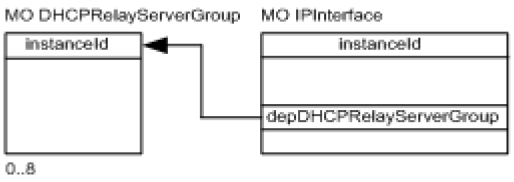

Figure 10 Dependency for DHCP Relay

Follow the steps below to configure DHCP relay on a site interface:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **DHCPRelayServerGroup** with a unique name identifying the group of DHCP servers.

   Example:
   ```
   OSmon> createMO trans1 STN=0,DHCPRelayServerGroup=DhcpSrvGr
   p_1
   ```

3. Add the IP address of a DHCP server to the group.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,DHCPRelayServerGroup=Dhcp
   SrvGrp_1 DHCPServerAddress_1 192.50.20.2
   ```

4. If needed, repeat step 3 with `DHCPServerAddress_2`, `DHCPServerAddress_3`, and `DHCPServerAddress_4` attributes to add additional DHCP servers in this server group.

Note: At least one of the attributes *`DHCPServerAddress_1-4`* must have a valid IP address value and should be mutually exclusive.

5. Activate DHCP relay on the desired site interface by connecting it to a group of DHCP servers.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=WCDMA
   depDHCPRelayServerGroup DhcpSrvGrp_1
   ```

6. Repeat step 5 for each site interface where DHCP relay should be activated.

7. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

8. The **`commit`** command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the **`commit`** command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:
   ```
   OSmon> commit trans1  forcedCommit
   ```

9. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

10. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.5 Configuring DHCP Server

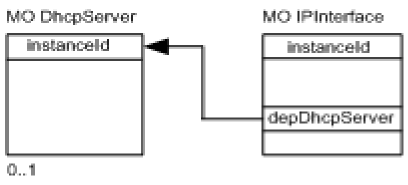

Figure 11 Dependency for DHCP Server

Follow the steps below to configure the DHCP server:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **DhcpServer**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,DhcpServer=0
   ```

3. Set DHCP options.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,DhcpServer=0 option1
   domain-name-servers ip-address 10.10.10.10
   ```

4. If not already created, create an instance of MO **IPInterface** as described in Section 4.6 on page 28.

5. Associate an instance of MO **IPInterface** with the **DhcpServer** instance.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0
   depDhcpServer STN=0,DhcpServer=0
   ```

6. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

7. The `commit` command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:
   ```
   OSmon> commit trans1  forcedCommit
   ```

8. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

9. End the transaction.

   Example:
   ```
   OSmon> endTransaction trans1
   ```

4.6 Configuring IPv4 Interfaces

Each enabled IPv4 interface requires at least one IP subnet and one IP address within this subnet. All IP configuration is static and subnets must be non-overlapping.

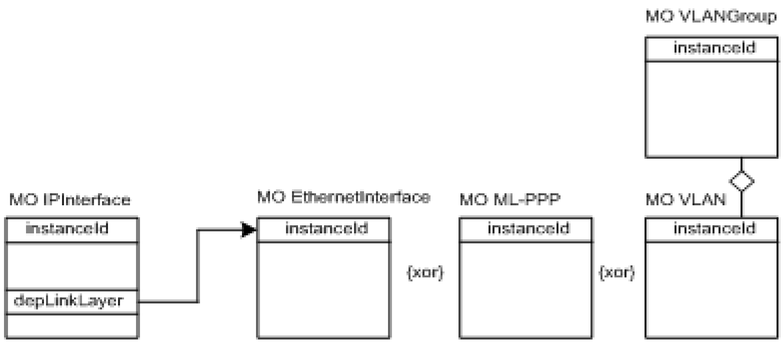

Figure 12 Dependencies for IPv4 Interfaces

Follow the steps below to enable an IPv4 interface:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **IPInterface**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,IPInterface=0
   ```

3. Set the IP address, subnet mask, and default gateway for the primary IP interface according to the IP address plan.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0
   primaryIP_Address 10.10.10.254
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0
   primarySubNetMask 255.255.255.0
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0
   defaultGateway 10.10.10.1
   ```

   Note: The attribute *defaultGateway* is retained for backwards compatibility only. Use an instance of MO **IpRoute** instead. See Section 4.10 on page 36

   Note: If *defaultGateway* is configured, an instance of MO **IpRouteSys** will be created by the system to represent the route via the *defaultGateway*. This route will not be monitored by connectivity check even if the attribute *depCcConfig* is configured.

   Note: The attribute *defaultGateway* can only be defined for **one** instance of MO **IPInterface**. For all other instances it must be set to "0.0.0.0" (not defined).

4. If needed, create an instance of MO **VirtualIPInterface** as described in Section 4.8 on page 33.

Note: This step illustrates the use of various local IP addresses and is not necessary if the primary IP interface is used for all types of traffic, for example synchronization and O&M traffic.

5. Set the dependency link layer between the IP interface and an instance of MO **EthernetInterface** (A), **ML-PPP** (B), or **VLAN** (C).

   Example A:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0
   depLinkLayer STN=0,EthernetInterface=0
   ```

   Example B:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0
   depLinkLayer STN=0,ML-PPP=0
   ```

   Example C:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=0
   depLinkLayer STN=0,VLANGroup=0,VLAN=0
   ```

6. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

7. The `commit` command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:
   ```
   OSmon> commit trans1 forcedCommit
   ```

8. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

9. End the transaction.

   Example:
   ```
   OSmon> endTransaction trans1
   ```

4.7 Configuring IPv6 Interfaces

Enabling IPv6 is done by creating an instance of MO **IPv6Interface**, and adding an MO **IPv6Address** instance to it. For each IPv6 interface it is possible to configure up to seven global IPv6 addresses and each IPv6 interface will also have a system generated link-local address. It is also possible to enable router advertisement and then optionally configure up to seven IPv6 advertisement prefixes.

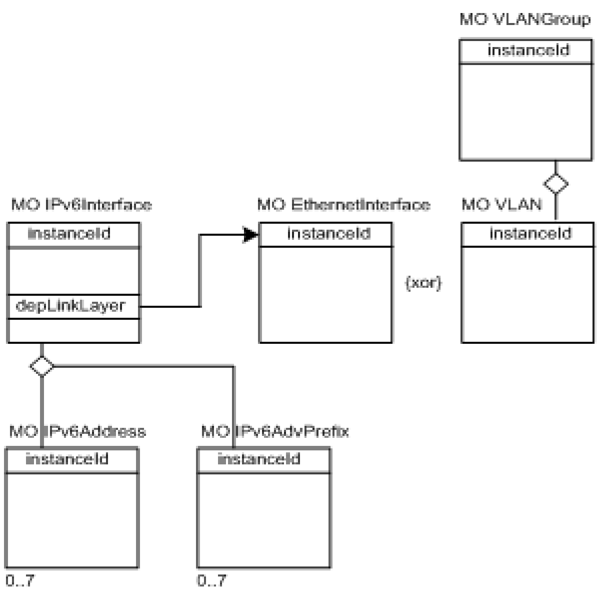

Figure 13 Dependencies for IPv6 Interfaces

Follow the steps below to enable an IPv6 interface:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **IPv6Interface**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,IPv6Interface=0
   ```

3. Create an instance of MO **IPv6Address** on the **IPv6Interface**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,IPv6Interface=0,IPv6Address=0
   ```

4. Set the global IPv6 address and prefix according to the IP address plan.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6Interface=0,IPv6Addre
   ss=0 ipAddress 2001:db8::202:b3ff:0:8329/64
   ```

   Note: Creation of an instance of **IPv6Address** will trigger the system to create an instance of MO **IPv6RouteSys** that represents attributes for one system created IPv6 route in the IPv6 routing table. The route is connected and will route all addresses in the subnet defined by attribute *ipAddress*. In this example, the subnet will automatically be set to 2001:db8::/64 and will be routed to the **IPv6Interface** associated to the current instance of MO **IPv6Address**.

5. If needed, create an instance of MO **IPv6VirtualInterface** as described in Section 4.8 on page 33.

   Note: This step illustrates the use of various local IPv6 addresses and is not necessary if the primary interface is used for all types of traffic.

6. Set the dependency link layer between the IPv6 interface and an instance of MO **EthernetInterface** (A) or **VLAN** (B).

   Example A:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6Interface=0
   depLinkLayer STN=0,EthernetInterface=0
   ```

   Example B:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6Interface=0
   depLinkLayer STN=0,VLANGroup=0,VLAN=0
   ```

7. Follow the steps below to enable router advertisements or continue to Step 11.

8. Create an instance of MO **IPv6AdvPrefix** on the **IPv6Interface**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,IPv6Interface=0,IPv6AdvPrefix=0
   ```

9. Set the IPv6 advertisement prefix.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6Interface=0,IPv6AdvPr
   efix=0 advPrefix 2001:db8::202:b3ff:0:8329/64
   ```

10. Enable router advertisement. Because **IPv6AdvPrefix** instances are contained within the **IPv6Interface**, the router advertisements will include prefix information options for those instances.

    Example:
    ```
    OSmon> setMOAttribute trans1 STN=0,IPv6Interface=0 raSend on
    ```

11. Check the validity and consistency of the configuration.

    Example:
    ```
    OSmon> checkConsistency trans1
    ```

12. The `commit` command is required to activate the configuration.

    Example:
    ```
    OSmon> commit trans1
    ```

Note: If attributes that require a restart are changed, the **`commit`** command is rejected and has to be reissued with the *forcedCommit* parameter.

Example:
```
OSmon> commit trans1 forcedCommit
```

13. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

14. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.8 Configuring Virtual IPv4 Interfaces

In order to be able to use resilient static routes, it is recommended that internal applications (O&M, Packet Relay, Synchronization, and CES PWE) use virtual IPv4 interfaces, with IP addresses independent from any physical interface.

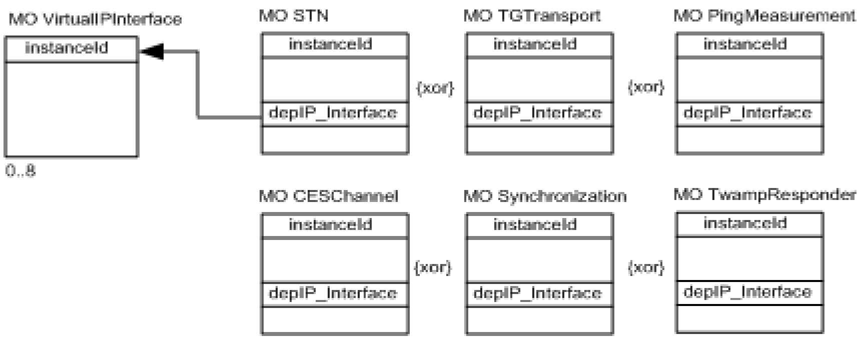

Figure 14 Dependencies for Virtual IPv4 Interface

Follow the steps below to create and configure a virtual IPv4 interface.

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **VirtualIPInterface**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,VirtualIPInterface=0
   ```

3. Set the IP address of the virtual IP interface.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,VirtualIPInterface=0
   IP_Address <IP address>
   ```

4. Set the dependency (attribute *depIP_Interface*) from an internal application that should use this interface. This should be an instance of MO **STN**, **TGTransport**, **PingMeasurement**, **Synchronization**, **CESChannel**, or **TwampResponder**.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,Synchronization=0
   depIP_Interface STN=0,VirtualIPInterface=0
   ```

5. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

6. The `commit` command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:
   ```
   OSmon> commit trans1  forcedCommit
   ```

7. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

8. End the transaction.

   Example:
   ```
   OSmon> endTransaction trans1
   ```

4.9 Configuring Virtual IPv6 Interfaces

In order to be able to use resilient static routes with IP addresses independent from any physical interface, virtual IPv6 interfaces can be used.

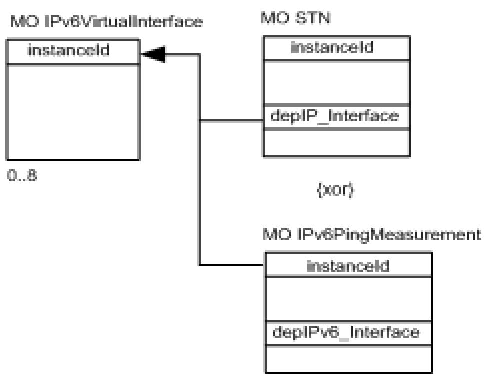

Figure 15   Dependencies for Virtual IPv6 Interfaces

Follow the steps below to create and configure a virtual IPv6 interface.

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **IPv6VirtualInterface**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,IPv6VirtualInterface=0
   ```

3. Set the IPv6 address prefix of the virtual IPv6 interface.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6VirtualInterface=0
   address <IPv6AddressPrefix>
   ```

4. Set the dependency from an internal application that should use this interface. This should be an instance of MO **STN** or **IPv6PingMeasurement**.

   Example A:
   ```
   OSmon> setMOAttribute trans1 STN=0 depIP_Interface
   STN=0,IPv6VirtualInterface=0
   ```

   Example B:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6PingMeasurement=0
   depIPv6_Interface STN=0,IPv6VirtualInterface=0
   ```

5. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

6. The `commit` command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:
   ```
   OSmon> commit trans1  forcedCommit
   ```

7. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

8. End the transaction.

   Example:

```
OSmon> endTransaction trans1
```

4.10 Configuring Static IPv4 Routes

Up to 64 static IPv4 routes can be configured (in addition to the automatically generated routes for attached subnets). Each distinct external IP subnet that must be reachable must have an instance of MO **IpRoute** configured (unless a default gateway is defined).

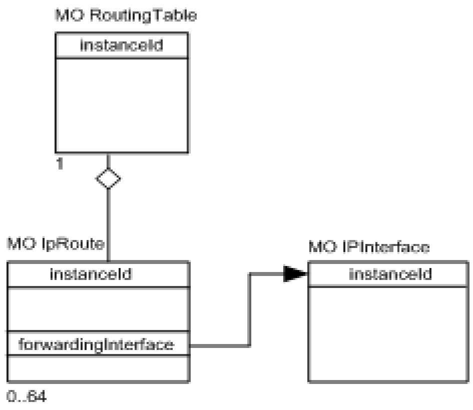

Figure 16 Dependencies for Static IPv4 Routes

Follow the steps below to configure a static IPv4 route.

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create a static IPv4 route in the IPv4 routing table by creating an instance of MO **IpRoute**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,RoutingTable=0,IpRoute=route1
   ```

3. Set the attribute *destIpSubnet* that shall have the value of the destination IP subnet in the form <net number>/<size of the net mask> .

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=ro
   ute1 destIpSubnet 192.168.5.0/24
   ```

   Note: If instances of MO **IpRoute** or **IpRouteSys** have overlapping values of *destIpSubnet*, the longest prefix matching has priority. If two MO **IpRoute** or **IpRouteSys** have identical values of *destIpSubnet* and the same value of *admDistance*, only one will be used by the system.

4. Set the attribute *nextHopIpAddress* that shall have the value of the IP address of the next-hop router to reach the destination subnet.

Example:
```
OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=ro
ute1 nextHopIpAddress 192.168.5.2
```

Note: The attribute *nextHopIpAddress* must be within exactly one of the subnets defined by an instance of MO **IpRouteSys** and cannot be an IP address belonging to the SIU.

5. Set the attribute *forwardingInterface* which is the egress interface for traffic forwarded according to this route. The attribute currently only has an informational purpose.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=ro
   ute1 forwardingInterface STN=0,IPInterface=0
   ```

   Note: This attribute will be updated to show the interface that *nextHopIpAddress* is connected to. If *nextHopIpAddress* is not within one of the subnets defined by an instance of MO **IpRouteSys**, it will be set to <empty string>.

6. Set the attribute *admDistance* which is the relative distance to the destination when using this route. This value is used to choose between alternative paths to the same destination, that is the route with the lowest value for active interfaces will be used.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=ro
   ute1 admDistance 1
   ```

7. The attribute *disableConnectivityCheck* controls whether or not the route should be included in BFD connectivity check monitoring. If the route should be supervised with BFD connectivity check monitoring, follow the instructions in Section 4.14 on page 45. If only port availability status monitoring is required for the route, then this attribute should be set to "true".

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=ro
   ute1 disableConnectivityCheck true
   ```

8. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

9. The `commit` command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

Example:
```
OSmon> commit trans1 forcedCommit
```

10. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

11. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.11 Configuring Static IPv6 Routes

Up to 64 static IPv6 routes can be configured (in addition to the automatically generated routes for attached subnets).

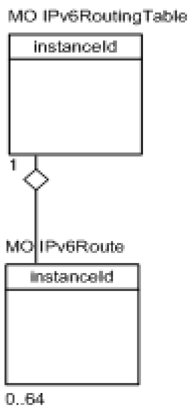

Figure 17 Dependency for Static IPv6 Routes

Follow the steps below to configure a static IPv6 route.

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create a static IPv6 route in the IPv6 routing table by creating an instance of MO **Ipv6Route**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,IPv6RoutingTable=0,IPv6Route=
   route1
   ```

3. Set the attribute *destIpSubnet* that shall have the value of the destination IPv6 subnet prefix in the form <net number>/<size of the net mask>.

Example:
```
OSmon> setMOAttribute trans1 STN=0,IPv6RoutingTable=0,IPv6Ro
ute=route1 destIpSubnet 2001:DB8::8:800:200C:4100/120
```

Note: If instances of MO **IPv6Route** or **IPv6RouteSys** have overlapping values of *destIpSubnet*, the longest prefix matching has priority. If one MO **IPv6Route** has the same value of *destIpSubnet* and *metric* as another MO **IPv6Route** or **IPv6RouteSys**, only one of the routes will be used by the system. Multiple instances of **IPv6RouteSys** are not allowed to overlap with each other. **IPv6Route** instances can have overlap, also with instances of **IPv6RouteSys**.

4. Set the attribute *nextHopIpAddress* that shall have the value of the IPv6 address of the next-hop router to reach the destination subnet.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6RoutingTable=0,IPv6Ro
   ute=route1 nextHopIpAddress 2001:DB8::8:800:200C:0001
   ```

   Note: The attribute *nextHopIpAddress* must lie within exactly one of the subnets defined by an instance of MO **IPv6RouteSys**. Otherwise this MO instance will have attribute *active* = "false". The *nextHopIpAddress* cannot be an IPv6 address belonging to the SIU.

5. Set the attribute *metric* for this route. This value is used to choose between alternative paths to the same destination, that is the route with the lowest value for active interfaces will be used.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6RoutingTable=0,IPv
   6Route=route1 metric 1
   ```

6. The attribute *disableConnectivityCheck* controls whether or not the route should be included in BFD connectivity check monitoring. If the route should be supervised with BFD connectivity check monitoring, follow the instructions in Section 4.15 on page 48. If only port availability status monitoring is required for the route, then this attribute should be set to "true".

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6RoutingTable=0,IPv6Ro
   ute=route1 disableConnectivityCheck true
   ```

7. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

8. The `commit` command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

Example:
```
OSmon> commit trans1 forcedCommit
```

9. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

10. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.12 Configuring Policy Based Routing

Follow the steps below to configure a routing policy with two rules and attach it to an IPv4 interface. This example will filter UDP traffic and make routing decision based on source port. See also Section 3.9.1 on page 15 for information about handling fragmented packets in PBR.

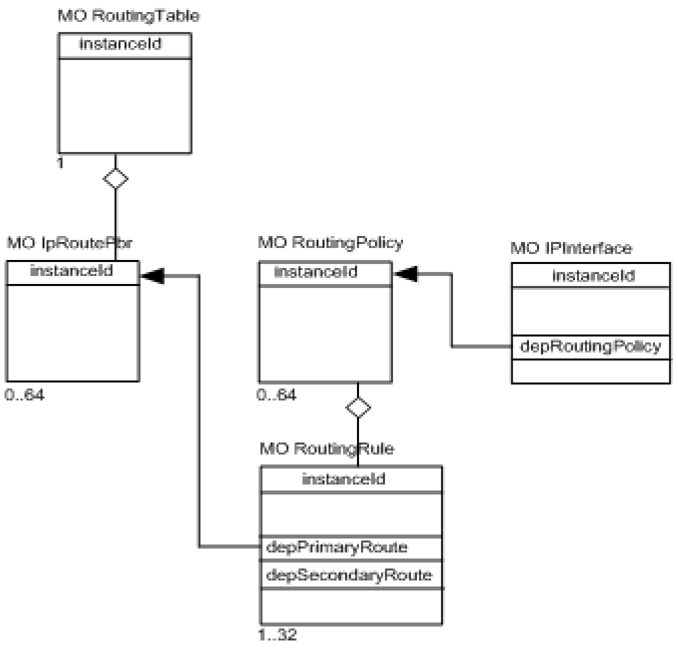

Figure 18 Dependencies for Policy Based Routing

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create two instances of MO **IpRoutePbr**.

   Example:

```
OSmon> createMO trans1 STN=0,RoutingTable=0,IpRoutePbr=pbrou
te1
OSmon> createMO trans1 STN=0,RoutingTable=0,IpRoutePbr=pbrou
te2
```

3. Set the attribute *nextHopIpAddress* for each **IpRoutePbr** instance that shall have the value of the IP address of the next hop router to reach the destination subnet.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRouteP
   br=pbroute1 nextHopIpAddress 10.10.10.2
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRouteP
   br=pbroute2 nextHopIpAddress 10.10.10.3
   ```

   Note: The attribute *nextHopIpAddress* must be within exactly one of the subnets defined by the attributes *primaryIP_Address* and *primarySubNetMask* of an MO **IPInterface**.

4. The attribute *disableConnectivityCheck* controls whether or not the route should be included in BFD connectivity check monitoring. If the route should be supervised with BFD connectivity check monitoring, follow the instructions in Section 4.14 on page 45. If only port availability status monitoring is required for the route, then this attribute should be set to "true".

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=pb
   route1 disableConnectivityCheck true
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=pb
   route2 disableConnectivityCheck true
   ```

5. Create an instance of MO **RoutingPolicy**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,RoutingPolicy=0
   ```

6. Create two instances of MO **RoutingRule**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,RoutingPolicy=0,RoutingRule=0
   OSmon> createMO trans1 STN=0,RoutingPolicy=0,RoutingRule=1
   ```

7. Set the priority for each **RoutingRule** instance. It must be unique among the **RoutingRule** instances in a given **RoutingPolicy**.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingR
   ule=0 priority 10
   OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingR
   ule=1 priority 20
   ```

Tip: Make large gaps between priority numbers so that new entries can easily be defined and inserted wherever required in an existing **RoutingPolicy**.

8. Set the *depPrimaryRoute* attribute for each **RoutingRule** instance to reference an instance of MO **IpRoutePbr** that holds the primary route to use if rule criteria match.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingRu
   le=0 depPrimaryRoute STN=0,RoutingTable=0,IpRoutePbr=pbroute1
   OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingRu
   le=1 depPrimaryRoute STN=0,RoutingTable=0,IpRoutePbr=pbroute2
   ```

9. Set the protocol selector for each **RoutingRule** instance specifying the upper protocol type. For protocol numbers, see Reference [19].

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingRu
   le=0 criteriaProtocol 17
   OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingRu
   le=1 criteriaProtocol 17
   ```

10. Set the source port number for each **RoutingRule** instance.

    Example:
    ```
    OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingRu
    le=0 criteriaSourcePort 4711
    OSmon> setMOAttribute trans1 STN=0,RoutingPolicy=0,RoutingRu
    le=1 criteriaSourcePort 4712
    ```

11. Associate an instance of MO **IPInterface** with the **RoutingPolicy** instance.

    Example:
    ```
    OSmon> setMOAttribute trans1 STN=0,IPInterface=0
    depRoutingPolicy STN=0,RoutingPolicy=0
    ```

    If the IP interface has to be created, see Section 4.6 on page 28.

12. Check the validity and consistency of the configuration.

    Example:
    ```
    OSmon> checkConsistency trans1
    ```

13. The `commit` command is required to activate the configuration.

    Example:
    ```
    OSmon> commit trans1
    ```

Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

Example:
```
OSmon> commit trans1 forcedCommit
```

14. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

15. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.13 Configuring OSPFv2

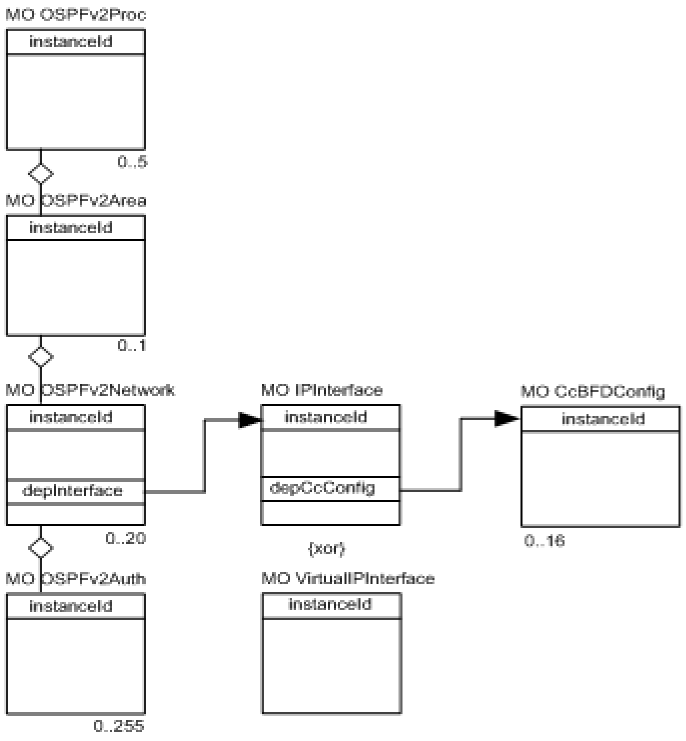

Figure 19 Dependencies for OSPFv2

Follow the steps below to configure OSPFv2 with authentication on an IPv4 interface.

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **OSPFv2Proc**.

Example:

```
OSmon> createMO trans1 STN=0,OSPFv2Proc=0
```

3. Create an instance of MO **OSPFv2Area**.

   Example:

   ```
   OSmon> createMO trans1 STN=0,OSPFv2Proc=0,OSPFv2Area=0
   ```

4. Create an instance of MO **OSPFv2Network**.

   Example:

   ```
   OSmon> createMO trans1 STN=0,OSPFv2Proc=0,OSPFv2Area=0,OSPFv2
   Network=0
   ```

5. The attribute *authentication* controls if MD authentication should be enabled for the network (default value is "none").

   Example:

   ```
   OSmon> setMOAttribute trans1 STN=0,OSPFv2Proc=0,OSPFv2Area=0
   ,OSPFv2Network=0 authentication md5
   ```

   Note: When this attribute is set to "md5" at least one instance of MO **OSPFv2Auth** must exist with one or more key-ids and passwords configured. See step 6 and 7.

6. Create an instance of MO **OSPFv2Auth**.

   Example:

   ```
   OSmon> createMO trans1 STN=0,OSPFv2Proc=0,OSPFv2Area=0,OSPFv2
   Network=0,OSPFv2Auth=0
   ```

   Note: One or more keys (password) and key-ids are configured on each OSPF network. The system begins a rollover process until all the neighbors have adopted the new password. This allows neighboring routers to continue communication while the network administrator is updating them with a new password. The router will stop sending duplicate packets once it detects that all of its neighbors have adopted the new password.

7. Set the attribute *keyId* and *password* of the MO **OSPFv2Auth**.

   Example:

   ```
   OSmon> setMOAttribute trans1 STN=0,OSPFv2Proc=0,OSPFv2Area
   =0,OSPFv2Network=0,OSPFv2Auth=0 keyId 1

   OSmon> setMOAttribute trans1 STN=0,OSPFv2Proc=0,OSPFv2Area=0
   ,OSPFv2Network=0,OSPFv2Auth=0 password pass
   ```

8. The attribute *enableOspfBfd* controls whether or not BFD should be enabled on the interface to monitor OSPF neighbor states (default value is "false").

   Example:

```
OSmon> setMOAttribute trans1 STN=0,OSPFv2Proc=0,OSPFv2Area=0
,OSPFv2Network=0 enableOspfBfd true
```

Note: If this attribute has the value "true", BFD configuration will be provided by the attribute *depCcConfig* on the MO **IPInterface** referenced to by attribute *depInterface*. See Section 4.14 on page 45.
If this attribute has the value "false", OSPF neighbors will not be monitored by BFD, even when attribute *depCcConfig* is enabled on the MO **IPInterface** referred to by attribute *depInterface*.

9. Set the dependency to the IP interface on which OSPFv2 shall be configured. This should be an instance of MO **IPInterface** or **VirtualIPInterface**.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,OSPFv2Proc=0,OSPFv2Area=0
   ,OSPFv2Network=0 depInterface STN=0,IPInterface=0
   ```

10. Check the validity and consistency of the configuration.

    Example:
    ```
    OSmon> checkConsistency trans1
    ```

11. The `commit` command is required to activate the configuration.

    Example:
    ```
    OSmon> commit trans1
    ```

    Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

    Example:
    ```
    OSmon> commit trans1  forcedCommit
    ```

12. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

13. End the transaction.

    Example:
    ```
    OSmon> endTransaction trans1
    ```

4.14 Configuring IPv4 BFD Connectivity Check

To successfully create and configure BFD connectivity check for a route, a working **IPInterface** needs to exist or be created in the same transaction. A similar configuration is also required on the route destination.

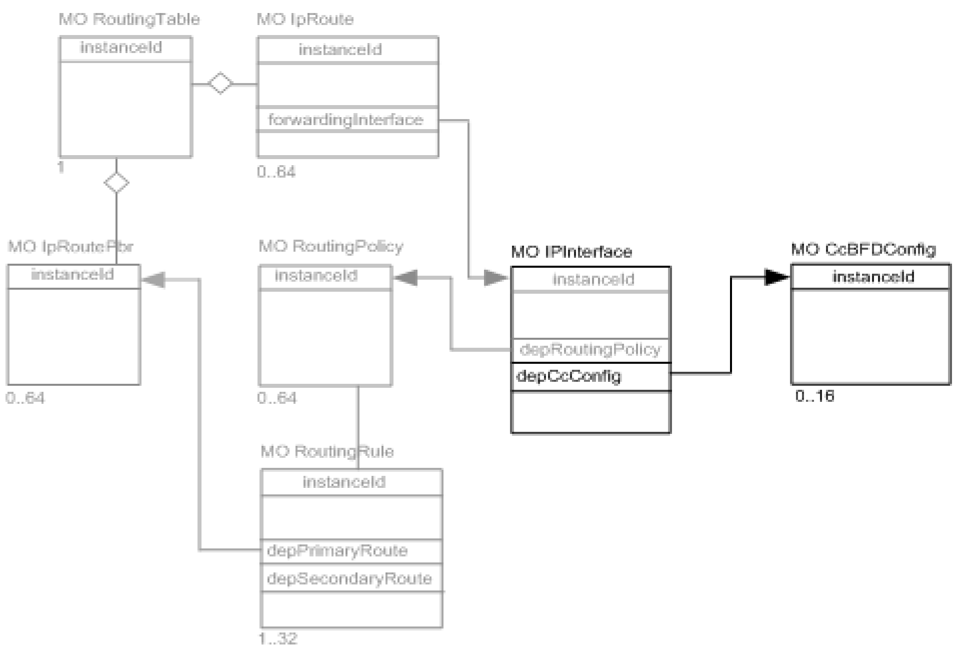

Figure 20 Dependencies for IPv4 BFD Connectivity Check

Follow the steps below to create and configure BFD connectivity check for a route:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **IpRoute** as described in Section 4.10 on page 36 or **IpRoutePbr** as described in Section 4.12 on page 40.

3. Create an instance of MO **CcBFDConfig**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,CcBFDConfig=0
   ```

4. To enable connectivity check, set the reference from the MO **IPInterface** to the connectivity check configuration (an instance of MO **CcBFDConfig**).

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPInterface=WAN
   depCcConfig STN=0,CcBFDConfig=0
   ```

   Note: When the *depCcConfig* attribute refers to an instance of MO **CcBFDConfig** (not equal to <empty string>), connectivity check is enabled and will monitor all instances of MO **IpRoute** or **IpRoutePbr** that refer to this MO **IPInterface** instance. Each individual MO **IpRoute** or **IpRoutePbr** instance can disable connectivity check monitoring by using the *disableConnecitivyCheck* attribute. See the next step.

5. If there are any routes (instances of MO **IpRoute** or **IpRoutePbr**) on the referenced **IPInterface** that shall not be supervised with connectivity check monitoring, this has to be disabled for these routes. This can be used in the case when the next hop for the route does not support BFD.

   Example:

   ```
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoute=2
   disableConnectivityCheck true
   OSmon> setMOAttribute trans1 STN=0,RoutingTable=0,IpRoutePbr
   =pbroute1 disableConnectivityCheck true
   ```

   Note: If this attribute has the value "true", this route will not be monitored by connectivity check, even if connectivity check is enabled on the *forwardingInterface*. If this attribute has the value "false", the route will be monitored by connectivity check, provided that the *depCcConfig* attribute on the MO **IPInterface** instance referred to by attribute *forwardingInterface* refers to a connectivity check configuration.

   Note: If the route refers to an MO **IPInterface** instance that has the *depLinkLayer* attribute referencing a MO **Bridge** instance, the value of *disableConnectivityCheck* must be "true".

6. Check the validity and consistency of the configuration.

   Example:

   ```
   OSmon> checkConsistency trans1
   ```

7. The `commit` command is required to activate the configuration.

   Example:

   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:

   ```
   OSmon> commit trans1  forcedCommit
   ```

8. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

9. End the transaction.

   Example:

   ```
   OSmon> endTransaction trans1
   ```

4.15 Configuring IPv6 BFD Connectivity Check

To successfully create and configure BFD connectivity check for an IPv6 route, a working **IPv6Interface** needs to exist or be created in the same transaction. A similar configuration is also required on the route destination.

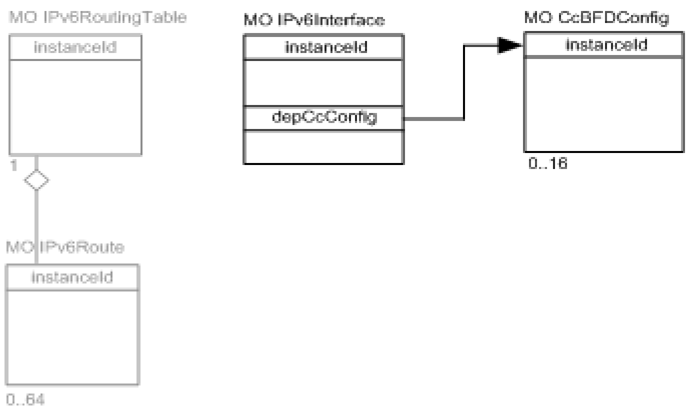

Figure 21 Dependencies for IPv6 BFD Connectivity Route

Follow the steps below to create and configure IPv6 BFD connectivity check for an IPv6 route:

1. Start a basic CM transaction.

   Example:
   ```
   OSmon> startTransaction trans1
   ```

2. Create an instance of MO **IPv6Route** as described in Section 4.11 on page 38.

3. Create an instance of MO **CcBFDConfig**.

   Example:
   ```
   OSmon> createMO trans1 STN=0,CcBFDConfig=0
   ```

4. To enable connectivity check, set the reference from the MO **IPv6Interface** to the connectivity check configuration (an instance of MO **CcBFDConfig**).

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6Interface=0
   depCcConfig STN=0,CcBFDConfig=0
   ```

   Note: When the *depCcConfig* attribute of an **IPv6Interface** instance refers to an instance of MO **CcBFDConfig** (not equal to <empty string>), connectivity check is enabled and will monitor all instances of MO **IPv6Route** that has a *nextHopIpAddress* which will be routed through this **IPv6Interface**. Each individual MO **IPv6Route** instance can disable connectivity check monitoring by using the *disableConnecitivyCheck* attribute. See the next step.

5. Connectivity check monitoring can be disabled individually on any route (instance of **IPv6Route**), even if the *nextHopIpAddress* will be routed through an **IPv6Interface** where the *depCcConfig* attribute is set (BFD enabled). This can be used in the case when the next hop for the route does not support BFD.

   Example:
   ```
   OSmon> setMOAttribute trans1 STN=0,IPv6RoutingTable=0,IPv6Ro
   ute=route1 disableConnectivityCheck true
   ```

   Note: If this attribute has the value "true", this route will not be monitored by connectivity check, even if connectivity check is enabled on the interface. If this attribute has the value "false", the route will be monitored by connectivity check, provided that the *depCcConfig* attribute on the MO **IPv6Interface** instance that is used by this route, refers to a connectivity check configuration.

6. Check the validity and consistency of the configuration.

   Example:
   ```
   OSmon> checkConsistency trans1
   ```

7. The `commit` command is required to activate the configuration.

   Example:
   ```
   OSmon> commit trans1
   ```

   Note: If attributes that require a restart are changed, the `commit` command is rejected and has to be reissued with the *forcedCommit* parameter.

   Example:
   ```
   OSmon> commit trans1  forcedCommit
   ```

8. If the unit is restarted, this takes a few minutes and all traffic is closed down during this time. In this case, reconnect and log on again.

9. End the transaction.

   Example:
   ```
   OSmon> endTransaction trans1
   ```

5 Concepts

Bridge	Defined by IEEE 802.1Q, a L2 circuit used for transporting VLAN tagged or untagged ingress Ethernet traffic to a VLAN tagged or untagged egress port. In the context of this User Guide, the Bridge defines a method for defining a L2 circuit by connecting several VLANs from different ports.
Bridged VLAN	Bridged VLAN is the concept of simultaneously bridging multiple VLANs together. This is occasionally needed in order to bridge non-routable protocols or unsupported routed protocols between multiple VLANs.
Duplicate Address Detection (DAD)	How a node determines whether or not an address it wishes to use is already in use by another node.
Bridged Virtual Interface (BVI)	In the context of this document, this is understood as the possibility to define an IP host function on an L2 Bridge.
Destination based forwarding	The forwarding decision for a packet is based on its destination address. The destination address is used to look up an entry in a routing table.
IPoE	Internet Protocol over Ethernet.
Match criteria	A criteria for matching a packet filter, for example a certain source IPv4 address.
Policy	A set of rules about how to handle a certain subject.
Resilience	The ability to provide and maintain an acceptable level of service in the fact of faults and challenges to normal operation.
Routed VLAN	As defined in IEEE standard 802.1Q, virtual LANs offer a method of dividing one physical network into multiple broadcast domains. In enterprise networks, these broadcast domains usually match with IP subnet boundaries, so that each subnet has its own VLAN. However, VLAN-enabled switches cannot, by themselves, forward traffic across VLAN boundaries. For inter-VLAN communication, a Layer 3 router is required. The addition of a router makes it possible to send traffic between VLAN boundaries. The router uses

	IP subnets to move traffic between VLANs. Each VLAN has a different IP subnet and there is a one-to-one correspondence of VLAN and IP subnet boundaries. If a host is in a given IP subnet, it is also in a given VLAN, and vice versa.
Router Advertisement	Routers advertise their presence together with various link and Internet parameters either periodically, or in response to a Router Solicitation message. Router Advertisements contain prefixes that are used for determining whether another address shares the same link (on-link determination) and/or address configuration, a suggested hop limit value, etc.
Router Discovery	How hosts locate routers that reside on an attached link.
Router Solicitation	When an interface becomes enabled, hosts may send out Router Solicitations that request routers to generate Router Advertisements immediately rather than at their next scheduled time.
Routing policy	A table of routes, with IP addresses or subnets as key and for each key specifying output (egress) interface and/or the next hop address.
Rule	A rule consists (in this context) of a filter part with one or more matching criteria and an action part. A rule in a routing policy selects a routing table or defines a route.
Site port	An Ethernet or ML-PPP interface used for connecting hosts at the RBS site.
Source based forwarding	The forwarding decision for a packet is based on its source address.
Virtual IP Interface	A virtual IP interface is an IP interface that is not associated with a physical interface. It is also known as loopback interface.
WAN port	An Ethernet or ML-PPP interface used for connecting to the backhaul network.

Glossary

See Reference [2]

Reference List

Document numbers that ends with an asterisk have a release specific suffix.

Ericsson Documents

[1] *Library Changes*

[2] *Glossary*

[3] *SIU 02 Description*

[4] *User Guide, Security*

[5] *User Guide, Synchronization*

[6] *User Guide, Transport Sharing and QoS*

[7] *User Guide, IP over E1/T1*

[8] *User Guide, CESoPSN*

[9] *User Guide, Ethernet OAM and TWAMP*

[10] *Managed Object Model*

[11] *Command Descriptions*

[12] *OSPFv2 Authentication Failure*

[13] *User Description, Abis over IP*, 263/1553-HSC 103 12/*

[14] *Packet Abis Dimensioning Guideline*, 220/100 56-HSC 103 12/*

Standards

[15] *IPv6 Stateless Address Autoconfiguration*, RFC 4862

[16] *Neighbor Discovery for IP version 6 (IPv6)*, RFC 4861

[17] *OSPF Version 2*, RFC 2328

[18] *Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)*, RFC 5881

[19] *http://www.iana.org/assignments/protocol-numbers*, Assigned Internet Protocol Numbers