IPv6 Addressing

Teldat-Dm 805-I

Legal Notice

Warranty

This publication is subject to change.

Teldat offers no warranty whatsoever for information contained in this manual.

Teldat is not liable for any direct, indirect, collateral, consequential or any other damage connected to the delivery, supply or use of this manual.

Table of Contents

I Related Documents

Teldat-Dm 754-I NSLA

Teldat-Dm 806-I DHCPv6 Protocol

Teldat-Dm 807-I Static Routing IPv6

Teldat-Dm 808-I IPv6 Access Control

Teldat-Dm 810-I IPv6 Tunnel over IPv4

Teldat-Dm 814-I RIPng Protocol

Teldat-Dm 815-I IPv6 Multicast

Teldat-Dm 816-I OSPFv3 Protocol

Chapter 1 Introduction

1.1 IPv6 Protocol: Introduction

IPv6 is the most recent version of the Internet Protocol (IP). It was first described in RFC 2460, Internet Protocol, Version 6 (IPv6) Specification. IPv6 was proposed when it became clear that the 32-bit addressing scheme of IPv4 was inadequate to meet the demands of Internet growth. It is based on the Internet Protocol but with a greatly expanded address space (from 32 bits to 128 bits). Unnecessary fields have also been eliminated in order to simplify the header and thus reduce processing time. Although the address size is 128 bits, the IPv6 header size is only 40 bytes. All additional fields are organized in the *extension headers*.

IPv6 architecture can coexist with that of IPv4 users, thus easing migration to IPv6 while providing improvements to security, quality of service and globally unique IP addresses. IPv6 offers greater address space and allows networks to scale and provide global reachability.

The flexibility of IPv6 addresses reduces the need for private addresses and the use of *Network Address Translation* (NAT). In addition, IPv6 enables new application protocols that do not require special processing by border routers sitting at the edge of the network.

1.1.1 IPv6 Addresses: Format

As explained in the previous section, full IPv6 addresses are represented as 32 hexadecimal digits (although there is a simplified nomenclature that greatly reduces their length, as explained below). These digits are grouped in 8 groups of 4 digits to improve legibility, with groups being separated by colons (:). The following examples show different possible notations for an IPv6 address:

2001:0db8:fedc:ba98:7654:3210:fedc:ba98

2001:0db8:0000:0000:0000:0000:1395:a980

As you can see, due to the length set by the new format, the DNS protocol assumes an important role. Furthermore, some situations allow for an abbreviated format, greatly reducing the length of the addresses. The rules defining the abbreviated format are:

(1) Leading zeros (on the left) in each group can be omitted. A group made up entirely of zeros can be replaced by a single zero.

(2) A sequence of two or more groups consisting of only zeros can be replaced with a colon (:). To avoid ambiguity, this abbreviation can only be used once in each IPv6 address.

Many of the addresses, some of which (like the *loopback address*) are used frequently, can be greatly simplified thanks to these rules. For instance:

2001:0db8:0000:8001:0000:0000:0023:1005 (extended format)

2001:db8:0:8001:0:0:23:1005 (rule 1)

2001:db8:0:8001::23:1005 (rule 2)

The representation of prefixes in IPv6 (address/prefix-length) can be used to represent complete blocks of the address space. It is similar to the representation of the mask length in IPv4 written in *Classless Inter-Domain Routing* (CIDR) notation. According to RFC 4291, prefixes must be represented using eight groups of 4 hexadecimal digits separated, like the addresses, by a colon (:). The prefix length is a decimal value that specifies how many of the leftmost contiguous bits make up the prefix. For example:

2001:db8:8001::/48

Not all addresses are handled in the same way. IPv6 supports three different types of addresses (*unicast*, *anycast* and *multicast*) and the treatment of each type varies substantially.

1.1.1.1 IPv6 Address Type: Unicast

This is the most common type of address used in IPv6. You could call them individual address types because they identify a single recipient. These addresses must be unique on a link to pass the *Duplicated Address Detection* (DAD) procedure (explained later on in this manual). The format, defined by the *RFC 3587* standard, is as follows:
Unicast address format.

n bits	m bits	128 – n – m bits
Global routing prefix	Subnet ID	Interface ID

- **Global routing prefix**. This is the network address in *IPv4* language. This prefix uniquely identifies the network connected to the Internet.
- **Subnet ID**. This is the subnet identifier. This allows you to create multiple networks within the destination network. Thus, you can identify each subnet individually using the *subnet ID*.
- **Interface ID**. This is the network interface identifier. Interface identifiers are unique within a subnet, meaning there may be other devices with the same *interface ID* in other subnets.

For most scenarios where the *global routing prefix* and the *subnet ID* lengths are 48 and 16 bits, respectively, the *2001:db8:0:0:1234:56ff:fe78:9abc* address would be broken down in the following way: 2001:db8:0 (*Global routing prefix*), 0 *(Subnet ID)* and 1234:56ff:fe78:9abc *(Interface ID)*.

Not all unicast addresses are global. There are a number of addresses that are bound to an interface, such as the *loopback* address, or to the link, such as the *link-local* address. These addresses and the unspecified address are detailed in the following subsections.

1.1.1.1.1 Unspecified Address

An unspecified address is only used to indicate the absence of an address. The address is defined as follows:

:: (or in its extended version 0:0:0:0:0:0:0:0)

The use of an unspecified address is justified in certain cases, such as when it is used as a source address in any packet sent by an initializing host before it has learned its own address.

The unspecified address is never used as a destination address in IPv6 packets or routing headers. Furthermore, routers cannot forward any packets that have an unspecified address as source address.

1.1.1.1.2 Loopback Address

A unicast address is also called a loopback address:

::1 (or in other words 0:0:0:0:0:0:0:1)

The loopback address should not be assigned to any physical network interface nor be used as a source address for packets sent outside of a node. In addition, a packet addressed to the *loopback*address cannot be sent outside of a node to be routed by a router at a later time. Similarly, a packet received by an interface that contains the *loopback* address as its destination address is rejected.

1.1.1.1.3 Link-local Address

This address is only valid for communications on the link associated with the interface. It can be automatically configured and starts with the following prefix:

fe80::/10

A router never retransmits a packet with a link-local address as destination.

1.1.1.2 IPv6 Address Type: Anycast

An IPv6 anycast address is an address assigned to more than one interface (usually belonging to different nodes). When an anycast packet is sent to an anycast address, it is delivered to the nearest interface with this address (in accordance with routing protocols).

Anycast addresses occupy the same address space as unicast addresses, using any of the defined unicast address formats. In addition, anycast addresses are syntactically identical to unicast addresses. When a unicast address is assigned to more than one interface it becomes an anycast address and must be specified as such.

In anycast addresses, the longest prefix identifies the topological region to which all interfaces with this address belong.

Anycast addresses are only used by routers and not hosts. An anycast address must not be used as the source of an IPv6 packet.

1.1.1.3 IPv6 Address Type: Multicast

A *multicast* address is simply an identifier for a group of interfaces (typically belonging to different nodes). Unlike *anycast* addresses, *multicast* addresses have their own dedicated address space and are easily identified by the prefix ff00::/8. Their format is as follows:

Multicast address format.

8 bits	4 bits	4 bits	112 bits
ff	Flags	Scope	Group ID

- The **flags** help distinguish different types of addresses.
- The **scope** field allows you to easily control the scope of the transmission. Permitted levels are: node (1), link (2), place (5), organization (8) or global (E).
- The **group ID** field stores the multicast group identifier. This identifier differentiates a group of interfaces that normally belong to other nodes. A packet sent to a *multicast* address is delivered to all interfaces identified by the *multicast* address.

For example, *group ID* 101 has been assigned to all NTP servers. Variations in field values produce different behaviors.

- **ff01::101** . All NTP servers on the same interface.
- **ff02::101** . All NTP servers on the same link.
- **ff0e::101** . All NTP servers on the Internet.

Some *multicast* addresses are reserved for certain situations. RFC 3513 defines some addresses to replace the old IPv4 broadcast addresses. These addresses allow packets to be sent to all nodes or routers in a given field.

- **ff01::1** . All nodes on the same interface.
- **ff02::1** . All-nodes link-local multicast group.
- **ff01::2** . All routers on the same interface.
- **ff02::2** . All-routers link-local multicast group.
- **ff05::2** . All routers in the same site.
- **ff02::1:ff00:0/104** . Solicited-node multicast group. This is used as a destination address in Neighbor Solicitation (NS).

Any interface connected to the Internet with a *unicast* address automatically configures a *link-local* address that activates IPv6 on that interface. In addition, the configured interface automatically joins the following multicast groups (all belonging to the same link): *solicited-node multicast group* (ff02::1:2ff00:0*), all-nodes link-local multicast group* (ff02::1) and *all-routers link-local multicast group* (ff02::2).

1.1.2 Simplifying the IPv6 header

Needless to say, the core of the protocol is the datagram format defined by RFC 2460. The datagram design focused mainly on simplicity in order to reduce processing time (headers can be processed at close to link speed) and to keep the size of the headers fixed.

The IPv4 header format contains a lot of fields. Some of these, for example the options field, lie next to the header, leading to fluctuating header sizes.

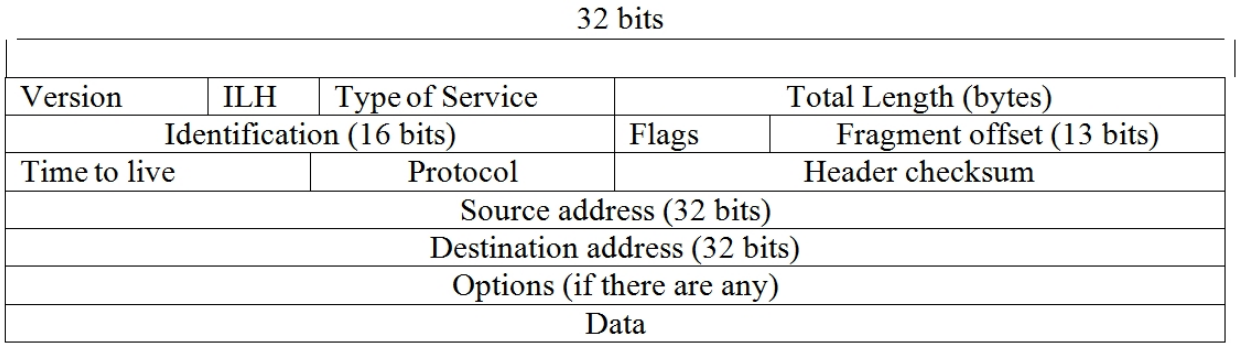

Fig. 1: IPv4 Header

In IPv6, the basic header was minimized and assigned a fixed size. Only essential fields (like addresses or datagram length) are maintained. Everything else has been moved into so-called extension headers, which are attached on demand.

Comparing the IPv4 and IPv6 headers, we can see that although the addresses are four times longer in IPv6, the final size of the IPv6 header is only double that of the IPv4 header.

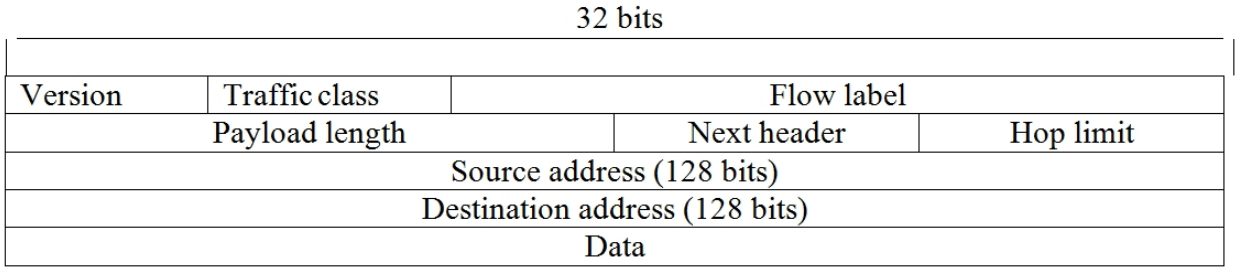

Fig. 2: IPv6 Header

The fields that define the IPv6 header are as follows:

- **Version.** Identifies the protocol version. In the IPv6 protocol, this value is constant (6).
- **Traffic Class** . Distinguishes several traffic classes or priorities (geared towards Quality of Service – QoS).
- **Flow label.** Identifies the flow to which the datagram belongs.
- **Payload length**. Specifies the datagram content length, including extension headers, in bytes. The maximum size is 64 Kbytes.
- **Next header**. Indicates where to find the next header. It can be an extension header or an upper-layer protocol, such as TCP or UDP.
- **Hop limit.** Specifies the maximum number of hops or routings the datagram can cross. Its aim is to protect the IPv6 system against routing loops. The sending node assigns a value to this field to define the scope of the datagram. Every node to route the datagram decreases the value by one. If 0 is reached, the datagram is dropped and an ICMPv6 message is generated to inform the sender.
- **Source address.** Contains the IPv6 address for the node that sent the datagram.
- **Destination address**. Contains the IPv6 address of the node to which the datagram should be delivered.

As mentioned above, extension headers are appended after the basic datagram header. The mechanism that allows extension headers to be attached dynamically is called header chaining. It is implemented using the “next header” field. The “next header” field has two duties: it determines the following extension header or, if there are no more headers, identifies the upper-layer protocol to which the datagram content should be passed.

The extension header types are:
Table 1. Extension header types.

Value	Extension Header
0	Hop-by-hop option
6	TCP
8	EGP
9	IGP
17	UDP
43	Routing
44	Fragmentation
46	RSVP
47	GRE
50	ESP (Encapsulating Security Payload)
51	Authentication Header (AH)
58	ICMP
59	No next header
60	Destination option
62	Mobility Header

1.1.3 MTU Discovery for a route or Path MTU Discovery

The *Path MTU Discovery* is a technique used to determine the minimum MTU (*Maximum Transmission Unit*) for each of the links connecting the sender with the datagram receiver (in other words, the maximum datagram size deliverable along a given route). Datagrams of this size should be sent because they are more efficient.

Path MTU Discovery is based on ICMPv6 messages. The algorithm to calculate the size is simple. The MTU of the outgoing link is used as the first estimation. The sender attempts to send a datagram of this size and, if it receives an

ICMPv6 message saying that a smaller MTU is required, it decreases the size and reapplies the algorithm until it reaches the receiver.

The minimum MTU size allowed on IPv6 supporting links is 1280 bytes. However, we recommend using a value of 1500 bytes to decrease the need for fragmentation.

1.1.4 ICMPv6

The ICMP (*Internet Control Message Protocol*) for IPv6 works in a similar way to the ICMP for IPv4. In ICMPv6 there are two fundamental types of messages: error messages (such as unreachable destination) and information messages (such as requests and echo response messages). Additionally, ICMP packets in IPv6 are used in *Neighbor Discovery*, *Path MTU Discovery* and in *Multicast Listener Discovery*.

All ICMPv6 messages have value 58 in the next header field of the IPv6 header. However, they differ depending on their type and message code.

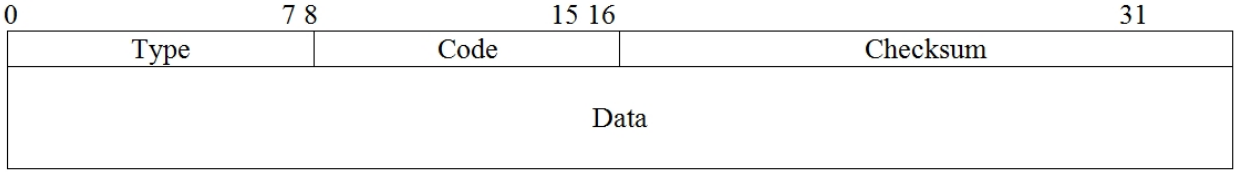

Fig. 3: ICMP message format.

The value of the checksum field is calculated from the other fields. The data field contains error or diagnostic information relevant to IP packet processing.

1.1.4.1 Limitations on sending ICMP error packets

The maximum number of ICMPv6 error packets that can be sent to the network per unit of time is limited by software. A fixed time interval between the sending of error messages is not flexible enough for applications such as *traceroute* and can cause problems. To resolve these errors, limitation is done through the *token bucket* algorithm, which is more flexible than a fixed time interval scheme.

1.1.5 Neighbor Discovery Protocol

Neighbor Discovery (ND) is a protocol that allows different nodes on the same link to advertise their presence to neighbors and to discover neighboring nodes. This is a basic IPv6 functionality.

Neighbor Discovery replaces *router discovery* (RDISC), *Address Resolution Protocol* (ARP) and ICMPv4 *redirect*. It is defined in the following documents:

- RFC 4861, *Neighbor Discovery* for IPv6.
- RFC 4862, *Stateless Address Autoconfiguration*.
- RFC 4443, *Internet Control Message Protocol* for IPv6.

The combination of these protocols allows the IPv6 nodes to detect the presence of other nodes on the same link. *Neighbor Discovery* mandates duplicate address detection so that a node cannot use an IPv6 address that is already being used by another node on the same link. This also allows a node to detect when another node on the link becomes unreachable.

Neighbor Discovery uses the following ICMPv6 protocol messages:

- *Router solicitation* (RS).
- *Router advertisement* (RA).
- *Neighbor solicitation* (NS).
- *Neighbor advertisement* (NA).
- *Redirect*.

The *Neighbor Discovery* process uses ICMPv6 messages and multicast addresses from the solicited node to determine a neighbor's physical address in the same network. The goal is to find out if a neighbor is reachable and to register neighboring routers.

1.1.5.1 Neighbor Discovery Process

Neighbor Discovery is based on the use of two types of messages: *Neighbor Solicitation* (NS) and *Neighbor Advertisement* (NA). The use of each message depends on the context in which the protocol is found.

(1) **NS**. Checks the reachability of a neighbor after having identified the neighbor's link-local address.

(2) **NA**. Indicates any change in the link-local address of a node in the same link. The NA destination address is that of all nodes in the same link (ff02::1).

(3) **NS** and **NA**. Checks the reachability of one of the neighbors. The NS message destination address is the neighbor's unicast address.

A node sends an NS message when it wants to discover the physical address of another node in the same link. The following considerations are taken into account:

- The source address is the IPv6 address of the node sending the message.
- The destination address is the solicited node's multicast address corresponding to the destination node's IPv6 address.
- The NS message also includes the source node's link-local address.

After receiving the NS message, the destination node responds with an NA message. The source address in the NA message is the IPv6 address of the node sending the NS message. The data portion of the NA message includes the physical address of the node sending the NA message. After the source node receives the NA, the source and destination nodes can communicate.

Communication to or through a neighbor may fail for numerous reasons. If the destination fails, communication cannot be recovered and is lost. However, if the route fails, recovery may be possible (a node actively registers the state of the neighbors it is sending packets to).

The *Neighbor Unreachability Detection* (NUD) mechanism is used for all paths between hosts and neighboring nodes (for instance, host to host, host to router, and router to host). NUD can also be used between routers but is not required if similar mechanisms are available.

When the path to a neighbor fails, the specific recovery process depends on how the neighbor is being used. If the neighbor is the destination, address resolution should be performed again. However, if the neighbor is a router, a different router should be selected. In this case, the recovery process is next-hop determination: NUD warns of the need to determine the next hop deleting the neighbor cache entry. NUD is only performed for neighbors that are sent unicast packets; not when sending to multicast addresses.

A neighbor is considered reachable when it returns a positive confirmation (indicating that packets previously sent to the neighbor have been received and processed). If packets are reaching the destination, they are also reaching the next neighbor along the route from the sender. Furthermore, the routing process also confirms that the next neighbor is reachable. For destinations that are not on the same link, the routing process implies that the first router along the route is reachable.

When confirmations from the upper layer protocol are not available, a node probes the neighbor using unicast NS messages to check that the routing path is still available. The receipt of a solicited NA message from the neighbor provides positive confirmation that the routing path is still working. Unsolicited messages only confirm the unidirectional path from the transmission node and receiver, whereas solicited NA messages indicate that the route is working in both directions.

1.1.5.2 Duplicate Address Detection

The *Duplicate Address Detection* (RFC 4862) process detects duplicate unicast addresses on a link. This mechanism should be performed for all unicast addresses, regardless of how they were obtained (*stateless*, *DHCPv6* or manually).

This feature is based on the use of ND messages. A node cannot begin to use a configured address until the DAD process has successfully completed. Meanwhile, the address is set in a *tentative* state (where it can only be used in the *Neighbor Discovery* procedure). This mechanism is set up to prevent a host from processing packets not addressed to it while using, before DAD completion, an address that is being simultaneously used by a node on the link.

It is important to note that the DAD process is particularly demanding for link-local addresses. Since these addresses are formed from the physical address of the interface connected to the link, a DAD failure would duplicate the physical addresses. Therefore, the existence of duplicated link-local addresses on a link would disable IPv6 addressing on the interface until the address is no longer in a *tentative* state.

To perform DAD, the node joins the solicited node multicast address and sends neighbor solicitations with the unspecified address as the source address and the *tentative* address as the destination address. The node also joins the all-nodes multicast group and receives advertisements from other nodes. If another node on the link is using the same address, two situations may occur:

- The duplicate node will receive the NS message and respond with an NA (sent to all-nodes multicast address) thereby exposing the duplicate address.
- The host will receive an NS message with its own address as the destination address from a duplicate node that is also in the process of performing the DAD procedure.

Thus, the DAD procedure expressly notifies the host about another node using its address.

1.1.5.3 IPv6 Neighbor Cache

The *Neighbor Unreachability Detection* (NUD) procedure allows you to detect when a neighbor, be it a host or a router, becomes unreachable on a link. Knowing when a device is unreachable is important because it allows nodes to modify their behavior depending on the state of their neighbors. This reachability information is stored and dynamically updated in the so-called *Neighbor Cache*.

The *Neighbor Cache* contains one entry for each neighbor to which the node has recently sent traffic. Each entry contains an IPv6 unicast address and state information for the address (this state information is mainly used by the NUD algorithm). The permitted states (RFC 4861) are as follows:

- **Incomplete.** The entry has been created but the link layer address has not yet been determined as address resolution is in progress.
- **Reachable.** A positive confirmation has been received indicating that the routing path to the neighbor is working properly.
- **Stale.** There has been no response from the neighbor. Until traffic is sent to the neighbor, no attempt should be made to verify its reachability.
- **Delay.** There has been no response from the neighbor but traffic has been sent to it. In this state, NS packets are delayed for a short time to give upper layer protocols a chance to obtain confirmation of the availability of the neighbor.
- **Probe.** Neighbor reachability is uncertain and messages have been sent to verify reachability.

1.1.5.4 Router Advertisement Messages

Router Advertisement messages are periodically sent by nodes that act as routers on a link. The aim is to advertise information to other hosts. RA messages typically include the following information:

- One or more prefixes with their associated times, valid on the link, which can be used to automatically configure the IPv6 addresses (the prefixes advertised in RA messages must always be 64 bits for autoconfiguration to work properly).
- A set of flags that indicate the type of autoconfiguration (*stateless* or *stateful*) that can be performed.
- Information on the default router and its lifetime value.
- Additional information for hosts, such as the number of hops or the MTU.
- Optional information for hosts, recursive DNS server option (RDNSS, *Recursive DNS Server*).

Router Solicitation (RS) messages are sent by hosts to request information in order to autoconfigure without having to wait for the next planned RA message.

Since RS messages are usually sent when the hosts start up, the source address is usually the unspecified IPv6 address. If the host has a unicast address configured, this is used as the source address in the message. The destination address in RS messages is the special *all-nodes link-local multicast group* multicast address.

1.1.5.5 Default Router Preferences

The *Neighbor Discovery* protocol specifies a conceptual model for hosts that involves defining a list of default routers and a list of prefixes. Hosts send RS messages and receive RA messages from routers. They build their list of default routers and their list of prefixes based on the information advertised in RA messages.

In some network topologies, where the hosts have multiple routers on their default router list, the choice of router for an off-link destination is important. In certain situations, one router may reach the destination better than another. In others, however, choosing the wrong router may lead to a communication error.

The *Default Router Preferences* (RFC 4191) extension adds a priority level (low, medium or high) for each of the default routers advertised. A default router's DRP is communicated in the unused bits of the RA messages. This extension is compatible for both routers (assigning DRP bits) and hosts (interpreting DRP bits), and ignored by those hosts that do not need to implement the DRP extension. Similarly, the values sent by routers that do not implement the DRP extension will be interpreted by the hosts that do implement it as having medium priority.

1.1.5.6 Neighbor Redirect Messages

In IPv6, routers are responsible for detecting when a host within the local network has not chosen the best next hop, and for trying to correct it. *Redirect* messages are sent by an IPv6 router to alert the source host that a better first hop exists to reach a specified destination. These messages can also be sent by a router to tell a host that the message destination is on the same link.

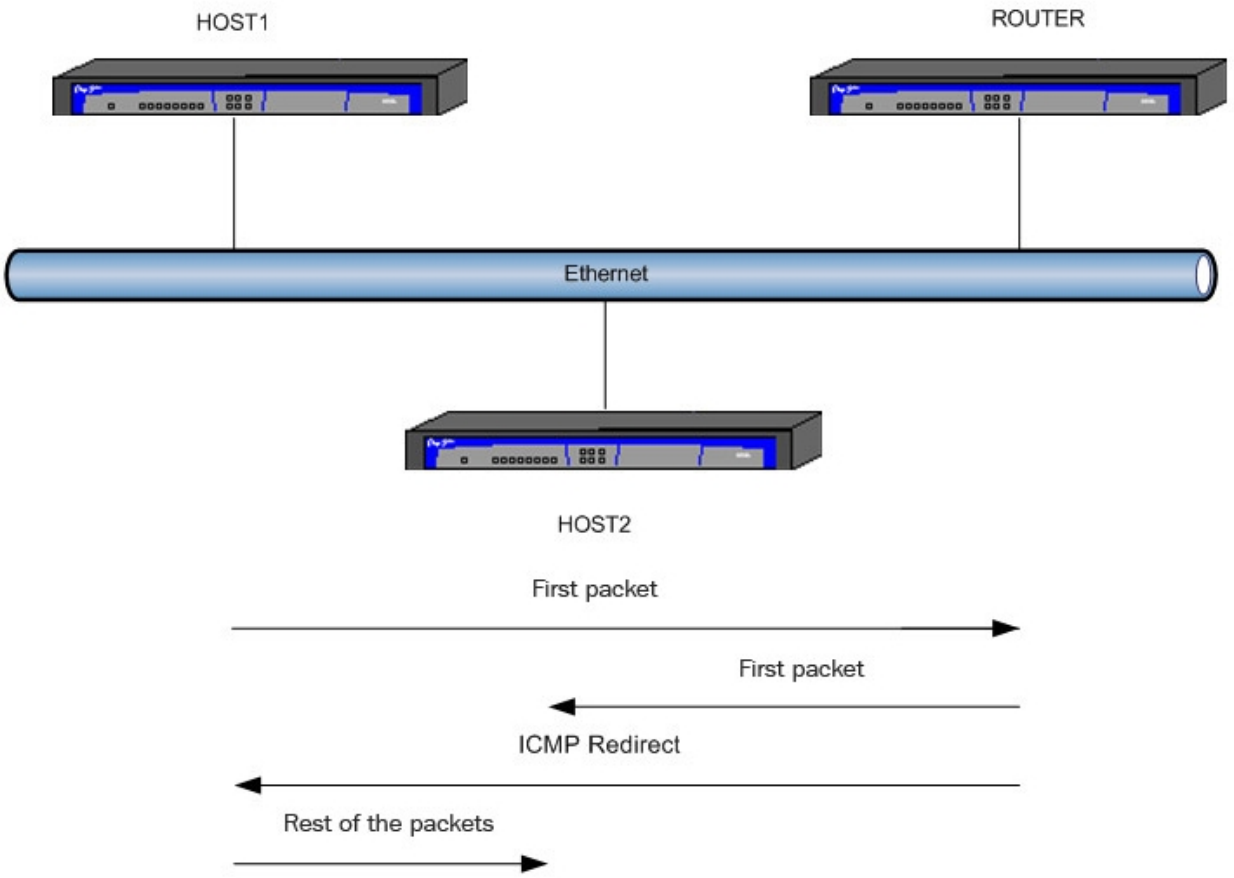

Fig. 4: Redirect example

Only routers (not hosts) can send *Redirect* messages. Hosts are only responsible for listening and processing these messages. A host receiving such a message inspects it to find the address to which the datagram should be routed for that destination.

1.1.6 Configuring addresses through SLAAC and General prefix

As already mentioned, addresses can be generated using three fundamental mechanisms: *stateful, stateless*, and *manual*. The *stateful* and *stateless* configurations provide autoconfiguration address mechanisms.

Stateful autoconfiguration is achieved by means of the DHCPv6 protocol. This feature is explained in the *Teldat*- **Dm806-I DHCPv6 Protocol** manual. *Stateless* autoconfiguration is based on the SLAAC procedure, whose fundamental idea is to form an address from a 64-bit prefix (announced in an RA message) and a suffix generated in the host itself. Finally, manual configuration offers several possibilities. The most basic one is to assign an absolute address to the interface. However, an address can also be generated from a *general prefix* along with a subnet identifier and an address. It is important to note that, in the latter case and in our devices, identification of the *general prefix* is performed by means of a label. Consequently, the prefix value can change (adding some degree of automation to address generation) over time.

Stateless autoconfiguration and address generation through a *general prefix* are explained in more detail in the following subsections.

1.1.6.1 Stateless Autoconfiguration in IPv6

As already mentioned in previous sections, all interfaces on IPv6 nodes must have a *link-local* address (which is usually automatically configured) in order to communicate on the link. This address is essential for the SLAAC procedure.

Nodes can connect to a network and automatically generate global IPv6 addresses without having to manually configure them or to explicitly ask a DHCPv6 server for them (using *stateful* autoconfiguration). Thanks to the *Router Advertisement* messages, a node acting as router can advertise prefixes and function as the default route for the link.

These prefixes, periodically obtained or requested through *Router Solicitation* messages, allow hosts to build addresses provided they do not exceed 64 bits in length. The remaining 64 bits are obtained from the identifier belonging to the interface to which the address will be assigned. The value of this last part of the address will depend on the physical address assigned to the interface. Typically, this means obtaining an EUI-64 identifier from the MAC address while respecting the following procedure:

- The seventh most significant bit in the MAC address is inverted.
- The 4 hexadecimal "*fffe*" digits are inserted between the third and fourth byte of the result obtained in the previous step

Thus, if prefix 2001:0db8:a23:8005::/64 is obtained and the user wishes to configure an address using SLAAC on an interface with MAC address 00:a0:26:00:00:01, the address would be defined as follows:

2001:0db8:a23:8005::20a0:26ff:fe00:1

Once the address is created, and since it is a unicast one, it will be marked as *tentative* until the DAD procedure completes successfully. Furthermore, if the prefixes advertised are globally unique, then so are the global addresses stemming from them.

1.1.6.1.1 *Simplified network renumbering*

One of the advantages of using *stateless* autoconfiguration is the ability to add or remove prefixes dynamically whenever the service provider introduces a change.

Whenever a new prefix is incorporated, it is announced in *Router Advertisement* messages (with a corresponding valid and preferred lifetime). Thus, prefix transitions become transparent to the host when they cease to be used according to their associated lifetimes. An advertised prefix with a preferred lifetime equal to 0 explicitly indicates to all hosts on the same link that the prefix should be eliminated and no longer used.

1.1.6.2 General Prefixes

The basic aim of *general prefixes* is to define a "general" prefix with which to generate addresses with their own *subnet identifier* (which may be 0 bits in length). *General prefix* identification is performed by means of a label, so when a prefix is modified the address changes too. For example, if we start with base prefix 2001:db8::/48 and address ::1:20a0:26ff:fe00:1/64, we would get the following address:

2001:db8:0:1:20a0:26ff:fe00:1/64 (this being the identifier of subnet 1/16).

A specific prefix is created from the 64 bits that result from combining the *general prefix* with the *subnet identifier*. In the above example, this is 2001:db8:0:1::/64.

General prefixes can be defined in different ways:

- Manually.
- Through the DHCPv6 protocol (for further information on this, please see the "**Dm806-I DHCPv6 Protocol**" manual).
- Using a 6to4 interface (for further information on this, please see the "**Dm810-I IPv6 Tunnel over IPv4"** manual).
- Using a 6rd interface (for further information on this, please see the "**Dm810-I IPv6 Tunnel over IPv4"** manual).

This feature allows you to define a hierarchy within the IPv6 address space. For example, a prefix can be subdivided into several prefixes in an internal network, or several prefixes can be joined together into a single and shorter prefix to be advertised by the network.

1.1.7 Multihoming

In IPv6, multiple IPv6 prefixes can be assigned to the same host. This allows, for example, access to multiple ISPs without having to change the global routing table should a link become unreachable. This is shown in the following example.

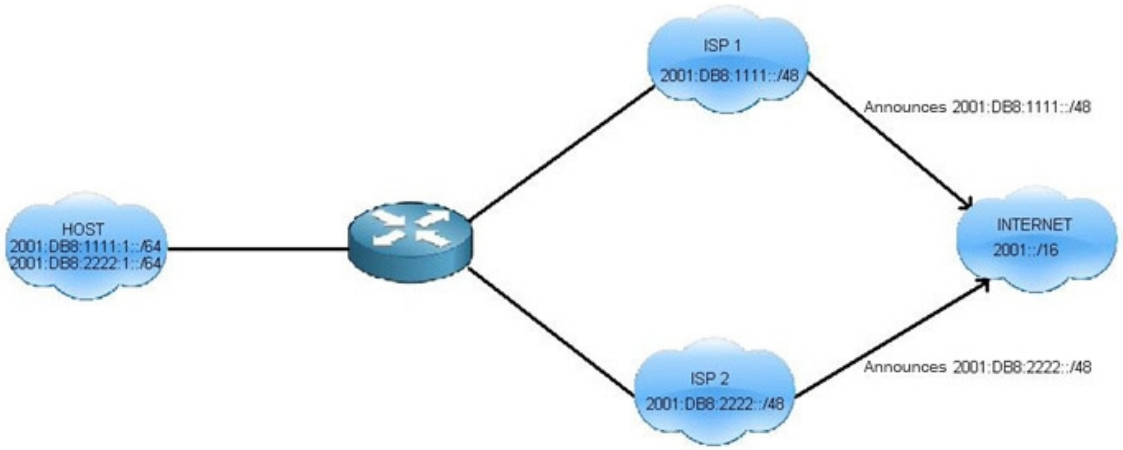

Fig. 5: Multihoming

As you can see, the two ISPs advertise different prefixes to the same host. The host, in turn, creates two addresses which can be used to access the Internet by one of the two paths.

1.1.8 Data link

A data link is a network that shares a *link-local* prefix. An administrator can segment the network to provide a multi-level hierarchy that protects it from the complexity of the connected networks. A subnet prefix can be associated with one data link, while a data link can be assigned many subnet prefixes.

Data links supported by our devices in IPv6 are: PPP, Ethernet, Ethernet subinterface, FastEthernet, TNIP, WLAN, Bvi and Loopback.

1.1.9 Dual Stacks

According to RFC 4213, *dual stack* is a technique that provides full support for IPv4 and IPv6 protocols in hosts and routers.

The IPv6 nodes that fully implement both protocols are called “IPv6/IPv4 nodes”. These nodes have the ability to send and receive IPv4 packets with IPv4 nodes, as well as to interoperate directly with IPv6 nodes using IPv6 packets.

This type of nodes, with both implementations, have three operating modes:

- With the IPv4 protocol stack enabled and the IPv6 one disabled.
- With the IPv6 protocol stack enabled and the IPv4 one disabled.
- With both stacks enabled.

The last operating mode means that applications of both protocols can co-exist in the same node, each using its own stack, as shown in the following figure:

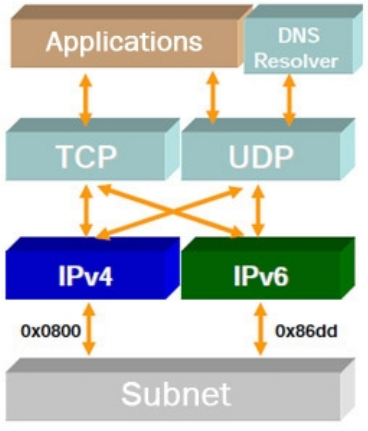

Fig. 6: Dual Stack.

Chapter 2 Configuration

2.1 IPv6 Configuration Commands

This chapter summarizes and details the router configuration commands involved in IPv6 protocol addressing. Enter the following commands to access the IPv6 protocol configuration menu:

```
*config
Config>protocol ipv6
-- IPv6 user configuration --
IPv6 config>
```

Command	Function
? (HELP)	Lists the commands or their options.
ACCESS-GROUP	Configures the IPv6 access control.
GENERAL-PREFIX	Configures the general prefixes.
ICMP	Configures the ICMP parameters.
IPV6-PARAM	Configures the IPv6 parameters.
NEIGHBOR	Configures a neighbor.
NO	Deletes a previously added parameter or establishes its default value.
ROUTE	Adds a static route to a network or a subnet.
UNICAST-ROUTING	Enables unicast routing.
VRF	Enters the IPv6 configuration menu for a VPN Routing/Forwarding instance.
VRRP	Enters the VRRP global configuration menu for IPv6.
EXIT	Exits the IPv6 configuration menu.

2.1.1 ? (HELP)

Lists the valid commands at the layer at which the router is programmed. You can also use this command after a specific command to list the available options:

Syntax:

```
IPv6 config>?
```

Example:

```
IPv6 config>?
  access-group       IPv6 access-list filter
  general-prefix     Configure a general IPv6 prefix
  icmp               Configure ICMP parameters
  ipv6-param         Configure other IPv6 parameters
  neighbor           Neighbor
  no                 Negate a command or set its defaults
  route              Configure a static network/subnet IPv6 route
  unicast-routing    Enable unicast routing
  vrrp               Enter in the VRRP configuration menus
  exit
IPv6 config>
```

Command history:

Release	Modification
11.00.04, 11.00.03.01.02, 11.01.00	The VRRPv3 protocol for IPv6 was introduced and the **vrrp** command was added as a result.

2.1.2 access-group

Associates an access list with the IPv6 protocol. This filter will only apply to the device's local traffic (i.e., traffic generated locally or addressed to the device). The configuration is as follows:

Syntax:

```
access-group <AL-name> {in|out}
```

In:	Applies the access list to the traffic addressed to the device.
Out:	Applies the access list to the traffic generated in the device.

Example:

```
IPv6 config>access-group ?
  <1..50 chars>    IPv6 access-list name
IPv6 config>access-group list1 ?
  in     Inbound traffic filter
  out    Outbound traffic filter
IPv6 config>access-group list1 out
```

For further information on *Stateless* filtering, please see the "**Dm808-I IPv6 Access Control**" manual.

2.1.3 general-prefix

Configures a *general prefix*. With a *general prefix* you can configure global addresses or define a short prefix from which to obtain more specific ones.

Syntax:

```
general-prefix <prefix-name> {prefix/pref-len | 6to4 <interface> | 6rd <interface>}
```

prefix-name	Name of the *general-prefix*.
prefix/pref-len:	Configures a prefix with the prefix /prefix length format.
6to4:	Configures a *general-prefix* for interfaces with 6to4 tunnels. Please see the **Teldat-Dm810-I IPv6 over IPv4 Tunnel** manual.
6rd:	Configures a *general-prefix* for interfaces with 6rd tunnels. Please see the **Teldat-Dm810-I IPv6 over IPv4 Tunnel** manual.

Example:

```
IPv6 config>general-prefix prefix1 ?
  6to4         Create 6to4 prefix based on 6to4 interface
  6rd          Create 6rd prefix based on 6rd interface
  <a::b/l>     IPv6 prefix
IPv6 config>general-prefix prefix1 2001:db8:1111::/64
```

For these prefixes to become global addresses, an address associated with the *general prefix* must be configured in the IPv6 menu of each interface. Please see **IPv6 commands per interface** in section 2 of this chapter.

2.1.4 icmp

Allows you to configure certain parameters of the ICMPv6 protocol.

Syntax:

```
IPv6 config>icmp ?
  burstlimit    Burstlimit between ICMP error messages
  ratelimit     Ratelimit between ICMP error messages
```

2.1.4.1 burstlimit

Allows you to configure the number of ICMPv6 error messages sent in the time period defined by the rate limit. This parameter is set to 0 by default, which means that there is no limit in the number of messages sent in a period of time. The configuration is as follows:

Syntax:

```
icmp burstlimit <number-of-messages>
```

number-of-messages:	This number represents the maximum amount of ICMPv6 error packages that are to be sent within a given time period. Values range from 0 to 100. The default value is 0.

Example:

```
IPv6 config>icmp burstlimit ?
  <0..100>    Number of ICMP error responses in the timeout defined
IPv6 config>icmp burstlimit 8
```

In this example, we have set the maximum number of ICMPv6 error messages to eight for the time period defined by the ratelimit.

Command history:

Release	Modification
11.01.07	The "*burstlimit*" option has been added to the "*icmp*" command.

2.1.4.2 ratelimit

Allows you to limit the rate of ICMP error messages in a period of time. The configuration is as follows:

Syntax:

```
icmp ratelimit <time-in-milliseconds>
```

time-in-milliseconds:	Transmission time interval for messages produced by an error. Values range from 0 to 2147483647 milliseconds. The default value is 1 second.

Example:

```
IPv6 config>icmp ratelimit ?
  <0..2147483647>    Ratelimit in milliseconds
IPv6 config>icmp ratelimit 2000
```

In this example, we have set a maximum transmission rate of 2 seconds.

2.1.5 Ipv6-param

Configures other parameters related to IPv6, such as:

2.1.5.1 routing-table-size

The size of the FIB routing table. By default, there is no limit.

Syntax:

```
ipv6-param routing-table-size <number-of-routes>
```

number-of-routes:	Maximum number of IPv6 routes in the FIB routing table. By default, there is no limit.

Example:

```
IPv6 config>ipv6-param ?
  routing-table-size    Set the maximum size of the IPv6 fib route table
IPv6 config>ipv6-param routing-table-size ?
  <0..2147483647>    Number of routes
IPv6 config>ipv6-param routing-table-size 4000
```

In this example, the maximum number of routes has been limited to 4000 IPv6 routes. You cannot add more routes to the table once this limit has been reached.

Note

Any dynamic changes in value take effect when they are introduced. As a result, if, for example, you reduce the size of the routing table dynamically and the number of routes in the table at the time exceeds the limit you have set, the existing routes will not be deleted but you will be unable to add any others.

2.1.6 neighbor

Configures a static entry in the neighbor cache. This way, you can define a neighbor and avoid the *Address Resolution* (*AR,* ND protocol) process through which you obtain a device's hardware address. To do so, follow these steps:

Syntax:

```
ipv6 neighbor <ipv6-address> <interface> <hardware-address>
```

ipv6-address:	IPv6 address corresponding to the neighbor you wish to add. This is documented in RFC 4291.
interface:	Interface through which you can reach that neighbor.
hardware-address:	Neighbor's hardware address (48 bits).

Example:

```
IPv6 config>neighbor ?
  <a::b>    IPv6 Neighbor address
IPv6 config>neighbor 1111::1111 ?
  <interface>    Interface name
IPv6 config>neighbor 1111::1111 ethernet0/0 ?
  <mac>    MAC format (xx-xx-xx-xx-xx-xx)
IPv6 config>neighbor 1111::1111 ethernet0/0 aa-bb-cc-dd-ee-ff
```

2.1.7 route

Sets the static IPv6 routes. For further information on static routes, please see the Teldat- **Dm807-I IPv6 Static Routing** manual.

Syntax:

```
IPv6 config>route <prefix> {<nhop-addr> [interface <ifc>]} |
{interface <ifc> [<nhop-addr>]} [<dist>] [track nsla-advisor <advisor-id>]
```

Options:

prefix:	Defines the route's destination network.
nhop-addr:	IPv6 address of the next hop. If the *ifc* output interface is not specified, this is a recursive route.
ifc:	Output interface. If *nhop-addr* is configured, it is a fully specified route. If not, it is a directly connected route.
dist:	Administrative distance. This is used to define floating routes.
advisor-id:	Allows you to link the defined static route to an advisor so that, when the advisor output is enabled (TRUE), the route behaves like a normal static route. However, when the advisor output is disabled (FALSE), the route is considered nonexistent (i.e., not configured). For further information on the **Network Service Level Advisor**, please see the **Teldat-DM754-I NSLA** manual.

Example:

```
IPv6 config>route 2001:db8:1000::/48 2001:db8:3000::1
```

2.1.8 no

Negates another command or restores the default configuration for a specific parameter.

Syntax:

```
IPv6 config>no ?
  access-group       IPv6 access-list filter
  general-prefix     Configure a general IPv6 prefix
  icmp               Configure ICMP parameters
  ipv6-param         Configure other IPv6 parameters
  neighbor           Neighbor
```

```
  route              Configure a static network/subnet IPv6 route
  unicast-routing    Enable unicast routing
```

2.1.9 unicast-routing

Enables unicast datagram routing in IPv6. This causes the device to function as a router and not a host.

Syntax:

```
unicast-routing
```

Example:

```
IPv6 config>unicast-routing
```

2.1.10 vrf

Accesses a specified VRF IPv6 configuration. On accessing a VRF IPv6 configuration menu, the prompt changes to **IPv6 vrf config>**. Use the **exit** command to return to the main VRF IPv6 configuration.

Example:

```
IPv6 config>vrf v1

-- IPv6 user configuration for a VRF --
IPv6 vrf config>
```

Command history:

Release	Modification
11.01.09	The VRF command was introduced as of version 11.01.09.

2.1.11 vrrp

Accesses the global configuration menu of the VRRP protocol for IPv6. For more information, please see the "**Dm821-I VRRPv3 Protocol for IPv6**" manual.

Command history:

Release	Modification
11.00.04	The VRRPv3 protocol for IPv6 was introduced as of version 11.00.04.
11.00.03.01.02	The VRRPv3 protocol for IPv6 was introduced as of version 11.00.03.01.02.
11.01.00	The VRRPv3 protocol for IPv6 was introduced as of version 11.01.00.

2.1.12 exit

Returns to the previous prompt level.

Syntax:

```
exit
```

Example:

```
IPv6 config>exit
Config>
```

2.2 IPv6 Command Per Interface

Specifies the configuration commands related to IPv6 that are available in the configuration menus of the interfaces that support IPv6. The IPv6 commands available are:

Command	Function
? (HELP)	Lists the commands or their options.
ACCESS-GROUP	Configures the access control per interface.

ADDRESS	Configures IPv6 addresses on the interfaces.
DHCP	Configures the DHCP parameters for IPv6.
ENABLE	Enables IPv6 on the interface.
HOP-LIMIT	Configures a hop limit.
HOST-MODE	Configures the interface in host mode.
MTU	Establishes the maximum transmission unit.
ND	Configures *Neighbor Discovery.*
OSPFV3	Configures OSPFv3 protocol parameters on the selected interface.
REDIRECTS	Enables IPv6 *Redirect* message sending.
RIPNG	Configures RIPng protocol parameters on the selected interface.
UNREACHABLES	Enables IPv6 *Unreachable* message sending.
VRRP	Configures VRRP Virtual Routers in IPv6.

To access the IPv6 configuration menu belonging to each interface, you need to type the following:

```
Config>network ethernet0/0

-- Ethernet Interface User Configuration --
ethernet0/0 config>
```

2.2.1 ? (HELP)

Lists the commands found at the layer where the router is programmed. You can also use this command after a specific command to list the available options:

Syntax:

```
ethernet0/0 config>ipv6 ?
```

Example:

```
ethernet0/0 config>ipv6 ?
  access-group    Specify per-interface access control system
  address         Assign an IPv6 address to this network interface
  dhcp            Dhcp ipv6 configuration
  enable          Enable IPv6 protocol
  hop-limit       Configure hop count limit
  host-mode       Inhibit forwarding on interface
  mtu             Set IPv6 Maximum Transmission Unit
  nd              Configure IPv6 Neighbor Discovering
  ospfv3          Open Shortest Path First version 3
  redirects       Enable sending of ICMP Redirect messages
  ripng           RIPng
  unreachables    ICMP Unreachable messages
  vrrp            Enter in the VRRP configuration menus
```

Command history:

Release	Modification
11.00.04, 11.00.03.01.02, 11.01.00	The VRRPv3 protocol for IPv6 was introduced and the **vrrp** command was added as a result.

2.2.2 access-group

Applies an IPv6 access list to the required interface.

Syntax:

```
ipv6 access-group <AL-name> {in|out}
```

In:	Applies the access list to incoming traffic on the interface.
Out:	Applies the access list to outbound traffic on the interface.

Example:

```
ethernet0/0 config>ipv access-group ?
  <word>    IPv6 access-list name
ethernet0/0 config>ipv access-group list1 ?
  in     Inbound traffic filter
  out    Outbound traffic filter
ethernet0/0 config>ipv access-group list1 in
```

For further information on *Stateless* filtering, please see the Teldat- **Dm808-I IPv6 Access Control** manual.

2.2.3 address

Configures an IPv6 address on the interface (and enables it when the **ipv6 enable** command has not been configured). This command is used to configure:

- The interface's *link-local* address.
- Global addresses defining the prefix and length. You may also choose whether the address is meant to be an anycast address or one made up by the interface identifier (through the *eui-64* command).
- Global addresses through autoconfiguration.
- Global addresses through *general prefix*.

Syntax:

```
ipv6 address {<ipv6-address> link-local | <pref/pref-len> [anycast | eui-64] | autoconfig [address-limit
<limit> | default] | general-prefix <genpref-name> <pref/pref-len> [anycast | eui-64]}
```

ipv6-address:	*link-local* address for this interface.
pref/pref-len:	Global address, with prefix and length configured on this interface.
anycast:	Indicates whether the previously configured global address (*pref/pref-len*) is an anycast address.
eui-64:	Indicates whether the previously configured global address (*pref/pref-len*) is an address formed with the interface identifier.
autoconfig:	Option that allows automatic configuration of global addresses through the receipt of prefixes included in the RAs.
address-limit:	Limits the number of addresses that can be automatically configured on this interface.
limit:	Maximum number of addresses that will be automatically configured on this interface.
default:	Option allowing you to select the default router from among the RAs received by the interface, in the event the device is in router mode and the interface in host mode (please see the **host-mode** command).
genpref-name:	Name of the *general prefix* with which the preceding global address is formed.

Examples:

An example of a *link-local* address:

```
ethernet0/0 config>ipv6 address fe80::1234 link-local
```

An example of a global address:

```
ethernet0/0 config>ipv6 address 2001:db8:1111::2222/64 ?
  anycast    Configure as an anycast
  eui-64     Use eui-64 interface identifier
  <cr>
ethernet0/0 config>ipv6 address 2001:db8:1111::2222/64 anycast
```

Autoconfiguration example:

```
ethernet0/0 config>ipv6 address autoconfig
```

An example of a global address with *general prefix* defined (2001:db8:3333::/48):

```
ethernet0/0 config>ipv6 address general-prefix teldat 0:0:0:1212::12/64
```

The resulting global address is: 2001:db8:3333:1212::12/64

2.2.4 dhcp

Enables the DHCP client and triggers the *prefix delegation* process on this interface.

Syntax:

```
ipv6 dhcp client pd <genpref-name>
```

genpref-name: Name of the *general prefix* that stores the prefix obtained through DHCP.

Example:

```
ethernet0/0 config>ipv6 dhcp ?
  client    Client dhcp ipv6 configuration
ethernet0/0 config>ipv6 dhcp clie
ethernet0/0 config>ipv6 dhcp client ?
  pd    Prefix delegation
ethernet0/0 config>ipv6 dhcp client pd ?
  <word>    Prefix name
ethernet0/0 config>ipv6 dhcp client pd teldat
```

For further information, please see the Teldat-**Dm806-I DHCPv6 Protocol** manual.

2.2.5 enable

Enables IPv6 on the interface. This command must be used to enable IPv6 if no other command does so (as is the case with **ipv6 address**, for example).The **"ipv6 enable"** command automatically configures the *link-local* address for this interface.

As with enabling, to disable IPv6 on an interface you cannot simply negate the command with **"no ipv6 enable"** if there is another IPv6 command configured that enables it. You must delete these commands from the configuration.

Syntax:

```
ipv6 enable
```

Example:

```
ethernet0/0 config>ipv6 enable
```

2.2.6 hop-limit

Configures the maximum number of hops used in an RA and in all the IPv6 packets that originated in the device.

Syntax:

```
ipv6 hop-limit <value>
```

value:	Maximum number of hops. This value ranges between 1 and 255. The default value is 64.

Example:

```
ethernet0/0 config>ipv6 hop-limit ?
  <1..255>    Hop Limit value
ethernet0/0 config>ipv6 hop-limit 28
```

2.2.7 host-mode

The **"ipv6 host-mode unicast"** command forces the interface where the behavior settings have been configured to function as a host and not as a router. It inhibits the routing of packets through this interface.

Syntax:

```
ipv6 host-mode unicast
```

Example:

```
ethernet0/0 config>ipv6 host-mode ?
  unicast    Unicast traffic
ethernet0/0 config>ipv6 host-mode unicast
```

2.2.8 mtu

The *Maximum Transmission Unit* (MTU) sets the maximum size of IPv6 packets that can be transmitted by an interface. This value is configurable. The MTU value is sent in the RAs to enable the *Path MTU Discovery* process and to fragment the packets solely at their source and destination (since intermediate routers do not fragment).

Syntax:

```
ipv6 mtu <value>
```

value:	MTU value in bytes. The MTU default value changes depending on the type of interface. The minimum MTU value is 1280 bytes.

Example:

```
ethernet0/0 config>ipv6 mtu ?
  <1280..1500>    MTU (bytes)
ethernet0/0 config>ipv6 mtu 1500
```

2.2.9 nd

Encompasses all configurable parameters of the *Neighbor Discovery* protocol. These commands are listed below and then explained in detail:

```
ethernet0/0 config>ipv6 nd ?
  accept-redirects       Accept receiving ICMP Redirect messages
  dad                    Duplicate Address Detection
  managed-config-flag    Hosts should use DHCP for address config
  ns-interval            Set advertised NS retransmission interval
  other-config-flag      Hosts should use DHCP for non-address config
  prefix                 IPv6 prefix
  ra                     Router Advertisement setup
  rdnss                  Set advertised Recursive DNS server
  reachable-time         Set advertised reachability time
  router-preference      Set default router preference value
  rs                     Router Solicitation setup
  target-ll-option       Force sending Target Link-Layer Address Option
```

2.2.9.1 accept-redirects

The **"ipv6 nd accept-redirects"** subcommand is used to enable the reception of ICMP *Redirect* messages from other routers. This command is enabled by default and redirect messages are always received. Thus, you have to negate the command to ensure that any redirect messages received by this interface are ignored.

Syntax:

```
no ipv6 nd accept-redirects
```

Example:

```
ethernet0/0 config>no ipv6 nd accept-redirects
```

2.2.9.2 dad

The **"ipv6 nd dad"** subcommand, in turn, contains the following options:

```
ethernet0/0 config>ipv6 nd dad ?
  attempts    Set IPv6 Duplicate Address Detection Transmits
  disable     Disable IPv6 Duplicate Address Detection
  time        Set IPv6 Duplicate Address Detection Time
```

2.2.9.2.1 attempts

The **"ipv6 nd dad attempts"** command configures the number of NS messages that are sent to an interface while performing DAD when this is satisfactory.

Syntax:

```
ipv6 nd dad attempts <value>
```

value:	Number of times the NS is sent. Valid values range from 1 to 600. The default value is 1.

Example:

```
ethernet0/0 config>ipv6 nd dad attempts ?
  <1..600>    Number of attempts
ethernet0/0 config>ipv6 nd dad attempts 5
ethernet0/0 config>
```

2.2.9.2.2 disable

The **"ipv6 nd dad disable"** command disables duplicate address detection (*DAD*).

Syntax:

```
ipv6 nd dad disable
```

Example:

```
ethernet0/0 config>ipv6 nd dad disable
```

2.2.9.2.3 time

The **"ipv6 nd dad time"** command configures the time interval between NS retransmissions for the DAD procedure. This is a different time interval from the one that appears later (**ns-interval**), which is used for *address resolution*.

Syntax:

```
ipv6 nd dad time <value>
```

value:	Time interval, in milliseconds, between NS message retransmissions. Valid values range from 1 millisecond to 6 seconds. The default value is 1 second

Example:

```
ethernet0/0 config>ipv6 nd dad time ?
  <1..6000>    time in milliseconds
ethernet0/0 config>ipv6 nd dad time 2000
```

2.2.9.3 managed-config-flag

Enables the **managed-config-flag** (or M bit) in the RAs and tells the hosts that addresses are available via DHCP.

Syntax:

```
ipv6 nd managed-config-flag
```

Example:

```
ethernet0/0 config>ipv6 nd managed-config-flag
```

2.2.9.4 ns-interval

Configures the time interval between NS retransmissions on an interface. By default, this value changes the NS retransmission interval for *Address Resolution* and *DAD*. If another interval is required for NS retransmission in DAD, it must be configured through the **ipv6 nd dad time** command.

Example:

```
ipv6 nd ns-interval <value>
```

value:	Time interval, in milliseconds, between NS retransmissions. Valid values range from 1000 to 172800000 milliseconds.

Syntax:

```
ethernet0/0 config>ipv6 nd ns-interval 20000
```

2.2.9.5 other-config-flag

Enables the **other-config-flag** (or O bit) in the RAs and tells the hosts that other information is available via DHCP. If the M bit is set to 1, the O bit is unnecessary as it is redundant.

Syntax:

```
ipv6 nd other-config-flag
```

Example:

```
ethernet0/0 config>ipv6 nd other-config-flag
```

2.2.9.6 prefix

The **"ipv6 nd prefix"** subcommand, in turn, contains the following options:

```
ethernet0/0 config>ipv6 nd prefix ?
  <a::b/l>    IPv6 prefix
  default     Specify prefix default parameters
```

2.2.9.6.1 Configuring a prefix

The first option that appears when you run the help command allows you to configure a prefix with the following options:

```
ethernet0/0 config>ipv6 nd prefix 2001:db8:1111::/64 ?
  <0..4294967294>    Valid Lifetime (secs)
  infinite           Infinite Valid Lifetime
  no-advertise       Do not advertise prefix
  <cr>
```

If you select the *no-advertise* option, you cannot configure anything else for this prefix.

The first three options allow you to set a valid and a preferred lifetime for the prefix, followed by a few flags that indicate the characteristics of that particular prefix.

Syntax:

```
ipv6 nd prefix <prefix/pref-len> [<valid-lifetime> <preferred-lifetime> [no-autoconfig | no-onlink |
off-link]] | [infinite {<preferred-lifetime> | infinite} [no-autoconfig | no-onlink | off-link]] |
[no-advertise]
```

prefix/preflen:	Prefix to be included in the RA.
valid-lifetime:	Length of time, in seconds, during which the prefix is advertised as valid. Optional.
preferred-lifetime:	Length of time, in seconds, during which the prefix is advertised as preferred. Optional.
no-autoconfig:	Specifies that hosts that receive prefixes with this flag enabled cannot use it to automatically configure global addresses. The prefix is advertised with the A bit set to 0. Optional.
no-onlink:	Configures the prefix as *no-onlink*. The prefix is advertised with the L bit set to 0. Optional.
off-link:	Configures the prefix as *off-link*. The prefix is advertised with the L bit set to 0. This prefix is not added to the routing table as a directly connected prefix. If this was already added, this flag indicates that it is no longer valid and it is deleted from the routing table. Optional.
infinite:	Infinite valid/preferred prefix lifetime. Optional.
no-advertise:	The prefix is not advertised. Optional.

The default values are those taken by the prefixes if their options are not specifically configured. A prefix has:

- A 30-day valid lifetime.
- A 7-day preferred lifetime.
- They will be inserted in a routing table as directly connected.
- They will be advertised as *on-link* prefixes.
- They can be used in autoconfiguration.

Example:

```
ethernet0/0 config>ipv6 nd prefix ?
  <a::b/l>    IPv6 prefix
  default     Specify prefix default parameters
ethernet0/0 config>ipv6 nd prefix 2001:db8:1111::/64 ?
  <0..4294967294>    Valid Lifetime (secs)
```

```
  infinite           Infinite Valid Lifetime
  no-advertise       Do not advertise prefix
  <cr>
ethernet0/0 config>ipv6 nd prefix 2001:db8:1111::/64 infinite ?
  <0..4294967294>    Preferred Lifetime (secs)
  infinite           Infinite Preferred Lifetime
ethernet0/0 config>ipv6 nd prefix 2001:db8:1111::/64 infinite 10000 ?
  no-autoconfig      Do not use prefix for autoconfiguration
  no-onlink          Do not use prefix for onlink determination
  off-link           Prefix is offlink
  <cr>
ethernet0/0 config>ipv6 nd prefix 2001:db8:1111::/64 infinite 10000 no-onlink ?
  no-autoconfig      Do not use prefix for autoconfiguration
  <cr>
ethernet0/0 config>ipv6 nd prefix 2001:db8:1111::/64 infinite 10000 no-onlink no-autoconfig
```

2.2.9.6.2 Configuring the default values

Default values can be configured through the **"ipv6 nd prefix default"** command. Prefixes without options take these values. You use the same configuration pattern as for a prefix.

Syntax:

```
ipv6 nd prefix default [<valid-lifetime> <preferred-lifetime> [no-autoconfig | no-onlink |
off-link]] | [infinite {<preferred-lifetime> | infinite} [no-autoconfig | no-onlink | off-link]] |
[no-advertise]
```

Example:

```
ethernet0/0 config>ipv6 nd prefix ?
  <a::b/l>     IPv6 prefix
  default      Specify prefix default parameters
ethernet0/0 config>ipv6 nd prefix default ?
  <0..4294967294>    Valid Lifetime (secs)
  infinite           Infinite Valid Lifetime
  no-advertise       Do not advertise prefix
ethernet0/0 config>ipv6 nd prefix default 7800 ?
  <0..4294967294>    Preferred Lifetime (secs)
ethernet0/0 config>ipv6 nd prefix default 7800 7200 ?
  no-autoconfig     Do not use prefix for autoconfiguration
  no-onlink         Do not use prefix for onlink determination
  off-link          Prefix is offlink
  <cr>
```

Now, all prefixes without configuration options take the default values configured in the **default** command.

2.2.9.7 ra

The **"ipv6 nd ra"** subcommand, in turn, has the following options:

```
ethernet0/0 config>ipv6 nd ra ?
  accept-ra            Accept Router Advertisements
  default-router       Learn Default Router in Router Advertisements
  hop-limit            IPv6 RA hop-limit value
  interval             Set Router Advertisement Interval
  lifetime             Set Router Advertisement Lifetime
  mtu                  IPv6 RA MTU Option
  prefix-info          Learn Prefix Information in Router Advertisement
  source-ll-address    IPv6 RA Source Link Local Address Option
  suppress             Suppress IPv6 Router Advertisements
```

2.2.9.7.1 accept-ra

Allows the device, in host mode, to reject the RAs received. Since this command is enabled by default, you need to place the **no** command in front to modify device behavior.

Syntax:

```
no ipv6 nd ra accept-ra
```

Example:

```
ethernet0/0 config>no ipv6 nd ra accept-ra
```

2.2.9.7.2 accept-ra forced

Allows the device to function as a host, even though it isn't, and accept the RAs received from the routers.

Syntax:

```
ipv6 nd ra accept-ra [forced]
```

forced:	Accepts the RAs, even if the device or interface where this option is configured is not a host.

Example:

```
ethernet0/0 config>ipv6 nd ra accept-ra ?
  forced    Accept Router Advertisements even if mode is not host
  <cr>
ethernet0/0 config>ipv6 nd ra accept-ra forced
```

2.2.9.7.3 default-router

With the **"ipv6 nd ra default-router"** command, the device, when in host mode, selects a default router from among the received RAs. Since this command is enabled by default, you need to place the **no** command in front to modify device behavior.

Syntax:

```
[no] ipv6 nd ra default-router
```

Example:

```
ethernet0/0 config>no ipv6 nd ra default-router
```

Thus, when the host receives an RA, it is not selected as default router.

2.2.9.7.4 hop-limit unspecified

Changes the *hop-limit* value sent in RA messages to "unspecified" (0). The "unspecified" value tells the host to ignore this value and use the previous one (which may be different from the default).

Syntax:

```
ipv6 nd ra hop-limit unspecified
```

Example:

```
ethernet0/0 config>ipv6 nd ra hop-limit unspecified
```

While this command is enabled, the numeric *hop-limit* value assigned through the *hop-limit* command (section 2.6) is ignored.

2.2.9.7.5 interval

Interval between RA message transmissions. You can define a maximum interval or a time range within which they are sent. The minimum time between RA message transmissions is 3 seconds.

Syntax:

```
ipv6 nd ra interval <max-time> [<min-time>]
```

max-time:	Maximum interval between RA message transmissions. The valid range for this field is 4 to 1800 seconds. The default value is 600 seconds.
min-time:	Minimum interval between RA message transmissions. The valid range for this field is 3 to ¾ of the maximum time value configured. If no value is configured, it takes 1/3 the value of the maximum time interval between the transmission of unsolicited RAs.

Example:

```
ethernet0/0 config>ipv6 nd ra interval 100 ?
  <3..75>   Minimum time between router advertisements
```

```
  <cr>
ethernet0/0 config>ipv6 nd ra interval 100 20
```

With this configuration, RA messages are sent within a random interval of 100 to 20 seconds.

2.2.9.7.6 lifetime

The *lifetime* value is included in all RA messages sent from the interface.

When an RA is received, the *lifetime* value indicates for how long the router can be used as a default router on the selected interface. A 0 value indicates that it should not be used as the default router on that interface. A value other than 0 indicates that it can be used as the default router on that interface during the time marked by *lifetime*. This value cannot be less than the aforementioned *RA interval.*

Syntax:

```
ipv6 nd ra lifetime <value>
```

value:	Time, in seconds, that the router can be used as the default router. Valid values range from 0 to 9000 seconds. A 0 value indicates that it cannot be used as the default router, in which case it must be removed and a new default router selected (initiating *Router Discovery* if necessary). If no value is configured, it takes triple the value of the maximum time interval between the transmission of unsolicited RAs.

Example:

```
ethernet0/0 config>ipv6 nd ra lifetime ?
  <0..9000>    RA Lifetime (seconds)
ethernet0/0 config>ipv6 nd ra lifetime 500
```

2.2.9.7.7 mtu suppress

Modifies the default behavior of the device, when it is functioning as a router, so that it does not send the MTU option in the RAs.

Syntax:

```
ipv6 nd ra mtu suppress
```

Example:

```
ethernet0/0 config>ipv6 nd ra mtu suppress
```

2.2.9.7.8 prefix-info

With the **"ipv6 nd ra prefix-info"** command, the device, when in host mode, learns the *Prefix information* option in the received RAs. Since this command is enabled by default, you need to place the **no** command in front to modify device behavior.

Syntax:

```
[no] ipv6 nd ra prefix-info
```

Example:

```
ethernet0/0 config>no ipv6 nd ra prefix-info
```

Thus, when the host receives an RA, it ignores the *Prefix information* .

2.2.9.7.9 source-ll-address

Modifies the default behavior of the device, when it is functioning as a router, so that it suppresses the *Source link-layer address* option of the RAs.

Syntax:

```
ipv6 nd ra source-ll-address suppress
```

Example:

```
ethernet0/0 config>no ipv6 nd ra source-ll-address ?
  suppress    Suppress IPv6 RA Source Link Local Address Option
ethernet0/0 config>no ipv6 nd ra source-ll-address suppress
```

2.2.9.7.10 suppress

Inhibits the transmission of automatic RA messages when the device is functioning as a router. The RAs sent in response to an RS are not suppressed.

Syntax:

```
ipv6 nd ra suppress
```

Example:

```
ethernet0/0 config>ipv6 nd ra suppress
```

2.2.9.8 rdnss

The **"ipv6 nd rdnss"** command is used to attach the RDNSS (*Recursive DNS Server*) option to the RA message with one or more recursive DNS server addresses. Each RDNSS option can take up to three RDNSS addresses with the same *lifetime*. RDNSS addresses with different *lifetimes* are separated into different options.

Syntax:

```
ipv6 nd rdnss <ipv6-address> [<lifetime> | infinite]
```

ipv6-address	Recursive DNS server address.
lifetime	Maximum time, in seconds, during which the RDNSS address can be used for name resolution. If no value is configured, it takes double the value of the maximum time interval between the transmission of unsolicited RAs.
infinite	The RDNSS address can be used indefinitely.

Example:

```
ethernet0/0 config>ipv6 nd rdnss ?
  <a::b>    Ipv6 address
ethernet0/0 config>ipv6 nd rdnss 2001:db8::1234 ?
  <0..4294967294>    RDNSS Lifetime (secs)
  infinite           Infinite RDNSS Lifetime
  <cr>
ethernet0/0 config>ipv6 nd rdnss 2001:db8::1234 300
ethernet0/0 config>
```

2.2.9.9 reachable-time

Configures for how long an IPv6 node is considered reachable since the last reachability confirmation. This time allows the device to detect neighbors that are unreachable. The information configured with this command is included in the RAs sent from the selected interface.

Syntax:

```
ipv6 nd reachable-time <value>
```

value:	How long, in milliseconds, a node can be considered reachable. Valid values range between 0 and 3600000 milliseconds. Default values: 0 milliseconds (unspecified) is advertised in RAs, and the value 30000 (30 seconds) is used for the neighbor discovery activity of the router itself.

Example:

```
ethernet0/0 config>ipv6 nd reachable-time ?
  <0..3600000>    Reachability time in milliseconds
ethernet0/0 config>ipv6 nd reachable-time 200
```

In IPv6 interface monitoring (**"list interface"** command) you can see the *reachable time* value configured in the following field:

```
ND advertised reachable time is 200 milliseconds
```

2.2.9.10 router-preference

Configures the Default Router Preference (DRP) for a specific interface. This information is sent in RA messages to inform the hosts of the router preference. A router preference of *Medium* is sent by default.

Syntax:

```
ipv6 nd router-preference {High | Medium | Low}
```

High:	High default router preference on an interface.
Medium:	Medium default router preference on an interface.
Low:	Low default router preference on an interface.

Example:

```
ethernet0/0 config>ipv6 nd router-preference ?
  High      High default router preference
  Medium    Medium default router preference
  Low       Low default router preference
ethernet0/0 config>ipv6 nd router-preference High
```

In IPv6 interface monitoring (**"list interface"** command) you can see the *router preference* value configured in the following field:

```
ND advertised default router preference is High
```

2.2.9.11 rs

The **"ipv6 nd rs"** subcommand, in turn, contains the following options related to *Router Solicitation* (RS):

```
ethernet0/0 config>ipv6 nd rs ?
  attempts    Set number of Router Solicitations attempts
  delay       Set Maximum delay time of sending RS after the interface goes UP
  interval    Set Router Solicitation Interval
```

2.2.9.11.1 attempts

Configures the number of times RS is sent during *Router Discovery*.

Syntax:

```
ipv6 nd rs attempts <value>
```

value:	Number of times the RS is sent. Valid values range from 1 to 10. The default value is 3.

Example:

```
ethernet0/0 config>ipv6 nd rs attempts ?
  <1..10>    Number of attempts
ethernet0/0 config>ipv6 nd rs attempts 5
```

2.2.9.11.2 delay

The **"ipv6 nd rs delay"** command is used to configure how long, in milliseconds, it takes for an RS message to be sent from the moment the interface is up.

Syntax:

```
ipv6 nd rs delay <value>
```

value:	Delay, in milliseconds, from the moment the interface is up until it sends the first RS. Valid values range from 0 to 2000 milliseconds. The default value, if none has been configured, is 1 second.

Example:

```
ethernet0/0 config>ipv6 nd rs delay ?
  <0..2000>    Delay (msec)
ethernet0/0 config>ipv6 nd rs delay 500
```

2.2.9.11.3 interval

Configures the time interval between RS transmissions.

Syntax:

```
ipv6 nd rs interval <value>
```

value:	How long, in seconds, it takes for a second RS message to be sent after the first one. Valid values range from 1 to 10 seconds. The default value is 4.

Example:

```
ethernet0/0 config>ipv6 nd rs interval ?
  <1..10>    Interval (seconds)
ethernet0/0 config>ipv6 nd rs interval 8
```

2.2.9.12 target-ll-option

Forces the *Target Link Layer* option to be sent in an NA. The *Target Link Layer* option is only sent when the NA is sent in response to a multicast NA. This way, endless requests by the soliciting node are avoided (since the latter does not have an entry in the neighbor cache with the physical address of the solicited node).

Via this command, the option is sent in all NAs.

Syntax:

```
ipv6 nd target-ll-option
```

Example:

```
ethernet0/0 config>ipv6 nd target-ll-option
```

2.2.10 ospfv3

The **"ipv6 ospfv3"** command configures OSPFv3 protocol parameters on the selected interface. For further information, please see the Teldat- **Dm816-I OSPFv3 Protocol** manual.

2.2.11 redirects

The **"ipv6 redirects"** command enables the generation of ICMP *Redirects*. These messages give hosts a better next hop to the destination. This command is enabled by default. To prevent these messages from being sent, you need to negate the command as shown below:

Syntax:

```
no ipv6 redirects
```

Example:

```
ethernet0/0 config>no ipv6 redirects
```

2.2.12 ripng

The **"ipv6 ripng"** command configures RIPng protocol parameters on the selected interface. For further information, please see the Teldat- **Dm814-I RIPng Protocol** manual.

2.2.13 unreachables

The **"ipv6 unreachables"** command is used to allow the selected interface to generate ICMP *Unreachable* messages. This command is enabled by default. To avoid these messages from being sent, you need to negate the command as shown below:

Syntax:

```
no ipv6 unreachables
```

Example:

```
ethernet0/0 config>no ipv6 unreachables
```

2.2.14 vrrp

The "**vrrp ipv6**" command allows you to configure one or more VRRP Virtual Routers in IPv6 on the selected interface. For more information, please see the "**Dm821 VRRPv3 Protocol for IPv6**" manual.

Command history:

Release	Modification
11.00.04	The VRRPv3 protocol for IPv6 was introduced as of version 11.00.04.
11.00.03.01.02	The VRRPv3 protocol for IPv6 was introduced as of version 11.00.03.01.02.
11.01.00	The VRRPv3 protocol for IPv6 was introduced as of version 11.01.00.

Chapter 3 Monitoring

3.1 IPv6 Monitoring Commands

This chapter describes the IPv6 monitoring commands. To access the protocol's monitoring environment, enter the following sequence of commands:

```
*monitor
Console Operator
+protocol ipv6
-- IPv6 protocol monitor --
IPv6+
```

This menu contains the following options:

Command	Function
? (HELP)	Lists the commands or their options.
CLEAR	Removes data from the IPv6 protocol.
ECHO	Multiprotocol echo for IPv6.
LIST	Lists the IPv6 elements.
PING	Ping for IPv6.
TRACEROUTE	Traceroute for IPv6.
VRF	Enters the IPv6 monitoring menu for a VPN Routing/Forwarding instance.
VRRP	Provides access to the VRRP protocol monitoring menus.
EXIT	Returns to the previous menu.

3.1.1 ? (HELP)

Lists the commands found at the *prompt* you are using. You can also enter it after a specific command to list the available options:

Syntax:

```
IPv6+?
```

Example:

```
IPv6+?
 clear         Removes data from the IPv6 protocol
  echo          IPv6 multiprotocol echo
  list          Show elements
  ping          IPv6 echo
  traceroute    Complete path to a particular IPv6 destination
  vrf           IPv6 in a VPN Routing/Forwarding instance
  vrrp          VRRP monitoring
  exit
IPv6+
```

Command history:

Release	Modification
11.00.04, 11.00.03.01.02, 11.01.00	The VRRPv3 protocol for IPv6 was introduced and the **vrrp** command was added as a result.
11.01.06	The **clear** command was introduced as of version 11.01.06.
11.01.07	The **traceroute** command was introduced as of version 11.01.07.
11.01.09	The **vrf** command was introduced as of version 11.01.09.

3.1.2 clear

Removes stored data from the IPv6 protocol. The available options for this command are:

Syntax:

```
IPv6+clear ?
  neighbor    Remove detected neighbors
IPv6+
```

Command history:

Release	Modification
11.01.06	The **clear** command was introduced as of version 11.01.06.

3.1.2.1 neighbor

Opens a list of suboptions the user can select to delete every neighbor discovered by the interface specified (*interface*), all neighbors detected (*all*) or a specific neighbor (*host*).

Syntax:

```
IPv6+clear neighbor ?
  all          Remove all detected neighbors
  host         Remove a specific neighbor
  interface    Remove all neighbors discovered by the interface specified
IPv6+
```

Command history:

Release	Modification
11.01.06	The **neighbor** option was added to the **clear** command as of version 11.01.06.

3.1.2.1.1 interface

Removes every neighbor detected by the interface specified.

Syntax:

```
IPv6+clear neighbor interface <interface name>
```

Example:

```
IPv6+list neighbors

IPv6-Address                                Link-Layer-Addr   State    Age Interface
fe80::2a0:26ff:fe6c:5c                      0:a0:26:6c:0:5c   STALE      0 ethernet0/0
fe80::2a0:26ff:fec2:f636                    0:a0:26:c2:f6:36  STALE      1 ethernet0/1
fe80::21b:54ff:fea9:4761                    0:1b:54:a9:47:61  STALE      0 ethernet0/0
IPv6+
IPv6+clear neighbor interface ethernet0/0
```

Command history:

Release	Modification
11.01.06	The **interface** suboption was added to the **neighbor** option under the **clear** command as of version 11.01.06.

3.1.2.1.2 host

Removes a neighbor identified by its IPv6 address.

Syntax:

```
IPv6+clear neighbor host <IPv6 address>
```

Example:

```
IPv6+list neighbors

IPv6-Address                                Link-Layer-Addr   State    Age Interface
fe80::2a0:26ff:fe6c:5c                      0:a0:26:6c:0:5c   STALE      0 ethernet0/0
fe80::2a0:26ff:fec2:f636                    0:a0:26:c2:f6:36  STALE      0 ethernet0/0
IPv6+
```

```
IPv6+clear neighbor host fe80::2a0:26ff:fec2:f636
```

Command history:

Release	Modification
11.01.06	The **host** suboption was added to the **neighbor** option under the **clear** command as of version 11.01.06.

3.1.2.1.3 all

Removes every neighbor registered in the system.

Syntax:

```
IPv6+clear neighbor all
```

Example:

```
IPv6+list neighbors

IPv6-Address                                Link-Layer-Addr   State    Age Interface
fe80::2a0:26ff:fe6c:5c                      0:a0:26:6c:0:5c   STALE      0 ethernet0/0
fe80::2a0:26ff:fec2:f636                    0:a0:26:c2:f6:36  STALE      0 ethernet0/0
IPv6+
IPv6+clear neighbor all
```

Command history:

Release	Modification
11.01.06	The **all** suboption was added to the **neighbor** option under the **clear** command as of version 11.01.06.

3.1.3 echo

Tests TCP/UDP sockets. It connects through sockets to an external server and allows you to choose which test to perform (i.e., TCP or UDP) using the IPv6 address of the external server and, optionally, the output interface.

Syntax:

```
IPv6+echo ?
  tcp    Perform TCP echo test
  udp    Perform UDP echo test
IPv6+echo tcp ?
  <a::b>    Ipv6 address
IPv6+echo tcp 2001:db8:1111::1 ?
  interface    Output interface
  <cr>
IPv6+echo tcp 2001:db8:1111::1 interface ?
  <interface>    Interface name
IPv6+
```

3.1.4 list

Allows you to list the elements associated with IPv6 monitoring. These elements can be grouped into addresses, prefixes, routes and tunnels.

Syntax:

```
IPv6+list ?
  address           List IPv6 address(es)
  fib               Show IPv6 fib route table entries
  general-prefix    List of general-prefix
  interface         List interface status
  neighbors         Show IPv6 neighbor cache entries
  prefix            List prefixes announced in router advertisements
  rdnss             List RDNSS address(es) announced in router advertisements
  route             Show IPv6 route table entries
  tunnel            Show configured IPv6 tunnels
```

3.1.4.1 list address

Lists IPv6 addresses linked to all interfaces or to one in particular.

Syntax:

```
list address [<interface-name>]
```

An interface is defined by its type, followed by an identifier if there is more than one interface. If IPv6 is enabled on an interface, this is shown next to each interface. In the event that it is enabled, the link-local address (should one exist), the global unicast addresses and the multicast groups that the interface has joined (*Joined group address(es)*) are also displayed.

Example:

```
IPv6+list address interface ethernet0/0
Interface ethernet0/0:
----------------------------
IPv6 is Enabled
Link-local address is:  fe80::2a0:26ff:fe44:0  [PERM]

Global unicast address(es):
  111:111::2a0:26ff:fe44:0/64 ra-auto [PERM/UP]
  444::2a0:26ff:fe44:0/64 ra-auto [UP] valid lifetime 2591965s, preferred lifetime 604765s
  9999:9999::888/64 cfg [PERM/UP]

Joined group address(es):
  ff02::1:ff44:0
  ff02::1:ff00:888
  ff02::1
IPv6+
```

The method used to configure each global unicast address is shown:

- Configured by the user: **cfg**.
- Configured by the user using EUI: **cfg-eui**.
- Configured by the user using a general-prefix: **gen-pref**.
- Automatically configured from a prefix received in an RA: **ra-auto**.

If a unicast global address is not permanent, the valid and preferred lifetime associated with the address is displayed in seconds. In addition, each has an associated set of flags that determine the type and state of the address:

- **ANY**. The address falls under the *anycast* category.
- **UP**. The address is active.
- **DOWN**. The address is not active.
- **OVR**. There is a conflict because another address is using the same prefix on the same interface or on a different one.
- **UNA**. The virtual address is not activated.
- **OOD**. DAD (*Overly Optimistic Duplicated Address Detection*) will not be performed for this address. This flag applies to virtual addresses.
- **DUP**. The address is a duplicate address as determined by DAD.
- **TEN**. The address is in a *tentative* state (it has not been possible to confirm that the address is unique within the link).
- **PERM**. This address is associated with infinite *preferred* and *valid lifetimes*.
- **DEPR**. The preferred lifetime has expired or, in other words, its value is 0.

3.1.4.2 list fib

Displays the content of the FIB routing table. For further information on the FIB table, please see the Teldat- **Dm807-I IPv6 Static Routing** manual.

Syntax:

```
list fib
```

Example:

```
IPv6+list fib
```

```
Codes: R - Receive, T - Transmit, U - Unreachable, * - Active.

R*  2001:db8:1111::3/128  via Interface internal
T*  2001:db8:1111::/64  via Interface ethernet0/0
T*  2001:db8:2222::/64  via Interface ethernet0/0
R*  2001:db8:2222:1234::1/128  via Interface internal
T*  2001:db8:2222:1234::/64  via Interface ethernet0/0
T*  2001:db8:3333::/64  via Interface ethernet0/0
R*  fe80::2a0:26ff:fe44:0/128  via Interface internal
T*  fe80::/64  via Interface ethernet0/1
T*  fe80::/64  via Interface ethernet0/0
R*  ff00::/8  via Interface ethernet0/1
R*  ff00::/8  via Interface ethernet0/0
U   ::/0  via Interface internal
```

3.1.4.3 list general-prefix

Lists the general prefixes that can be found in the device. You can filter by interface name, by prefix name or by type.

Syntax:

```
IPv6+list general-prefix [ [interface <interface>] | [name <name>] | [type {6rd | 6to4 | dhcp | prefix}]]
```

interface:	Interface you wish to filter through.
name:	Name of the *general-prefix* you want to use for filtering purposes.
type:	*general prefix* user type.

The set of addresses created from the selected prefix, if there are any, and the interface associated with it are shown for each existing *general prefix*. There may also be information on the *general prefix,* or on the address created from it, right after its list entry. The information is presented as follows:

- **6rd.** This is a *general prefix* created to configure IPv6 addresses for a 6rd domain.
- **6to4.** This is a *general prefix* created to configure IPv6 addresses for a 6to4 domain.
- **dhcp.** This is a *general prefix* obtained through DHCP.
- **anycast**. This is an anycast address.
- **eui-64**. The interface identifier has been created through the eui-64 mechanism (the identifier's seventh bit is inverted and the *fffe* value is inserted between the third and fourth byte of the MAC address).

Example:

```
IPv6+list general-prefix

General-prefix name: teldat-pref-3
  General-prefix                 2001:db8:2222::/48
  Address on ethernet0/0         ::2222:0:0:0:1/64
  Address on ethernet0/0         ::3333:0:0:0:1/64 (anycast)
  Address on ethernet0/0         ::4444:1:ff:fe00:1/64 (eui-64)

General-prefix name: teldat-pref-1
  General-prefix                 2001:db8:dec8::/48 (6to4)
  Address on tnip2               ::300/64

General-prefix name: teldat-pref-2
  General-prefix                 2001:db8:c800::/40 (6rd)
  Address on tnip1               ::200/64

General-prefix name: teldat-pref-4
  General-prefix                 2001:db8:ffff::/48 (dhcp) valid lifetime 2591856s,
                                        preferred lifetime 604656s
  Address on ethernet0/1         ::2/64
```

3.1.4.4 list interface

Lists the IPv6 addresses linked to all interfaces or to one in particular, as well as any additional information.

Syntax:

```
IPv6+list interface [<interface>]
```

This command is very similar to the **list address** command described above. Thus, many values shown have the same meaning and are to be interpreted in the same way. The most significant difference is the inclusion of additional information that increases the data available on each interface. Information can be grouped as follows.

- MTU size in bytes.
- ICMP protocol behavior.
- *Neighbor Discovery* protocol behavior.
- Default router, when the device or interface acts as host.
- Existing prefixes, together with their valid and preferred lifetimes.
- Recursive DNS server addresses that are sent in the RA RDNSS option, if it exists.

Example:

```
IPv6+list interface ethernet0/0

Interface ethernet0/0:
------------------------------
IPv6 is Enabled
Link-local address is:  fe80::2a0:26ff:fe44:0 [PERM]

Global unicast address(es):
  2001:db8:1111::3/64 cfg [PERM/UP]
  2001:db8:2222:1234::1/64 gen-pref [PERM/UP]

Joined group address(es):
  ff02::1:ff44:0
  ff02::1:ff00:1
  ff02::1:ff00:3
  ff02::1

MTU is 1500 bytes
ICMP error messages limited to one every 1000 milliseconds
ICMP redirects are enabled
ICMP unreachables are sent
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds
There is no default router

IPv6 Prefix Advertisements ethernet0/0
Codes: A - Address, P - Prefix-Advertisement, N - Not advertised,
       [L] - On-link, [A] - Autonomous
P  2001:db8:2222::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
PN  2001:db8:3333::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
A  2001:db8:2222:1234::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
A  2001:db8:1111::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800

IPv6 RDNSS Address(es) Advertised on ethernet0/0
  2222::2222  Lifetime: 1234
```

3.1.4.5 list neighbors

Displays the neighbor cache state.

Syntax:

```
IPv6+list neighbors [<interface>]
```

Using this command, you can see the IPv6 address of known neighbors, their physical address, state, the time that has gone by since the entry was last used and their associated interface.

Example:

```
IPv6+list neighbors

IPv6 Address               Link-Layer-Addr  State  Age Interface
```

```
fe80::215:63ff:feee:2120    0:15:63:66:21:20 STALE    1 ethernet0/0
999:999::1234               0:15:63:ee:21:20 REACH    0 ethernet0/0
```

Entries in the neighbor cache can have the following states:

- **Incomplete**. Address resolution is in progress.
- **Reachable**. The neighbor could be recently reached.
- **Stale**. There has been no recent response from the neighbor.
- **Delay**. There has been no response from the neighbor to which traffic was recently sent.
- **Probe.** Neighbor reachability is uncertain.

3.1.4.6 list prefix

Displays information on the prefixes that are advertised or that would be advertised through *Router Advertisement*.

Syntax:

```
IPv6+list prefix [<interface>]
```

Prefixes are organized at their associated interface. They may have been created statically (through the **general-prefix** and **address** commands) or obtained through *Router Advertisement* or *Prefix Delegation*.

The broad meaning of each of the *flags* accompanying the prefixes is listed below.

- A (*Address*). Prefix associated with an address.
- P *(Prefix-Advertisement).* Prefix to be announced through *Router Advertisement*.
- N *(Not advertised).* Indicates the prefix will not be announced.

The following flags are advertised with the prefix in an RA.

- [L] (*On-link*). Prefix that can be used and is added to the routing table.
- [A] (*Autonomous*). Prefix that can be used for autonomous address configuration or, in other words, *stateless* autoconfiguration.

Lastly, the *valid* and *preferred lifetimes* refer to the lifetimes that will be sent in *Router Advertisements.*

Example:

```
IPv6+list prefix

IPv6 Prefix Advertisements ethernet0/0
Codes: A - Address, P - Prefix-Advertisement, N - Not advertised,
       [L] - On-link, [A] - Autonomous
P  2001:db8:2222::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
PN  2001:db8:3333::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
A  2001:db8:2222:1234::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
A  2001:db8:1111::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
IPv6+
```

3.1.4.7 list rdnss

Lists the recursive DNS server addresses configured.

Syntax:

```
list rdnss [interface <interface-name>]
```

interface-name:	Name of the interface that filters the recursive DNS server list.

Example:

```
IPv6+list rdnss

IPv6 RDNSS Address(es) Advertised on ethernet0/0
  4444::4444  Lifetime: 60 (default)
  4444::6666  Lifetime: 1234
  4444::5555  Lifetime: infinite
IPv6+
```

3.1.4.8 list route

Displays the entries in the *RIB routing table*. For further information on the RIB table, please see the Teldat- **Dm807-I IPv6 Static Routing** manual.

Syntax:

```
list route [<a::b>] [<a::b/l>] [bgp] [connected] [ospfv3] [ripng] [static] [summary]
```

Example:

```
IPv6+list route
Codes: C - Connected, S - Static, R - RIPng, O - OSPFv3 Intra, OI - OSPFv3 Inter,
       OE1 - OSPFv3 ext 1, OE2 - OSPFv3 ext 2, B - BGP, * - Active

C*  2001:db8:1111::/64 [0/1] is directly connected, ethernet0/0
C*  2001:db8:2222::/64 [0/1] is directly connected, ethernet0/0
C*  2001:db8:2222:1234::/64 [0/1] is directly connected, ethernet0/0
C*  2001:db8:3333::/64 [0/1] is directly connected, ethernet0/0
```

Command history:

Release	Modification
11.01.09	The *list route* command options were introduced as of version 11.01.09

3.1.4.9 list tunnel

Displays the IPv6 tunnels configured on the device.

Syntax:

```
list tunnel [6rd [{destination <delegated-prefix>
<interface-name>}|{interface <interface-name>}|{prefix <ipv4> <interface-name>}]] |
[6to4 [{destination <delegated-prefix>}|{interface <interface-name>}|{prefix <ipv4>}]] |
[interface <interface-name>] | [manual [interface <interface-name>]]
```

Example:

```
IPv6+list tunnel
Interface tunnel tnip1: mode Manual 6to4
Tunnel Source: 10.10.10.10
Tunnel Destination: 10.10.10.11
```

For further information, please see the Teldat- **Dm810-I IPv6 Tunnel over IPv4** manual.

3.1.5 ping

Allows you to perform an IPv6 ping to another device. The syntax and options are as follows:

Syntax:

```
IPv6+ping ?
  <a::b%z>    Ipv6 address
IPv6+ping 2001:db8:1111::1%ether
IPv6+ping 2001:db8:1111::1 ?
  data-bytes        Number of data bytes
  interval-pings    Time between pings(>=10ms)
  num-pings         Number of pings (default: 0 = Stop when a key is pressed)
  source            IP source
  timeout           Time out(>=10ms)
  <cr>
```

<a::b%z>:	Pinged address. In the event that the *Echo Request* packet destination is a *link-local* address, you should specify the packet output interface through the '%' symbol followed by the interface.
data-bytes:	Number of packet data bytes.
interval-pings:	Time between *Echo Request* messages.
num-pings:	Number of *Echo Request* messages you want to send.

source:	Address you want as source in the *Echo Request.*
timeout:	Maximum wait time.

Example:

```
IPv6+ping fe80::2a0:26ff:fe4e:5a20%ethernet0/0 data-bytes 2048
PING (fe80::2a0:26ff:fe4e:5a20) 2048 data bytes
2056 bytes from fe80::2a0:26ff:fe4e:5a20: icmp_seq=1 time=8.00 ms
2056 bytes from fe80::2a0:26ff:fe4e:5a20: icmp_seq=2 time=5.00 ms
2056 bytes from fe80::2a0:26ff:fe4e:5a20: icmp_seq=3 time=5.00 ms

--- fe80::2a0:26ff:fe4e:5a20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 5.000/6.000/8.000/1.414 ms
IPv6+
```

3.1.6 traceroute

Displays the entire path to a given destination, hop by hop. For each successive hop, **traceroute** sends out various packets and displays the IPv6 address of the responding router, together with the round trip time associated with the response. If a particular packet receives no response, an asterisk appears.

This command is executed whenever the destination is reached, an ICMPv6 Destination Unreachable is received, or the path length surpasses the maximum number of hops specified by the user.

The following parameter is requested:

IPv6 destination: IPv6 address of the router whose path you want to see. This is specified through the IPv6 address.

This parameter is the only one needed to execute this command. A series of options that take default values (unless modified) will then appear. To accept default values for the remaining options, simply press the CR (carriage return) key. Said options are:

- *Protocol* (protocol): Probe packets protocol: UDP or ICMPv6. Default is UDP.
- *Beginning destination UDP port* (udp-port): This parameter is available only if the user selected UDP. It displays the destination port in the UDP packet sent, which increases for each probe. Default is 33434.
- *IPv6 source* (source): Packet output. By default, the router selects the output interface (logical) source address.
- *Seconds to wait for response* (timeout): Waiting time, in seconds, for a response on the probe packet sent. Default is 3.
- *Probes at each TTL* (probes): Number of probes to be sent by each TTL. Default is 3.
- *Maximum Time To Live* (max-ttl): Maximum number of hops. Default is 30.
- *Verbose* (verbose)*:* Type of trace view. If you select verbose, you can see on the left the distance (in hops) to the router. The consecutive lines show the results for each probe for this number of hops. It also shows the IPv6 address of the responding router. Traditional viewing shows a single line with the results of all polls, executed with the same TTL, with just one IPv6 address from one of the responding routers. Default is deactivated.

When a probe receives an unexpected result, several indications can be viewed:

!N references an ICMPv6 Destination Unreachable (net unreachable) packet has been received.

!H references an ICMPv6 Destination Unreachable (host unreachable) packet has been received.

!S references an ICMPv6 Destination Unreachable (admin prohibited) packet has been received.

If the probe packets are ICMPv6, the expected response is an ICMPv6 *Echo Reply* packet. When sending UDP packets to a remote port, the expected response is ICPv6 *Destination unreachable (Port unreachable)*.

Syntax:

```
IPv6+traceroute <destination_IPv6_addr>[protocol udp|icmp]
[udp-port <port_num>] [source <source_IPv6_addr>] [timeout <timeout_s>] [probes <num_probes>]
[max-ttl <maximum_ttl>] [verbose]
```

Example:

```
IPv6+traceroute 2001:db8:4444::46 protocol icmp timeout 2 max-ttl 15 verbose
Press any key to abort.

 Tracing the route to: 2001:db8:4444::46 from 2001:db8:8888::2
 Options: ICMP, 15 hops max, 20 byte packets
```

```
1
  Probe:   1, Time  1 ms, IPv6:  2001:db8:8888::1
  Probe:   2, Time  1 ms, IPv6:  2001:db8:8888::1
  Probe:   3, Time  2 ms, IPv6:  2001:db8:8888::1
2
  Probe:   1, Time  1 ms, IPv6:  2001:db8:2222::2
  Probe:   2, Time  1 ms, IPv6:  2001:db8:2222::2
  Probe:   3, Time  0 ms, IPv6:  2001:db8:2222::2
3
  Probe:   1, Time  1 ms, IPv6:  2001:db8:3333::1
  Probe:   2, Time  1 ms, IPv6:  2001:db8:3333::1
  Probe:   3, Time  1 ms, IPv6:  2001:db8:3333::1
4
  Probe:   1, Time  2 ms, IPv6:  2001:db8:4444::46
  Probe:   2, Time  2 ms, IPv6:  2001:db8:4444::46
  Probe:   3, Time  1 ms, IPv6:  2001:db8:4444::46
Trace complete.
```

The meaning of each field is:

Press any key to abort:	If the user presses a key while the **traceroute** command is being executed, said process is aborted.
Tracing the route to:	Displays the IPv6 destination address, the protocol used to send packets, the maximum number of hops and the size of the packet sent.
1:	First trace from the destination.
Probe:	Probe for a given TTL. It displays the response time and the IPv6 address of the responding device. In this case, three probe packets are sent for each hop.
Trace complete:	The trace has been completed.

In this case, the **traceroute** command is used when only the destination is entered (through its IPv6 address). Here, all configurable parameters take their default values.

Example:

```
IPv6+traceroute 2001:db8:4444::46
Press any key to abort.

 Tracing the route to: 2001:db8:4444::46 from 2001:db8:8888::2
 Options: UDP, 30 hops max, 28 byte packets

 1   3 ms  1 ms  1 ms 2001:db8:8888::1
 2   1 ms  1 ms  1 ms 2001:db8:2222::2
 3   3 ms  1 ms  2 ms 2001:db8:3333::1
 4   3 ms  2 ms  2 ms 2001:db8:4444::46
 Trace complete.
```

The meaning of each field is:

1:	First trace to display the destination NSAP, as well as the time necessary to reach this. Three probes are sent (packet is sent 3 times).
** * *:*	Despite this option not being illustrated in the example, the appearance of these asterisks indicates the router is waiting for a response from the destination (still waiting).

Command history:

Release	Modification
11.01.07	The *traceroute* command was introduced as of version 11.01.07.

3.1.7 vrf

Accesses a specified VRF IPv6 monitoring menu. On accessing a VRF IPv6 monitoring menu, the prompt changes to **IPv6 vrf+**. Use the **exit** command to return to the main VRF IPv6 monitoring menu.

Example:

```
IPv6+vrf v1
```

```
-- IPv6 protocol monitor for a VRF --

IPv6 vrf+
```

Command history:

Release	Modification
11.01.09	The VRF command was introduced as of version 11.01.09.

3.1.8 vrrp

Accesses the monitoring menus of the VRRP protocol for IPv6. For more information, please see the Teldat- **Dm821 VRRPv3 protocol for IPv6** manual.

Command history:

Release	Modification
11.00.04	The VRRPv3 protocol for IPv6 was introduced.
11.00.03.01.02	The VRRPv3 protocol for IPv6 was introduced.
11.01.00	The VRRPv3 protocol for IPv6 was introduced.

3.1.9 exit

Returns to the previous *prompt* level.

Syntax:

```
exit
```

Example:

```
IPv6+exit
+
```

Chapter 4 Events

4.1 Monitoring through Events

You can view information relevant to a subsystem or to a specific event through the events logging system.

Subsystems related to IPv6 addressing are as follows:

- *ICMP6:* Displays events related to the ICMPv6 protocol.
- *IP6:* Displays events related to the IPv6 protocol.
- *ND:* Displays events related to the *Neighbor Discovery* protocol.
- *NEIGH:* Displays events related to the neighbor cache table status.

To enable/disable events related to *addressing*, follow the sequence of steps in the *static* or *dynamic configuration* section:

```
{enable}|{disable} trace subsystem {icmp6}|{ip6}|{nd}|{neigh} [<level>]
```

Where:

- *level* indicates the threshold from which all events relating to the specified subsystem are shown. Some of the most common register levels are *standard*, *error*, *info* or *all*.

If you wish to enable/disable an event corresponding to a specific subsystem, the command syntax is as follows:

```
{enable}|{disable} trace event <event>
```

Where:

- *event*is the identifier for the event in question. This syntax is defined by the following expression:

```
{icmp6}|{nd}|{ip6}.<number>
```

Where:

- *number* is the event index in the specified subsystem.

If you require further information on one of the above subsystems, please see the relevant manual.

Chapter 5 Example

5.1 IPv6: Basic Configuration Example

This section aims to show, by way of example, a potential IPv6 network addressing scenario. This scenario includes two devices connected to the *Local Area Network* through their *Ethernet* interface. In this configuration, one of the devices acts as *router* while the other acts as *host*.

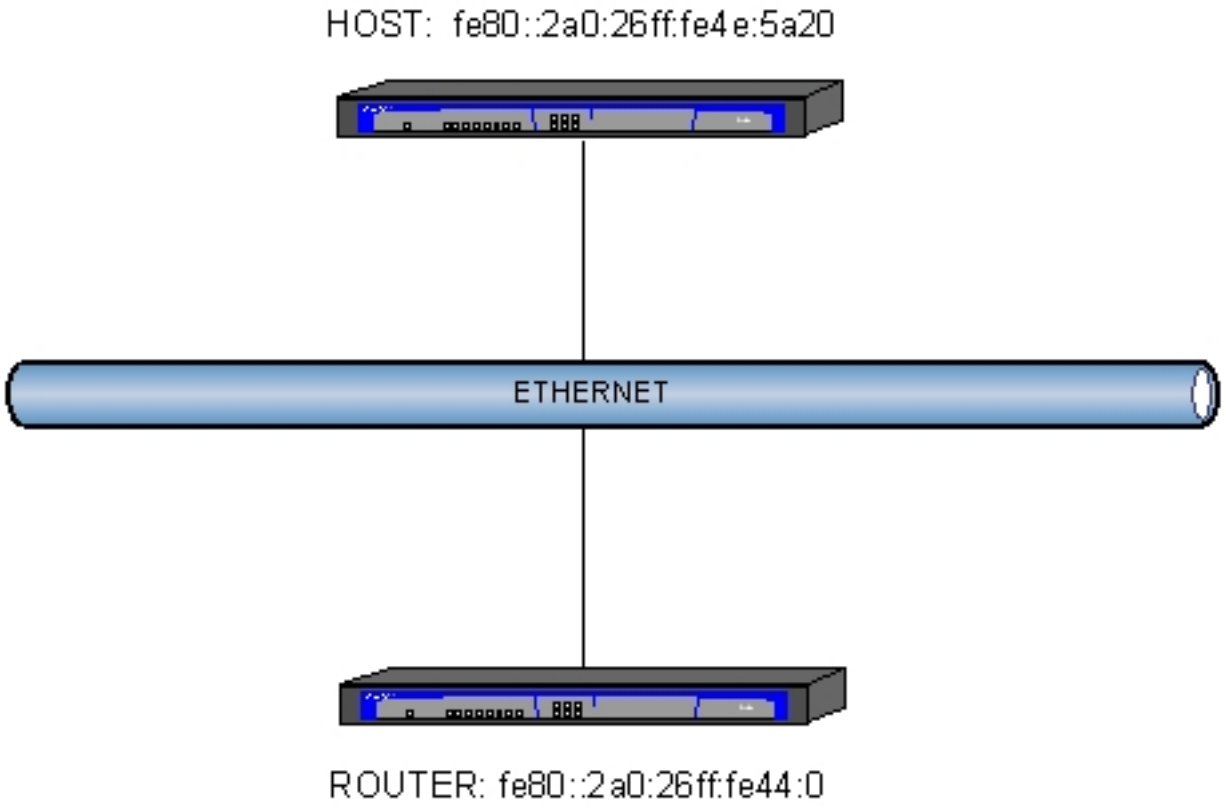

Fig. 7: IPv6 basic configuration example: Scenario.

Configurations:

Here we are going to display each of the two configurations used in the devices.

Router

Forwarding is the first thing configured on the router. To do this, find the “unicast-routing” command in the “protocol ipv6” menu. In addition, a *general prefix* is configured for global address configuration:

```
Config>protocol ipv6

-- IPv6 user configuration --
IPv6 config>unicast-routing
IPv6 config>general-prefix teldat 2001:db8:2222::/48
IPv6 config>
```

The interface is configured with two global addresses (one duplicated, since it is also configured on the host, and another generated on the basis of a *general prefix*) and two prefixes (one of them should not be advertised in the RA, and the host does not configure it).

```
Config>network ethernet0/0

-- Ethernet Interface User Configuration --
ethernet0/0 config>ipv6 enable
ethernet0/0 config>ipv6 address 2001:db8:1111::3/64
ethernet0/0 config>ipv6 address general-prefix teldat ::1234:0:0:0:1/64
ethernet0/0 config>ipv6 nd prefix 2001:db8:2222::/64
ethernet0/0 config>ipv6 nd prefix 2001:db8:3333::/64 no-advertise
ethernet0/0 config>ipv6 nd ra interval 30 10
```

```
ethernet0/0 config>ipv6 nd ra lifetime 90
ethernet0/0 config>
```

The final configuration for the router is as follows:

```
;
   set hostname Router
;
   network ethernet0/0
; -- Ethernet Interface User Configuration --
      ipv6 enable
      ipv6 address 2001:db8:1111::3/64
      ipv6 address general-prefix teldat ::1234:0:0:0:1/64
      ipv6 nd prefix 2001:db8:2222::/64
      ipv6 nd prefix 2001:db8:3333::/64 no-advertise
      ipv6 nd ra interval 30 10
      ipv6 nd ra lifetime 90
   exit
;
   protocol ipv6
; -- IPv6 user configuration --
      general-prefix teldat 2001:db8:2222::/48
      unicast-routing
   exit
;
```

Host

The host is configured with a global address, and autoconfiguration is enabled to create global addresses from the prefixes received from the router in the RAs.

```
Config>network ethernet0/0

-- Ethernet Interface User Configuration --
ethernet0/0 config>ipv6 enable
ethernet0/0 config>ipv6 address autoconfig
ethernet0/0 config>ipv6 address 2001:db8:1111::3/64
ethernet0/0 config>
```

The final configuration is as follows:

```
;
   set hostname Host
;
   network ethernet0/0
; -- Ethernet Interface User Configuration --
      ipv6 enable
      ipv6 address autoconfig
      ipv6 address 2001:db8:1111::3/64
   exit
;
```

Looking at IPv6 protocol monitoring through the **list interface ethernet0/0** command, you can see the state of the host's ethernet0/0 interface:

```
Host IPv6+list interface ethernet0/0

Interface ethernet0/0:
------------------------------
IPv6 is Enabled
Link-local address is:  fe80::2a0:26ff:fe32:28c8 [PERM]

Global unicast address(es):
  2001:db8:1111::3/64 cfg [PERM/UP]
  2001:db8:1111:0:2a0:26ff:fe32:28c8/64 ra-auto [OVR] valid lifetime 2591996s, preferred lifetime 604796s
  2001:db8:2222:0:2a0:26ff:fe32:28c8/64 ra-auto [UP] valid lifetime 2591996s, preferred lifetime 604796s
  2001:db8:2222:1234:2a0:26ff:fe32:28c8/64 ra-auto [UP] valid lifetime 2591996s, preferred lifetime 604796s

Joined group address(es):
  ff02::1:ff32:28c8
```

```
  ff02::1:ff00:3
  ff02::1

MTU is 1500 bytes
ICMP error messages limited to one every 1000 milliseconds
ICMP redirects are enabled
ICMP unreachables are sent
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds
Default router is fe80::2a0:26ff:fe28:3918 on ethernet0/0

IPv6 Prefix Advertisements ethernet0/0
Codes: A - Address, P - Prefix-Advertisement, N - Not advertised,
       [L] - On-link, [A] - Autonomous
A  2001:db8:1111::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
AN  2001:db8:2222::/64  [LA]  Valid lifetime: 2591995, preferred lifetime: 604800
AN  2001:db8:2222:1234::/64  [LA]  Valid lifetime: 2591995, preferred lifetime: 604800
Host IPv6+
```

The state of the router's ethernet0/0 interface is as follows:

```
Router IPv6+list interface ethernet0/0

Interface ethernet0/0:
----------------------------
IPv6 is Enabled
Link-local address is:  fe80::2a0:26ff:fe28:3918 [PERM]

Global unicast address(es):
  2001:db8:1111::3/64 cfg [DUP/TEN/PERM/UP]
  2001:db8:2222:1234::1/64 gen-pref [PERM/UP]

Joined group address(es):
  ff02::1:ff00:0
  ff02::1:ff28:3918
  ff02::1:ff00:1
  ff02::2
  ff02::1:ff00:3
  ff02::1

MTU is 1500 bytes
ICMP error messages limited to one every 1000 milliseconds
ICMP redirects are enabled
ICMP unreachables are sent
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds
ND advertised reachable time is 0 milliseconds (unspecified)
ND advertised retransmit interval is 0 milliseconds (unspecified)
ND router advertisements are sent every 10 to 30 seconds
ND router advertisements live for 90 seconds
ND advertised default router preference is Medium

IPv6 Prefix Advertisements ethernet0/0
Codes: A - Address, P - Prefix-Advertisement, N - Not advertised,
       [L] - On-link, [A] - Autonomous
P  2001:db8:2222::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
PN  2001:db8:3333::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
A  2001:db8:1111::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
A  2001:db8:2222:1234::/64  [LA]  Valid lifetime: 2592000, preferred lifetime: 604800
Router IPv6+
```