ETSI TS 102 514 V1.1.1 (2006-04)

Technical Specification

Methods for Testing and Specification (MTS);
Internet Protocol Testing (IPT);
IPv6 Core Protocol; Requirements Catalogue

Reference
DTS/MTS-IPT-003-IPv6-CoreReqCa

Keywords
IP, IPv6, testing

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

1 Scope

The purpose of the present document is to provide a catalogue of requirements extracted from the core IPv6 RFCs (see references in clause 2) and from ETSI Specifications. The catalogue follows the guidelines defined by the MTS IPv6 Testing: Methodology and Framework (see TS 102 351 in bibliography).

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] IETF RFC 1122: "Requirements for Internet Hosts -- Communication Layers".

[2] IETF RFC 1981: "Path MTU Discovery for IP version 6".

[3] IETF RFC 2373: "IP Version 6 Addressing Architecture".

[4] IETF RFC 2402: "IP Authentication Header".

[5] IETF RFC 2460: "Internet Protocol, Version 6 (IPv6) Specification".

[6] IETF RFC 2461: "Neighbor Discovery for IP Version 6 (IPv6)".

[7] IETF RFC 2462: "IPv6 Stateless Address Autoconfiguration".

[8] IETF RFC 2463: "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification".

[9] IETF RFC 2464: "Transmission of IPv6 Packets over Ethernet Networks".

[10] IETF RFC 2675: "IPv6 Jumbograms".

[11] IETF RFC 3513: " Internet Protocol Version 6 (IPv6) Addressing Architecture".

[12] ETSI TS 123 060: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); General Packet Radio Service (GPRS); Service description; Stage 2 (3GPP 23.060)".

[13] ETSI TS 123 221: " Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Architectural requirements (3GPP 23.221)".

[14] ETSI TS 123 228: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); IP Multimedia Subsystem (IMS); Stage 2 (3GPP 23.228)".

[15] ETSI TS 129 061: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN) (TS 129 061)".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CIDR	Common InterDomain Routing
ICMP	Internet Control Message Protocol
IE	Information Element
MTU	Maximum Transmission Unit
ND	Neighbor Discovery
PDP	Packet Data Protocol
PMTU	Path MTU
TCP	Transfer Control Protocol
UDP	User Datagram Protocol

4 Requirements Catalogue

The requirements below have been extracted from IETF RFCs 1981 [2], 2460 [5], 2461 [6], 2462 [7], 2463 [8], 2464 [9], 2675 [10], 3513[11]), and ETSI specifications TS 123 060 [12], TS 123 221 [13], TS 123 228 [14], TS 129061 [15]).

4.1 Requirements extracted from TS 123 060

RQ_COR_7003 Configure Address

TS 123 060 *Clause:* 9.2.1.1 ¶5 *Type:* MUST *applies to:* Host

Context: Implementation has started PDP context activation session.

Requirement: Implementation obtains interface identifier from GGSN prior to performing stateless or stateful address autoconfiguration.

RFC text: To ensure that the link-local address generated by the MS does not collide with the link-local address of the GGSN, the GGSN shall provide an interface identifier (see RFC 2462) to the MS and `{{the MS shall use this interface identifier to configure its link-local address. This is applicable for both stateful and stateless IPv6 address autoconfiguration.}}`.

RQ_COR_7004 Detect Duplicate Address

TS 123 060 *Clause:* 9.2.1.1 ¶10 *Type:* MAY *applies to:* Host

Context: Implementation is performing stateless address autoconfiguration. The prefix that the router (GGSN) advertises in a PDP context is unique within the scope of the prefix (i.e. global).

Requirement: Implementation does not perform duplicate address detection.

RFC text: Because any prefix that the GGSN advertises in a PDP context is unique within the scope of the prefix (i.e. site-local or global), `{{there is no need for the MS to perform Duplicate Address Detection for this IPv6 address}}`.

RQ_COR_7005 Detect Duplicate Address

TS 123 060 *Clause:* 9.2.1.1 ¶10 *Type:* MUST *applies to:* Router

Context: Implementation (GGSN) advertises prefixes in a PDP context that are unique (i.e. global). The implementation receives Neighbor Solicitation messages for Duplicate Address Detection.

Requirement: Implementation silently discards the Neighbor Solicitation messages.

RFC text: Because any prefix that the GGSN advertises in a PDP context is unique within the scope of the prefix (i.e. site-local or global), there is no need for the MS to perform Duplicate Address Detection for this IPv6 address. Therefore, the GGSN shall `{{silently discard Neighbor Solicitation messages that the MS may send to perform Duplicate Address Detection. }}.`

RQ_COR_7006 Stateless Autoconfiguration

TS 123 060 *Clause:* 9.2.1.1 ¶10 *Type:* MUST *applies to:* Router

Context: The implementation is activating PDP contexts for IPv6.

Requirement: Implementation does not advertise the same prefix on more than one PDP context on a given APN or set of APNs, within the same addressing scope.

RFC text: `{{A given prefix shall not be advertised on more than one PDP context on a given APN or set of APNs within the same addressing scope. }}`.

RQ_COR_7007 Stateless Autoconfiguration

TS 123 060 *Clause:* 9.2.1.1 ¶10 *Type:* MUST *applies to:* Host

Context: Implementation has received a Router Advertisement message which contains more than one prefix.

Requirement: The implementation uses the first prefix and silently discards the others.

RFC text: After the MS has received the Router Advertisement message, it constructs its full IPv6 address by concatenating the interface identifier received in step 3, or a locally generated interface identifier, and the prefix received in the Router Advertisement.

RQ_COR_7009 Router Advertisement Behavior

TS 123 060 *Clause:* 9.2.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: Implementation (GGSN) has activated a PDP context of type IPv6.

Requirement: The implementation sends automatic and periodic router advertisement messages towards the node (MS).

RFC text: The GGSN informs the MS that it shall perform stateful address autoconfiguration by means of the Router Advertisements, as defined in RFC 2461. For this purpose, `{{the GGSN shall automatically and periodically send Router Advertisement messages towards the MS after a PDP context of type IPv6 is activated}}.`

4.2 Requirements extracted from TS 123 221

RQ_COR_7010 3GPP UE supports IPv6

TS 123 221 *Clause:* 5.6 ¶2 *Type:* MUST *applies to:* Host

Context: 3GPP UE supports IPv6

Requirement: The implementation complies with the Basic IP group of specifications as defined in RFC 3316.

RFC text: The set of IPv6 functionality a 3GPP UE will require is dependent on the services (IMS, Packet Streaming etc.) it will use.
As a minimum, a `{{3GPP UE shall comply with the Basic IP group of specifications as defined in RFC 3316 }}`. This IPv6 functional is sufficient to provide compatibility towards IPv6 entities external to 3GPP.

4.3 Requirements extracted from TS 123 228

RQ_COR_7060 Configure Address

TS 123 228 *Clause:* 4.3.1 *Type:* MAY *applies to:* Host

Context: Implementation is assigned an IPv6 prefix.

Requirement: Implementation changes its global IPv6 address currently in use via the mechanism defined in RFC 3041 or similar means.

RFC text: `{{According to the procedures defined in TS 123 060 , when a UE is assigned an IPv6 prefix, it can change the global IPv6 address it is currently using via the mechanism defined in RFC 3041, or similar means}}`.

4.4 Requirements extracted from TS 129 061

RQ_COR_7000 Configure Address

TS 129 061 *Clause:* 11.2.1.3 ¶4-5 *Type:* MUST *applies to:* Host

Context: The implementation is capable of both stateless and stateful autoconfiguration and receives a Router Advertisement indicating both stateless and stateful autoconfiguration capabilities.

Requirement: The implementation uses stateless to configure the address and stateful to configure additional parameters.

RFC text: `{{Stateful and Stateless Autoconfiguration may also co-exist. In that case, the MS shall use Stateless to configure the address and stateful to configure additional parameters only}}`.
`{{The selection between Stateful and Stateless Autoconfiguration is dictated by the Router Advertisements}}` sent by the GGSN as described in the corresponding subclauses below and according to the principles defined in RFC 2461 and RFC 2462.

RQ_COR_7001 Configure Address

TS 129 061 *Clause:* 11.2.1.3 ¶4 *Type:* MUST *applies to:* Host

Context: The implementation is capable of both stateless and stateful autoconfiguration and router advertise both stateless and stateful autoconfiguration.

Requirement: The implementation does not use both stateless and stateful autoconfiguration simultaneously.

RFC text: `{{The MS shall not use Stateless and Stateful Address Autoconfiguration simultaneously since GPRS only supports one prefix per PDP Context}}`.

RQ_COR_7002 Configure Address

TS 129 061 *Clause:* 11.2.1.3 ¶5 *Type:* MUST *applies to:* Host

Context: The implementation does not support optional stateful autoconfiguration and router advertises both stateful and stateless autoconfiguration.

Requirement: The implementation uses stateless autoconfiguration only.

RFC text: `{{For MS, IPv6 Stateless Address Autoconfiguration is mandatory, and IPv6 Stateful Address Autoconfiguration is optional}}`.

RQ_COR_7008 Router Advertisement Behavior

TS 129 061 *Clause:* 11.2.1.3.2 *Type:* MAY *applies to:* Router

Context: Implementation (GGSN) is sending Router Advertisements.

Requirement: The implementation omits random sending of the Advertisements.

RFC text: The handling of Router Advertisements shall be consistent with what is specified in RFC 2461 . For the MS-GGSN link however, some specific handling shall apply. `{{The randomisation part to determine when Router Advertisements shall be sent may be omitted since the GGSN is the only router on the link}}`.

RQ_COR_7012 MaxRtrAdvInterval

TS 129 061 *Clause:* 11.2.1.3.4 *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation and MaxRtrAdvInterval is left at the default 3GPP value.

Requirement: The implementation uses the default value of 21,600s (6 hours) for MaxRtrAdvInterval.

RFC text: RFC 2461 specifies a set of conceptual router configuration variables. Some of these variables require particular attention in GPRS in order to preserve radio resources and MS power consumption while still allowing for appropriate robustness and fast use.

RQ_COR_7013 MinRtrAdvInterval

TS 129 061 *Clause:* 11.2.1.3.4 *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation and MinRtrAdvInterval is left at the default 3GPP value.

Requirement: The implementation uses the default value of 0,75 × MaxRtrAdvInterval, i.e. 16,200 s (4,5 hours), for MinRtrAdvInterval .

RFC text: RFC 2461 specifies a set of conceptual router configuration variables. Some of these variables require particular attention in GPRS in order to preserve radio resources and MS power consumption while still allowing for appropriate robustness and fast use.

RQ_COR_7014 RA Prefix Option

TS 129 061 *Clause:* 11.2.1.3.4 *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: The implementation uses an infinite lifetime, i.e. 0xFFFFFFFF, for the value of AdvValidLifetime for prefixes.

RFC text: RFC 2461 specifies a set of conceptual router configuration variables. Some of these variables require particular attention in GPRS in order to preserve radio resources and MS power consumption while still allowing for appropriate robustness and fast use.

RQ_COR_7015 RA Prefix Option

TS 129 061 *Clause:* 11.2.1.3.4 *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: The implementation uses an infinite lifetime, i.e. 0xFFFFFFFF, for the value of AdvPreferredLifetime for prefixes.

RFC text: RFC 2461 specifies a set of conceptual router configuration variables. Some of these variables require particular attention in GPRS in order to preserve radio resources and MS power consumption while still allowing for appropriate robustness and fast use.

RQ_COR_7016 ND Protocol Constants and Default Values

TS 129 061 *Clause:* 11.2.1.3.4 *Type:* MAY *applies to:* Router

Context: The implementation is functioning.

Requirement: The "constant" MAX_INITIAL_RTR_ADVERT_INTERVAL varies.

RFC text: RFC 2461 specifies a set of conceptual router configuration variables. Some of these variables require particular attention in GPRS in order to preserve radio resources and MS power consumption while still allowing for appropriate robustness and fast use.

RQ_COR_7017 ND Protocol Constants and Default Values

TS 129 061 *Clause:* 11.2.1.3.4 *Type:* MUST *applies to:* Router

Context: The implementation is functioning.

Requirement: The MAX_INITIAL_RTR_ADVERTISEMENTS is assigned a value so as to not overload the radio interface while still allowing node (MS) to complete its configuration in a reasonable delay.

RFC text: RFC 2461 specifies a set of conceptual router configuration variables. Some of these variables require particular attention in GPRS in order to preserve radio resources and MS power consumption while still allowing for appropriate robustness and fast use.

RQ_COR_7018 Configure Address

TS 129 061 *Clause:* 11.2.1.3.1 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation has started an IPv6 PDP context activation.

Requirement: Implementation provides an IPv6 prefix belonging to the Intranet/ISP addressing space.

RFC text: `{{The GGSN provides the MS with an IPv6 Prefix belonging to the Intranet/ISP addressing space}}`.

RQ_COR_7019 Configure Address

TS 129 061 *Clause:* 11.2.1.3.1 ¶1 *Type:* MUST *applies to:* Router

Context: Implementation has activated an IPv6 PDP context and started dynamic address configuration.

Requirement: Implementation gives a dynamic IPv6 address using either stateless or stateful address autoconfiguration.

RFC text: `{{A dynamic IPv6 address shall be given using either stateless or stateful address autoconfiguration}}`. This IPv6 address is used for packet forwarding within the packet domain and for packet forwarding on the Intranet/ISP.

RQ_COR_7020 RA Prefix Option

TS 129 061 *Clause:* 11.2.1.3.4 *Type:* MUST *applies to:* Router

Context: The implementation has activated an IPv6 PDP Context and assigned a dynamic IPv6 address to UE.

Requirement: The assigned prefix remains valid and preferred until PDP context deactivation.

RFC text: RFC 2461 specifies a set of conceptual router configuration variables. Some of these variables require particular attention in GPRS in order to preserve radio resources and MS power consumption while still allowing for appropriate robustness and fast use.

RQ_COR_7021 Configure Address

TS 129 061 *Clause:* 11.2.1.1 *Type:* MUST *applies to:* Router

Context: The implementation is assigning an IPv6 address.

Requirement: Implementation uses either stateless or stateful address autoconfiguration procedure to assign an IPv6 address to the node (MS).

RFC text: the MS is given an address or IPv6 Prefix belonging to the operator addressing space. The address or IPv6 Prefix is given either at subscription in which case it is a static address or at PDP context activation in which case it is a dynamic address. This

RQ_COR_7022 Neighbor Discovery

TS 129 061 *Clause:* 11.2.1.3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation supports IPv6.

Requirement: The implementation complies with the principles of RFC 2461 for sending Router Advertisements.

RFC text: The selection between Stateful and Stateless Autoconfiguration is dictated by the `{{Router Advertisements sent by the GGSN as described in the corresponding subclauses below and according to the principles defined in RFC 2461}}` and RFC 2462.

RQ_COR_7023 Stateless Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation supports IPv6.

Requirement: The implementation complies with the principles of RFC 2462 for stateless address autoconfiguration.

RFC text: The selection between Stateful and Stateless Autoconfiguration is dictated by the `{{Router Advertisements sent by the GGSN as described in the corresponding subclauses below and according to the principles defined in RFC 2461 and RFC 2462 }}`.

RQ_COR_7024 Form Link-local Address

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MUST *applies to:* Host

Context: The implementation has received an Interface Identifier during PDP context activation or IPV6CP negotiation.

Requirement: The implementation uses the Interface Identifier to create the link-local address for IPv6 address autoconfiguration.

RFC text: The MT finalises the IPV6CP negotiation by sending an IPV6CP Configure-Ack message to the TE with the appropriate options included, e.g. Interface-Identifier. The `{{Interface-Identifier shall be used in the TE to create a link-local address to be able to perform the IPv6 address autoconfiguration}}` (see clauses 11.2.1.3.2 and 11.2.1.3.3).

RQ_COR_7025 3GPP Startup Router Behavior

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MUST *applies to:* Router

Context: The implementation is operating.

Requirement: Implementation deduces from local configuration data associated with the APN:
IPv6 address allocation type (stateless or stateful);
the source of IPv6 Prefixes in the stateless case (GGSN internal prefix pool, or external address allocation server).

RFC text:
```
{{The GGSN deduces from local configuration data associated with the
APN:
-     IPv6 address allocation type (stateless or stateful);
-     the source of IPv6 Prefixes in the stateless case (GGSN
internal prefix pool, or external address allocation server)}}.
```

RQ_COR_7026 3GPP Startup Router Behavior

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MUST *applies to:* Router

Context: The implementation is configured for operation and PDP context activation procedure is in progress.

Requirement: The implementation sends in the Create PDP Context Response message an IPv6 address composed of a Prefix and Interface Identifier in the message's PDP Address IE.

RFC text:
```
{{The GGSN shall in the PDP Address IE in the Create PDP Context
Response return an IPv6 address composed of a Prefix and an
Interface-Identifier}}.
```

RQ_COR_7027 3GPP Startup Router Behavior

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MUST *applies to:* Router

Context: The implementation is configured for operation and generating a PDP Context Response message.

Requirement: The implementation forms the Interface Identifier with any value that does not conflict with the Interface-Identifier the GGSN has selected for its own side of the MS-GGSN link.

RFC text: The GGSN shall in the PDP Address IE in the Create PDP Context Response return an IPv6 address composed of a Prefix and an Interface-Identifier. `{{The Interface-Identifier may have any value and it does not need to be unique within or across APNs. It shall however not conflict with the Interface-Identifier the GGSN has selected for its own side of the MS-GGSN link}}`.

RQ_COR_7028 3GPP Startup Router Behavior

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MUST *applies to:* Router

Context: The implementation is configured for operation. PDP context activation procedure in progress. The APN uses stateless address autoconfiguration.

Requirement: The implementation assigns a Prefix that is globally unique.

RFC text: The GGSN shall in the PDP Address IE in the Create PDP Context Response return an IPv6 address composed of a Prefix and an Interface-Identifier. The Interface-Identifier may have any value and it does not need to be unique within or across APNs. It shall however not conflict with the Interface-Identifier the GGSN has selected for its own side of the MS-GGSN link. `{{The Prefix assigned by the GGSN or the external AAA server shall be globally or site-local unique, if stateless address autoconfiguration is configured on this APN}}`. If, on the other hand, stateful address autoconfiguration is configured on the APN, the Prefix part of the IPv6 address returned in the PDP Address IE shall be set to the link-local prefix (FE80::/64).

RQ_COR_7029 Configure Address

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MUST *applies to:* Host

Context: PDP context activation procedure in progress. IUT has received the PDP Address IE in the Create PDP Context Response which contains IPv6 address composed of a Prefix and an Interface-Identifier.

Requirement: The implementation extracts the Interface-Identifier from the address received in the PDP Address IE and ignores the Prefix part.

RFC text: `{{The MT extracts the Interface-Identifier from the address received in the PDP Address IE and ignores the Prefix part}}`.

RQ_COR_7030 Stateless Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.2 *Type:* MUST *applies to:* Node

Context: PDP context activation procedure is completed and stateless address autoconfiguration procedure has started.

Requirement: The implementation uses the IPv6 Interface-Identifier, as provided by the router(GGSN), to create its IPv6 Link-Local Unicast Address according to RFC 2373.

RFC text: `{{After the first phase of setting up IPv6 access to an Intranet or ISP, the MS shall use the IPv6 Interface-Identifier, as provided by the GGSN, to create its IPv6 Link-Local Unicast Address according to RFC 2373 }}`.

RQ_COR_7031 Stateless Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.2 *Type:* MUST *applies to:* Node

Context: PDP context activation procedure completed and stateless address autoconfiguration procedure has started.

Requirement: The implementation obtains an IPv6 Global or Site-Local Unicast Address using an IPv6 stateless address Autoconfiguration procedure.

RFC text: `{{Before the MS can communicate with other hosts or MSs on the Intranet/ISP, the MS must obtain an IPv6 Global or Site-Local Unicast Address. The simplest way is the IPv6 Stateless Address Autoconfiguration procedure described below and in TS 123 060. The procedure is consistent with RFC 2462 }}`.

RQ_COR_7032 Startup Router Advertisement Behavior

TS 129 061 *Clause:* 11.2.1.3.2 *Type:* MUST *applies to:* Router

Context: Implementation has sent a Create PDP Context Response message to complete the PDP context activation procedure.

Requirement: The implementation sends Router Advertisements periodically on the new MS-GGSN link established by the PDP Context.

RFC text: `{{After the GGSN has sent a Create PDP Context Response message to the SGSN, it shall start sending Router Advertisements periodically on the new MS-GGSN link established by the PDP Context}}`.

RQ_COR_7033 Stateless Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.2 *Type:* MUST *applies to:* Router

Context: The IUT uses stateless address autoconfiguration.

Requirement: Implementation sends Router Advertisements with the M-flag cleared to zero.

RFC text: `{{To indicate to the MS that stateless address autoconfiguration shall be performed, the GGSN shall leave the M flag cleared in the Router Advertisement messages}}`.

RQ_COR_7034 Simultaneous Stateless and Stateful

TS 129 061 *Clause:* 11.2.1.3.2 *Type:* MUST *applies to:* Host

Context: Implementation is performing stateless address autoconfiguration and receives Router advertisement message with M-flag set to indicate that hosts perform stateful address autoconfiguration

Requirement: Implementation does not perform stateless and stateful address autoconfiguration simultaneously, since multiple prefixes are not allowed in GPRS.

RFC text: To indicate to the MS that stateless address autoconfiguration shall be performed, the GGSN shall leave the M flag cleared in the Router Advertisement messages. `{{An MS shall not perform stateless and stateful address autoconfiguration simultaneously, since multiple prefixes are not allowed in GPRS}}`.

RQ_COR_7035 Use of O-Flag

TS 129 061 *Clause:* 11.2.1.3.2 *Type:* MUST *applies to:* Host

Context: Implementation receives a Router advertisement message with M-flag cleared to zero and the O-flag set to one.

Requirement: The MS performs stateless autoconfiguration for only one IPv6 address.

RFC text: To indicate to the MS that stateless address autoconfiguration shall be performed, the GGSN shall leave the M flag cleared in the Router Advertisement messages. An MS shall not perform stateless and stateful address autoconfiguration simultaneously, since

RQ_COR_7036 Stateful Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.3 ¶2 *Type:* MUST *applies to:* Router

Context: Implementation (GGSN) is generating a Router Advertisement message for Stateful address autoconfiguration

Requirement: The Router Advertisement does not contain the Prefix-Information option and the M-flag is set to one.

RFC text: `{{To indicate to the MS that Stateful Address Autoconfiguration shall be performed, the Router Advertisements shall not contain any Prefix-Information option and the M-flag ("Managed Address Configuration Flag") shall be set}}`.

RQ_COR_7037 Stateful Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MAY *applies to:* Node

Context: Implementation is performing Stateful address configuration.

Requirement: Implementation uses DHCPv6 to determine the address's prefix.

RFC text: `{{DHCPv6 may be used for IPv6 prefix allocation}}`.

RQ_COR_7038 RA Prefix Option

TS 129 061 *Clause:* 11.2.1.3.1 ¶6 *Type:* MUST *applies to:* Router

Context: The implementations has an internal GGSN IPv6 address Prefix pool.

Requirement: Implementation's internal IPv6 prefix pool is configurable and structured per APN.

RFC text: `{{IPv6 Prefixes in a GGSN internal Prefix pool shall be configurable and structured per APN}}`.

RQ_COR_7039 IPV6CP Behavior

TS 129 061 *Clause:* 11.2.1.3.1 ¶9 *Type:* MUST *applies to:* Host

Context: IPv6 Context Activation is underway. The implementation receives an Interface-Identifier in the PDP Address IE that is not identical to the tentative Interface-Identifier indicated in the IPV6CP Configure-Request message.

Requirement: Implementation sends an IPV6CP Configure-Nak packet indicating the Interface-Identifier extracted from the address contained in the PDP Address IE. The implementation then sends a new IPV6CP Configure-Request message indicating the same Interface-Identifier as was indicated in the received IPV6CP Configure Nak.

RFC text: `{{If the Interface-Identifier extracted from the address contained in the PDP Address IE is not identical to the tentative Interface-Identifier indicated in the IPV6CP Configure-Request message sent from the TE, the MT sends an IPV6CP Configure-Nak packet, indicating the Interface-Identifier extracted from the address contained in the PDP Address IE, to the TE. The TE then sends a new IPV6CP Configure-Request message to the MT, indicating the same Interface-Identifier as was indicated in the received IPV6CP Configure Nak}}`.

RQ_COR_7041 IPV6CP Behavior

TS 129 061 *Clause:* 11.2.1.3.1 ¶9 *Type:* MUST *applies to:* Host

Context: IPv6 Context Activation is underway. The implementation receives an Interface-Identifier in the PDP Address IE that is identical to the tentative Interface-Identifier indicated in the IPV6CP Configure-Request message.

Requirement: Implementation sends an IPV6CP Configure Ack packet.

RFC text: If the Interface-Identifier extracted from the address contained in the PDP Address IE is not identical to the tentative Interface-Identifier indicated in the IPV6CP Configure-Request message sent from the TE, the MT sends an IPV6CP Configure-Nak packet,

RQ_COR_7042 Stateless Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.2 ¶2 *Type:* MUST *applies to:* Router

Context: IPv6 layer is operating. Implementation receives a valid Router Solicitation.

Requirement: Implementation immediately sends a Router Advertisement.

RFC text: `{{The MS may issue a Router Solicitation directly after the user plane establishment. This shall trigger the GGSN to send a Router Advertisement immediately}}`.

RQ_COR_7043 RA Prefix Option

TS 129 061 *Clause:* 11.2.1.3.2 ¶2 *Type:* MUST *applies to:* Router

Context: An IPv6 PDP context has just been activated. Implementation is performing stateless address autoconfiguration.

Requirement: Implementation generates Router Advertisements containing only one Prefix Option whose Prefix value is identical to the Prefix returned in the Create PDP Context Response. The Prefix Option's A-flag is set to one, its L-flage is set to zero, and the Prefix lifetime is set to infinity.

RFC text: `{{The Prefix sent in the Router Advertisements shall be identical to the Prefix returned in the Create PDP Context Response. The Prefix is contained in the Prefix Information Option of the Router Advertisements and shall have the A-flag set ("Autonomous address configuration flag") and the L-flag cleared (i.e. the prefix should not be used for on-link determination). The lifetime of the prefix shall be set to infinity. In practice, the lifetime of a Prefix will be the lifetime of its PDP Context. There shall be exactly one Prefix included in the Router Advertisements.}}.`

RQ_COR_7047 Startup Router Advertisement Behavior

TS 129 061 *Clause:* 11.2.1.3.2 ¶2 *Type:* MUST *applies to:* Router

Context: Implementation supports IPv6.

Requirement: Implementation's handling of Router Advertisements is consistent with RFC 2461.

RFC text: `{{The handling of Router Advertisements shall be consistent with what is specified in RFC 2461 }}`.

RQ_COR_7048 Unicast Address

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* MUST *applies to:* Host

Context: Implementation is creating a Global Unicast Address.

Requirement: Implementation uses the Interface-Identifier received during the PDP Context Activation phase or it generates a new Interface-Identifier.

RFC text: `{{When creating a Global or Site-Local Unicast Address, the MS may use the Interface-Identifier received during the PDP Context Activation phase or it may generate a new Interface-Identifier}}`.

RQ_COR_7049 Unicast Address

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* MUST *applies to:* Host

Context: Implementation is creating a Global Unicast Address.

Requirement: Implementation puts no restriction on the value of the Interface-Identifier of the Global Unicast Address.

RFC text: When creating a Global or Site-Local Unicast Address, the MS may use the Interface-Identifier received during the PDP Context Activation phase or it may generate a new Interface-Identifier. `{{There is no restriction on the value of the Interface-Identifier of the Global or Site-Local Unicast Address, since the Prefix is unique}}`.

RQ_COR_7050 Unicast Address

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* MUST *applies to:* Host

Context: Implementation is creating a Global Unicast Address.

Requirement: Implementation's Interface-Identifier is 64-bits long.

RFC text: When creating a Global or Site-Local Unicast Address, the MS may use the Interface-Identifier received during the PDP Context Activation phase or it may generate a new Interface-Identifier. There is no restriction on the value of the Interface-Identifier.

RQ_COR_7051 Detect Duplicate Address

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* SHOULD *applies to:* Host

Context: Implementation is performing stateless address autoconfiguration.

Requirement: Implementation does not perform any Duplicate Address Detection on addresses it creates.

RFC text: Since the GGSN guarantees that the Prefix is unique, `{{the MS does not need to perform any Duplicate Address Detection on addresses it creates}}`.

RQ_COR_7052 Duplicate Address Detection Timers and

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* SHOULD *applies to:* Host

Context: Implementation is in Address Autoconfiguration.

Requirement: Implementation's DupAddrDetectTransmits variable is set to zero.

RFC text: Since the GGSN guarantees that the Prefix is unique, the MS does not need to perform any Duplicate Address Detection on addresses it creates. `{{That is, the 'DupAddrDetectTransmits' variable in the MS should have a value of zero}}`.

RQ_COR_7053 Configure Address

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* MUST *applies to:* Router

Context: Implementation is creating an IPv6 address for itself.

Requirement: Implementation does not generate its address using the Prefix assigned to any MS in Router Advertisement messages.

RFC text: `{{The GGSN shall not generate any globally unique IPv6 addresses for itself using the Prefix assigned to the MS in the Router Advertisement}}`.

RQ_COR_7054 Use of O-Flag

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* MAY *applies to:* Node

Context: Implementation receives O-flag ("Other stateful configuration flag") set to one in Router Advertisement.

Requirement: Implementation starts a DHCP session to retrieve additional configuration parameters.

RFC text: `{{If the O-flag ("Other stateful configuration flag") was set in the Router Advertisement, the MS may start a DHCP session to retrieve additional configuration parameters}}`.

RQ_COR_7055 Use of O-Flag

TS 129 061 *Clause:* 11.2.1.3.2 ¶3 *Type:* MAY *applies to:* Host

Context: Implementation is not DHCP capable. Implementation receives a Router Advertisement. The O-flag ("Other stateful configuration flag") is cleared to zero.

Requirement: The Implementation ignores the O-flag.

RFC text: If the O-flag ("Other stateful configuration flag") was set in the Router Advertisement, the MS may start a DHCP session to retrieve additional configuration parameters. See clause 13.2.2 "Other configuration by the Intranet or ISP". `{{If the MS is not DHCP capable, the O-flag may be ignored}}`.

RQ_COR_7056 Stateful Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.3 ¶1 *Type:* MUST *applies to:* Host

Context: PDP context activation procedure completed and stateful address autoconfiguration procedure has started. The first phase of access to Intranet or an ISP is complete.

Requirement: The implementation uses the IPv6 Interface-Identifier, as provided by the router(GGSN), to create its IPv6 Link-Local Unicast Address according to RFC 2373.

RFC text: `{{After the first phase of setting up IPv6 access to an Intranet or ISP, the MS shall use the IPv6 Interface Identifier, as provided by the GGSN, to create its IPv6 Link-Local Unicast Address according to RFC 2373 }}`.

RQ_COR_7057 Stateful Autoconfiguration

TS 129 061 *Clause:* 11.2.1.3.3 ¶1 *Type:* MUST *applies to:* Host

Context: Implementation is capable of Stateful Autoconfiguration. It receives Router Advertisements with the M-flag set to one.

Requirement: Implementation starts a DHCPv6 configuration to request an IPv6 address.

RFC text: `{{When the MS has received a Router Advertisement with the M-flag set, it shall start a DHCPv6 configuration as described in clause "Address allocation using DHCPv6" including a request for an IPv6 address}}`.

RQ_COR_7058 Autoconfigure Address

TS 129 061 *Clause:* 11.2.1.3.3 ¶1 *Type:* MUST *applies to:* Router

Context: Implementation (GGSN) supports IPv6.

Requirement: Implementation behaves as a IPv6 router and is consistent with the RFCs specifying Stateless and Stateful Address Autoconfiguration unless stated otherwise in TS 129 061 or other ETSI specifications.

RFC text: `{{For IPv6 Stateless and Stateful Address Autoconfiguration to work properly the GGSN shall behave as an IPv6 router towards the MS. In this respect the GGSN shall be consistent with the RFCs specifying this process (for example RFC 2462 and RFC 2461), unless stated otherwise in this or other ETSI specifications}}`.

RQ_COR_7059 Simultaneous Stateless and Stateful

TS 129 061 *Clause:* 13a.2.1 *Type:* MAY *applies to:* Host

Context: Implementation is DHCPv6 capable. Implementation receives a Router Advertisement with its "M-flag" is cleared and the "O- flag" set to one.

Requirement: Implementation simultaneously uses stateless address autoconfiguration for configuring its IPv6 address and stateful autoconfiguration for configuring IMS specific parameters.

RFC text: `{{When the "M-flag" is cleared, the "O- flag" shall be set in IPv6 Router Advertisement messages sent by the GGSN for APNs used for IMS services. This will trigger a DHCP capable MS to start a DHCPv6 session to retrieve server addresses and other configuration parameters. An MS which doesn't support DHCP will simply ignore the "O-flag". An MS may simultaneously use stateless address autoconfiguration for configuring its IPv6 address and stateful autoconfiguration for configuring IMS specific parameters}}`.

RQ_COR_7061 Configure Address

TS 129 061 *Clause:* 11.2.1.3 ¶5 *Type:* MAY *applies to:* Host

Context: The implementation supports optional stateful autoconfiguration and router advertises stateful autoconfiguration.

Requirement: The implementation performs stateful autoconfiguration.

RFC text: For MS, IPv6 Stateless Address Autoconfiguration is mandatory, and `{{IPv6 Stateful Address Autoconfiguration is optional}}`.

RQ_COR_7999 Configure Address

TS 129 061 *Clause:* 11.2.1.3.1 *Type:* MUST *applies to:* Router

Context: The implementation is configured for operation. PDP context activation procedure in progress. The APN uses stateful address autoconfigureation.

Requirement: The implementation assigns a Prefix that is set to the link-local prefix (FE80::/64).

RFC text: The GGSN shall in the PDP Address IE in the Create PDP Context Response return an IPv6 address composed of a Prefix and an Interface-Identifier. The Interface-Identifier may have any value and it does not need to be unique within or across APNs. It shall however not conflict with the Interface-Identifier the GGSN has selected for its own side of the MS-GGSN link. The Prefix assigned by the GGSN or the external AAA server shall be globally or site-local unique, if stateless address autoconfiguration is configured on this APN. `{{If, on the other hand, stateful address autoconfiguration is configured on the APN, the Prefix part of the IPv6 address returned in the PDP Address IE shall be set to the link-local prefix (FE80::/64)}}.`

4.5 Requirements extracted from RFC 1981

RQ_COR_1800 PMTU Discovery

RFC 1981 *Clause:* 1 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation uses IPv6. The implementation sends a large amount of series of IPv6 packets to another implementation.

Requirement: The implementation sends these packets at the largest size that can successfully traverse the path from the source implementation to the destination. This packet size is referred to as the Path MTU (PMTU), and it is equal to the minimum link MTU of all the links in a path.

RFC text: {{When one IPv6 node has a large amount of data to send to another node, the data is transmitted in a series of IPv6 packets. It is usually preferable that these packets be of the largest size that can successfully traverse the path from the source node to the destination node. This packet size is referred to as the Path MTU (PMTU), and it is equal to the minimum link MTU of all the links in a path. IPv6 defines a standard mechanism for a node to discover the PMTU of an arbitrary path}}.

RQ_COR_1801

RFC 1981 *Clause:* 1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The implementation sends a large amount of series of IPv6 packets to another implementation at the largest size that can successfully traverse the path from the source implementation to the destination. This packet size is referred to as the Path MTU (PMTU), and it is equal to the minimum link MTU of all the links in a path.

Requirement: The implementation uses a standard mechanism, [named Path MTU Discovery for IPv6], in order to discover the PMTU of an arbitrary path.

RFC text: {{When one IPv6 node has a large amount of data to send to another node, the data is transmitted in a series of IPv6 packets. It is usually preferable that these packets be of the largest size that can successfully traverse the path from the source node to the destination node. This packet size is referred to as the Path MTU (PMTU), and it is equal to the minimum link MTU of all the links in a path. IPv6 defines a standard mechanism for a node to discover the PMTU of an arbitrary path}}.

RQ_COR_1802

RFC 1981 *Clause:* 1 ¶2, 4 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation uses IPv6.

Requirement: The implementation implements Path MTU Discovery in order to discover and take advantage of paths with PMTU greater than the IPv6 minimum link MTU.

RFC text: {{IPv6 nodes SHOULD implement Path MTU Discovery in order to discover and take advantage of paths with PMTU greater than the IPv6 minimum link MTU [IPv6-SPEC]}}. A minimal IPv6 implementation (e.g., in a boot ROM) may choose to omit implementation of Path MTU Discovery.

RQ_COR_1803

RFC 1981 *Clause:* 1 ¶2 *Type:* MAY *applies to:* Minimal

Context: The implementation uses IPv6.

Requirement: The implementation does not implement Path MTU Discovery.

RFC text: IPv6 nodes SHOULD implement Path MTU Discovery in order to discover and take advantage of paths with PMTU greater than the IPv6 minimum link MTU [IPv6-SPEC]. `{{A minimal IPv6 implementation (e.g., in a boot ROM) may choose to omit implementation of Path MTU Discovery}}`.

RQ_COR_1804

RFC 1981 *Clause:* 1 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6. The implementation does not implement Path MTU Discovery.

Requirement: The implementation uses the IPv6 minimum link MTU defined in [IPv6-SPEC] as the maximum packet size.

RFC text: `{{Nodes not implementing Path MTU Discovery use the IPv6 minimum link MTU defined in [IPv6-SPEC] as the maximum packet size}}`. In most cases, this will result in the use of smaller packets than necessary, because most paths have a PMTU greater than the IPv6 minimum link MTU. A node sending packets much smaller than the Path MTU allows is wasting network resources and probably getting suboptimal throughput.

RQ_COR_1805 PMTU Discovery

RFC 1981 *Clause:* 3 ¶1, 5.2 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation needs to send IPv6 packets.

Requirement: The implementation initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path.

RFC text: This memo describes a technique to dynamically discover the PMTU of a path. `{{The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path}}`. If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages [ICMPv6]. Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message.

RQ_COR_1806 PMTU Discovery

RFC 1981 *Clause:* 3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation needs to sends IPv6 packets. The implementation initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path.

Requirement: The implementation sends packets at the (known) MTU of the first hop in the path.

RFC text: This memo describes a technique to dynamically discover the PMTU of a path. `{{The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path}}`. If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages [ICMPv6]. Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message.

RQ_COR_1807

RFC 1981 *Clause:* 3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation needs to sends IPv6 packets. The implementation initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. The implementation sends packets at the (known) MTU of the first hop in the path. Any of the packets sent on that path are too large to be forwarded by some implementation along the path.

Requirement: The implementation receives an ICMPv6 Packet Too Big message sent by the implementation that was unable to forward the too large packet.

RFC text: This memo describes a technique to dynamically discover the PMTU of a path. The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. `{{If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages [ICMPv6]}}`. Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message.

RQ_COR_1808 PMTU Discovery

RFC 1981 *Clause:* 3 ¶1, 4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation needs to sends IPv6 packets. The implementation initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. The implementation sends packets at the (known) MTU of the first hop in the path. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet.

Requirement: The implementation reduces its assumed PMTU for the path, based on the MTU field reported in the Packet Too Big message from the constricting hop.

RFC text: This memo describes a technique to dynamically discover the PMTU of a path. The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages [ICMPv6]. `{{Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message}}`. `{{.When a node receives a Packet Too Big message, it MUST reduce its estimate of the PMTU for the relevant path, based on the value of the MTU field in the message. The precise behavior of a node in this circumstance is not specified, since different applications may have different requirements, and since different implementation architectures may favor different strategies}}`.

RQ_COR_1809 PMTU Discovery

RFC 1981 *Clause:* 3 ¶2-3 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation needs to sends IPv6 packets. The implementation initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. The implementation sends packets at the (known) MTU of the first hop in the path. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message. The cycle of packet-sent/Packet-Too-Big-message-received is done may be several times.

Requirement: The implementation ends this cycle when the implementation estimates of the PMTU is less than or equal to the actual PMTU.

RFC text: {{The Path MTU Discovery process ends when the node's estimate of the PMTU is less than or equal to the actual PMTU. Note that several iterations of the packet-sent/Packet-Too-Big-message-received cycle may occur before the Path MTU Discovery process ends, as there may be links with smaller MTUs further along the path}}. Alternatively, the node may elect to end the discovery process by ceasing to send packets larger than the IPv6 minimum link MTU.

RQ_COR_1810 PMTU Discovery

RFC 1981 *Clause:* 3 ¶2-3 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation needs to sends IPv6 packets. The implementation initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. The implementation sends packets at the (known) MTU of the first hop in the path. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message. The cycle of packet-sent/Packet-Too-Big-message-received is done may be several times.

Requirement: The implementation elects to end the discovery process by ceasing to send packets larger than the IPv6 minimum link MTU.

RFC text: The Path MTU Discovery process ends when the node's estimate of the PMTU is less than or equal to the actual PMTU. Note that several iterations of the packet-sent/Packet-Too-Big-message-received cycle may occur before the Path MTU Discovery process ends, as there may be links with smaller MTUs further along the path. {{Alternatively, the node may elect to end the discovery process by ceasing to send packets larger than the IPv6 minimum link MTU}}.

RQ_COR_1811

RFC 1981 *Clause:* 3 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The PMTU of a path may change over time, due to changes in the routing topology. There exists a reduction of the PMTU.

Requirement: The implementation detects the reduction by receiving Packet Too Big messages.

RFC text: The PMTU of a path may change over time, due to changes in the routing topology. {{Reductions of the PMTU are detected by Packet Too Big messages}}. To detect increases in a path's PMTU, a node periodically increases its assumed PMTU. This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed. Therefore, attempts to detect increases in a path's PMTU should be done infrequently. RQ_COR_1819.

RQ_COR_1812 PMTU Discovery

RFC 1981 *Clause:* 3 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. A PMTU has been established for data delivery.

Requirement: The implementation periodically but infrequently increases the PMTU during data delivery.

RFC text: The PMTU of a path may change over time, due to changes in the routing topology. Reductions of the PMTU are detected by Packet Too Big messages. `{{To detect increases in a path's PMTU, a node periodically increases its assumed PMTU. This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed}}`. Therefore, attempts to detect increases in a path's PMTU should be done infrequently.

RQ_COR_1813

RFC 1981 *Clause:* 3 ¶4, 4 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The PMTU of a path may change over time, due to changes in the routing topology. The implementation probes the increase in a path's PMTU, increases its assumed PMTU. This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed.

Requirement: The implementation attempts to detect increases in a path's PMTU at infrequent intervals.

RFC text: The PMTU of a path may change over time, due to changes in the routing topology. Reductions of the PMTU are detected by Packet Too Big messages. `{{To detect increases in a path's PMTU, a node periodically increases its assumed PMTU. This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed. Therefore, attempts to detect increases in a path's PMTU should be done infrequently}}`. `{{...Nodes MAY detect increases in PMTU, but because doing so requires sending packets larger than the current estimated PMTU, and because the likelihood is that the PMTU will not have increased, this MUST be done at infrequent intervals}}`. See RQ_COR_1820.

RQ_COR_1814 PMTU: Multicast PMTU [Discover]

RFC 1981 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery.

Requirement: The implementation uses Path MTU Discovery for Multicast as well as unicast destinations.

RFC text: `{{Path MTU Discovery supports multicast as well as unicast destinations}}`. In the case of a multicast destination, copies of a packet may traverse many different paths to many different nodes. Each path may have a different PMTU, and a single multicast packet may result in multiple Packet Too Big messages, each reporting a different next-hop MTU. The minimum PMTU value across the set of paths in use determines the size of subsequent packets sent to the multicast destination.

RQ_COR_1815 PMTU: Multicast PMTU [Discover]

RFC 1981 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery for Multicast destinations. The implementation sends a single multicast packet. This single multicast packet results in multiple Packet Too Big messages, each reporting a different next-hop MTU.

Requirement: The implementation uses the minimum PMTU value across the set of paths in order to determine the final PMTU.

RFC text: Path MTU Discovery supports multicast as well as unicast destinations. `{{In the case of a multicast destination, copies of a packet may traverse many different paths to many different nodes. Each path may have a different PMTU, and a single multicast packet may result in multiple Packet Too Big messages, each reporting a different next-hop MTU. The minimum PMTU value across the set of paths in use determines the size of subsequent packets sent to the multicast destination}}`.

RQ_COR_1816 PMTU Discovery

RFC 1981 *Clause:* 3 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation "thinks" that a certain destination is attached to the same link as itself.

Requirement: The implementation performs the Path MTU Discovery even in this case, since is possible that the destination is not really in the same link.

RFC text: `{{Note that Path MTU Discovery must be performed even in cases where a node "thinks" a destination is attached to the same link as itself}}`. In a situation such as when a neighboring router acts as proxy [ND] for some destination, the destination can to appear to be directly connected but is in fact more than one hop away.

RQ_COR_1817

RFC 1981 *Clause:* 4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. When the implementation receives a Packet Too Big message, it MUST reduce its estimate of the PMTU for the relevant path, based on the value of the MTU field in the message.

Requirement: The implementation' precise behavior in this circumstance is not specified, since different applications may have different requirements, and since different implementation architectures may favor different strategies.

RFC text: `{{When a node receives a Packet Too Big message, it MUST reduce its estimate of the PMTU for the relevant path, based on the value of the MTU field in the message. The precise behavior of a node in this circumstance is not specified, since different applications may have different requirements, and since different implementation architectures may favor different strategies}}`. See RQ_COR_1808.

RQ_COR_1818 PMTU Discovery

RFC 1981 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. When the implementation receives a Packet Too Big message, it MUST reduce its estimate of the PMTU for the relevant path, based on the value of the MTU field in the message.

Requirement: The implementation reduces the size of the packets it is sending along the path.

RFC text: After receiving a Packet Too Big message, a node MUST attempt to avoid eliciting more such messages in the near future. `{{The node MUST reduce the size of the packets it is sending along the path}}`. Using a PMTU estimate larger than the IPv6 minimum link MTU may continue to elicit Packet Too Big messages. Since each of these messages (and the dropped packets they respond to) consume network resources, the node MUST force the Path MTU Discovery process to end. See RQ_COR_1808.

RQ_COR_1819

RFC 1981 *Clause:* 4 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The PMTU of a path may change over time, due to changes in the routing topology. There exists a reduction of the PMTU.

Requirement: The implementation detects the reduction in PMTU as fast as possible.

RFC text: `{{Nodes using Path MTU Discovery MUST detect decreases in PMTU as fast as possible}}`. See RQ_COR_1811.

RQ_COR_1820

RFC 1981 *Clause:* 4 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The PMTU of a path may change over time, due to changes in the routing topology. The implementation probes the increase in a path's PMTU, increases its assumed PMTU. [This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed]. The implementation attempts to detect increases in a path's PMTU at infrequent intervals.

Requirement: The implementation attempts to detect increases in a path's PMTU at least 5 minutes after a Packet Too Big message has been received for the given path. The recommended setting for this timer is twice its minimum value (10 minutes).

RFC text: Nodes MAY detect increases in PMTU, but because doing so requires sending packets larger than the current estimated PMTU, and because the likelihood is that the PMTU will not have increased, this MUST be done at infrequent intervals. `{{An attempt to detect an increase (by sending a packet larger than the current estimate) MUST NOT be done less than 5 minutes after a Packet Too Big message has been received for the given path. The recommended setting for this timer is twice its minimum value (10 minutes)}}`. See RQ_COR_1813.

RQ_COR_1821 PMTU Discovery

RFC 1981 *Clause:* 4 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery.

Requirement: The implementation does not reduce its estimate of the Path MTU below the IPv6 minimum link MTU.

RFC text: `{{A node MUST NOT reduce its estimate of the Path MTU below the IPv6 minimum link MTU}}`.

RQ_COR_1822 PMTU Discovery

RFC 1981 *Clause:* 4 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU.

Requirement: The implementation is not required to reduce the size of subsequent packets sent on the path to less than the IPv6 minimun link MTU, but rather it includes a Fragment header in those packets.

RFC text: {{Note: A node may receive a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU. In that case, the node is not required to reduce the size of subsequent packets sent on the path to less than the IPv6 minimun link MTU, but rather must include a Fragment header in those packets [IPv6- SPEC]}}.

RQ_COR_1823 PMTU Discovery

RFC 1981 *Clause:* 4 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message purporting to announce an increase in the Path MTU.

Requirement: The implementation does not increase its estimate of the Path MTU in response to the contents of a Packet Too Big message.

RFC text: {{A node MUST NOT increase its estimate of the Path MTU in response to the contents of a Packet Too Big message}}. A message purporting to announce an increase in the Path MTU might be a stale packet that has been floating around in the network, a false packet injected as part of a denial-of-service attack, or the result of having multiple paths to the destination, each with a different PMTU.

RQ_COR_1824 PMTU Discovery

RFC 1981 *Clause:* 5 ¶1-3 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery.

Requirement: The implementation follows the statements of RFC 1981, 5 provided as an aid for implementors.

RFC text: {{This section [5] discusses a number of issues related to the implementation of Path MTU Discovery. This is not a specification, but rather a set of notes provided as an aid for implementors. The issues include: - What layer or layers implement Path MTU Discovery?. - How is the PMTU information cached?. - How is stale PMTU information removed? - What must transport and higher layers do?}}.

RQ_COR_1825 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery.

Requirement: [INFORMATIVE] The implementation associates a PMTU value with a specific path traversed by packets exchanged between the source and destination implementations.[INFORMATIVE].

RFC text: {{Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes}}. However, in most cases a node will not have enough information to completely and accurately identify such a path. Rather, a node must associate a PMTU value with some local representation of a path. It is left to the implementation to select the local representation of a path.

RQ_COR_1826 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation does not have enough information to completely and accurately identify such a path.

Requirement: [INFORMATIVE] The implementation associates a PMTU value with some local representation of a path.[INFORMATIVE].

RFC text: `{{Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. However, in most cases a node will not have enough information to completely and accurately identify such a path. Rather, a node must associate a PMTU value with some local representation of a path. It is left to the implementation to select the local representation of a path}}`.

RQ_COR_1827 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. In the case of a multicast destination address, copies of a packet may traverse many different paths to reach many different implementations.

Requirement: [INFORMATIVE] The implementation associates a PMTU value with some adequate local representation of the "path" to a multicast destination representing in fact a potentially large set of paths. [INFORMATIVE].

RFC text: Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. However, in most cases a node will not have enough information to completely and accurately identify such a path. Rather, a node must associate a PMTU value with some local representation of a path. It is left to the implementation to select the local representation of a path. `{{In the case of a multicast destination address, copies of a packet may traverse many different paths to reach many different nodes. The local representation of the "path" to a multicast destination must in fact represent a potentially large set of paths}}`.

RQ_COR_1828 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1, 3 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation associates a PMTU value with some local representation of a path.

Requirement: [INFORMATIVE] The implementation at least maintains a single PMTU value to be used for all packets originated from the implementation itself.[INFORMATIVE].

RFC text: Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. `{{ Minimally, an implementation could maintain a single PMTU value to be used for all packets originated from the node}}`. This PMTU value would be the minimum PMTU learned across the set of all paths in use by the node. This approach is likely to result in the use of smaller packets than is necessary for many paths.

RQ_COR_1829 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1, 3 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementatation associates a PMTU value with some local representation of a path. The implementation at least maintains a single PMTU value to be used for all packets originated from the implementation itself.

Requirement: [INFORMATIVE] The implementation uses as the single PMTU value the minimum PMTU learned across the set of all paths in use by the implementation.[INFORMATIVE].

RFC text: Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. `{{...Minimally, an implementation could maintain a single PMTU value to be used for all packets originated from the node. This PMTU value would be the minimum PMTU learned across the set of all paths in use by the node}}`. This approach is likely to result in the use of smaller packets than is necessary for many paths.

RQ_COR_1830 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1, 4 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementatation associates a PMTU value with some local representation of a path.

Requirement: [INFORMATIVE] The implementation uses the destination address as the local representation of a path.[INFORMATIVE].

RFC text: Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes... `{{...An implementation could use the destination address as the local representation of a path}}`. The PMTU value associated with a destination would be the minimum PMTU learned across the set of all paths in use to that destination. The set of paths in use to a particular destination is expected to be small, in many cases consisting of a single path. This approach will result in the use of optimally sized packets on a per-destination basis. This approach integrates nicely with the conceptual model of a host as described in [ND]: a PMTU value could be stored with the corresponding entry in the destination cache.

RQ_COR_1831 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1, 4 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementatation associates a PMTU value with some local representation of a path. The implementation uses the destination address as the local representation of a path.

Requirement: [INFORMATIVE] The implementation associates that destination address with a PMTU value equal to the minimum PMTU learned across the set of all paths in use to that destination.[INFORMATIVE].

RFC text: Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. An implementation could use the destination address as the local representation of a path. `{{The PMTU value associated with a destination would be the minimum PMTU learned across the set of all paths in use to that destination. The set of paths in use to a particular destination is expected to be small, in many cases consisting of a single path. This approach will result in the use of optimally sized packets on a per-destination basis}}`. This approach integrates nicely with the conceptual model of a host as described in [ND]: a PMTU value could be stored with the corresponding entry in the destination cache.

RQ_COR_1832 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1, 5 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementatation associates a PMTU value with some local representation of a path. The implementation uses IPv6 Flow Labels.

Requirement: [INFORMATIVE] The implementation uses the Flow id as the local representation of a path. This approach will result in the use of optimally sized packets on a per-flow basis, providing finer granularity than PMTU values maintained on a per-destination basis.[INFORMATIVE]

RFC text: Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes. `{{...If flows [IPv6-SPEC] are in use, an implementation could use the flow id as the local representation of a path}}`. Packets sent to a particular destination but belonging to different flows may use different paths, with the choice of path depending on the flow id. This approach will result in the use of optimally sized packets on a per-flow basis, providing finer granularity than PMTU values maintained on a per-destination basis.

RQ_COR_1833 PMTU-Path Association

RFC 1981 *Clause:* 5.2 ¶1, 6 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementatation associates a PMTU value with some local representation of a path. The implementation uses source routed packets (i.e. packets containing an IPv6 Routing header [IPv6-SPEC]).

Requirement: [INFORMATIVE] The implementation uses the source route information in the local representation of a path.[INFORMATIVE].

RFC text: Ideally, a PMTU value should be associated with a specific path traversed by packets exchanged between the source and destination nodes... `{{...For source routed packets (i.e. packets containing an IPv6 Routing header [IPv6-SPEC]), the source route may further qualify the local representation of a path. In particular, a packet containing a type 0 Routing header in which all bits in the Strict/Loose Bit Map are equal to 1 contains a complete path specification. An implementation could use source route information in the local representation of a path}}`.

RQ_COR_1834 PMTU-Path Association using Packet Too Big

RFC 1981 *Clause:* 5.2 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet.

Requirement: [INFORMATIVE] The implementation determines which path the message applies to based on the contents of the Packet Too Big message.[INFORMATIVE].

RFC text: `{{When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message}}`. For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to.

RQ_COR_1835 PMTU-Path Association using Packet Too Big

RFC 1981 *Clause:* 5.2 ¶9 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation determines which path the message applies to based on the contents of the Packet Too Big message.

Requirement: [INFORMATIVE] The implementation uses the destination address from the original packet [which originates the Packet Too Big message] to determine which path the message applies to.[INFORMATIVE].

RFC text: When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message. `{{For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to}}`.

RQ_COR_1836 PMTU-Path Association using Packet Too Big

RFC 1981 *Clause:* 5.2 ¶9-10 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation determines which path the message applies to based on the contents of the Packet Too Big message. The original packet [which originates the Packet Too Big message] contained a Routing header.

Requirement: [INFORMATIVE] The implementation uses the Routing header to determine the location of the destination address within the original packet.[INFORMATIVE].

RFC text: When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message. For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to. `{{Note: if the original packet contained a Routing header, the Routing header should be used to determine the location of the destination address within the original packet}}`. If Segments Left is equal to zero, the destination address is in the Destination Address field in the IPv6 header. If Segments Left is greater than zero, the destination address is the last address (Address[n]) in the Routing header.

RQ_COR_1837 PMTU-Path Association using Packet Too Big

RFC 1981 *Clause:* 5.2 ¶9-10 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation determines which path the message applies to based on the contents of the Packet Too Big message. The original packet [which originates the Packet Too Big message] contained a Routing header. The implementation uses the Routing header to determine the location of the destination address within the original packet. The Segments Left in Routing header is equal to zero.

Requirement: [INFORMATIVE] The implementation gets the the destination address from the Destination Address field in the IPv6 header.[INFORMATIVE].

RFC text: When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message. For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to. Note: if the original packet contained a Routing header, the Routing header should be used to determine the location of the destination address within the original packet. `{{If Segments Left is equal to zero, the destination address is in the Destination Address field in the IPv6 header}}`. If Segments Left is greater than zero, the destination address is the last address (Address[n]) in the Routing header.

RQ_COR_1838 PMTU-Path Association using Packet Too Big

RFC 1981 *Clause:* 5.2 ¶9-10 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation determines which path the message applies to based on the contents of the Packet Too Big message. The original packet [which originates the Packet Too Big message] contained a Routing header. The implementation uses the Routing header to determine the location of the destination address within the original packet. The Segments Left in Routing header is is greater than zero.

Requirement: [INFORMATIVE] The implementation gets the the destination address from the last address (Address[n]) in the Routing header.[INFORMATIVE].

RFC text: When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message. For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to. Note: if the original packet contained a Routing header, the Routing header should be used to determine the location of the destination address within the original packet. If Segments Left is equal to zero, the destination address is in the Destination Address field in the IPv6 header. `{{If Segments Left is greater than zero, the destination address is the last address (Address[n]) in the Routing header}}`.

RQ_COR_1839 PMTU-Path Association using Packet Too Big

RFC 1981 *Clause:* 5.2 ¶9-11 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation determines which path the message applies to based on the contents of the Packet Too Big message. Then, the implementation uses the value in the MTU field in the Packet Too Big message as a tentative PMTU value, and compares the tentative PMTU to the existing PMTU, resulting that the tentative PMTU is lower than the existing PMTU estimate.

Requirement: [INFORMATIVE] The implementation replaces the tentative PMTU over the existing PMTU as the PMTU value for the path.[INFORMATIVE].

RFC text: `{{The node then uses the value in the MTU field in the Packet Too Big message as a tentative PMTU value, and compares the tentative PMTU to the existing PMTU. If the tentative PMTU is less than the existing PMTU estimate, the tentative PMTU replaces the existing PMTU as the PMTU value for the path}}`.

RQ_COR_1840 PMTU-Path Association using Packet Too Big

RFC 1981 *Clause:* 5.2 ¶9-11 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation determines which path the message applies to based on the contents of the Packet Too Big message. Then, the implementation uses the value in the MTU field in the Packet Too Big message as a tentative PMTU value, and compares the tentative PMTU to the existing PMTU, resulting that the tentative PMTU is higher than the existing PMTU estimate.

Requirement: [INFORMATIVE] The implementation does not replacethe existing PMTU.[INFORMATIVE].

RFC text: `{{The node then uses the value in the MTU field in the Packet Too Big message as a tentative PMTU value, and compares the tentative PMTU to the existing PMTU. If the tentative PMTU is less than the existing PMTU estimate, the tentative PMTU replaces the existing PMTU as the PMTU value for the path}}`.

RQ_COR_1841 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶12 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation reduces its estimate PMTU for the path.

Requirement: [INFORMATIVE] The implementation notifies any Packetization layer about decreases in the PMTU that is actively using the path. [INFORMATIVE].

RFC text: `{{The packetization layers must be notified about decreases in the PMTU. Any packetization layer instance (for example, a TCP connection) that is actively using the path must be notified if the PMTU estimate is decreased}}`.

RQ_COR_1842 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶12-13 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation reduces its estimate PMTU for the path. The implementation notifies about decreases in the PMTU to any Packetization layer that is actively using the path. The Packet Too Big message contains an Original Packet Header that refers only to UDP packets, but there are TCP connections using the given path.

Requirement: [INFORMATIVE] The implementation notifies about decreases in the PMTU also to the TCP layer.[INFORMATIVE].

RFC text: The packetization layers must be notified about decreases in the PMTU. Any packetization layer instance (for example, a TCP connection) that is actively using the path must be notified if the PMTU estimate is decreased. `{{Note: even if the Packet Too Big message contains an Original Packet Header that refers to a UDP packet, the TCP layer must be notified if any of its connections use the given path}}`.

RQ_COR_1843 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶14 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet.

Requirement: [INFORMATIVE] The implementation notifies to the instance that sent the packet that produced the Packet Too Big message that its packet has been dropped, so that the instance may retransmit the dropped data.[INFORMATIVE].

RFC text: `{{Also, the instance that sent the packet that elicited the Packet Too Big message should be notified that its packet has been dropped, even if the PMTU estimate has not changed, so that it may retransmit the dropped data}}`.

RQ_COR_1844 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶15 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation reduces its estimate PMTU for the path.

Requirement: [INFORMATIVE] The implementation avoidS the use of an asynchronous notification mechanism for PMTU decreases by postponing notification until the next attempt to send a packet larger than the PMTU estimate.[INFORMATIVE].

RFC text: `{{Note: An implementation can avoid the use of an asynchronous notification mechanism for PMTU decreases by postponing notification until the next attempt to send a packet larger than the PMTU estimate}}`. In this approach, when an attempt is made to SEND a packet that is larger than the PMTU estimate, the SEND function should fail and return a suitable error indication. This approach may be more suitable to a connectionless packetization layer (such as one using UDP), which (in some implementations) may be hard to "notify" from the ICMP layer. In this case, the normal timeout-based retransmission mechanisms would be used to recover from the dropped packets.

RQ_COR_1845 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶15 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation reduces its estimateD PMTU for the path. The implementation avoid the use of an asynchronous notification mechanism for PMTU decreases by postponing notification until the next attempt to send a packet larger than the PMTU estimate. An implementation's upper layer attempts to SEND a packet that is larger than the PMTU estimate.

Requirement: [INFORMATIVE] The implementation makes the SEND function fails and returns a suitable error indication.[INFORMATIVE].

RFC text: Note: An implementation can avoid the use of an asynchronous notification mechanism for PMTU decreases by postponing notification until the next attempt to send a packet larger than the PMTU estimate. `{{In this approach, when an attempt is made to SEND a packet that is larger than the PMTU estimate, the SEND function should fail and return a suitable error indication}}`. This approach may be more suitable to a connectionless packetization layer (such as one using UDP), which (in some implementations) may be hard to "notify" from the ICMP layer. In this case, the normal timeout-based retransmission mechanisms would be used to recover from the dropped packets.

RQ_COR_1846 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶16 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation reduces its estimate PMTU for the path. The implementation notifies about decreases in the PMTU to any Packetization layer that is actively using the path.

Requirement: [INFORMATIVE] The implementation delays the notify until the Packetization layer instance wants to create a new packet.[INFORMATIVE].

RFC text: `{{It is important to understand that the notification of the packetization layer instances using the path about the change in the PMTU is distinct from the notification of a specific instance that a packet has been dropped. The latter should be done as soon as practical (i.e., asynchronously from the point of view of the packetization layer instance), while the former may be delayed until a packetization layer instance wants to create a packet}}`. See RQ_COR_1841.

RQ_COR_1847 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶16 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation notifies to the instance that sent the packet that produced the Packet Too Big message that its packet has been dropped, so that the instance may retransmit the dropped data.

Requirement: [INFORMATIVE] The implementation notifies the instance that sent the packets as soon as practical (i.e., asynchronously from the point of view of the Packetization layer instance).[INFORMATIVE].

RFC text: `{{It is important to understand that the notification of the packetization layer instances using the path about the change in the PMTU is distinct from the notification of a specific instance that a packet has been dropped. The latter should be done as soon as practical (i.e., asynchronously from the point of view of the packetization layer instance), while the former may be delayed until a packetization layer instance wants to create a packet}}`. See RQ_COR_1843.

RQ_COR_1848 PMTU: Informing Packetization Layers of PMTU

RFC 1981 *Clause:* 5.2 ¶16 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives an ICMPv6 Packet Too Big message sent by an intermediate implementation that was unable to forward the too large packet. The implementation notifies to the instance that sent the packet that produced the Packet Too Big message that its packet has been dropped, so that the instance may retransmit the dropped data.

Requirement: [INFORMATIVE] The implementation does retransmission only for those packets that are known to be dropped as indicated by a Packet Too Big message.[INFORMATIVE].

RFC text: It is important to understand that the notification of the packetization layer instances using the path about the change in the PMTU is distinct from the notification of a specific instance that a packet has been dropped. The latter should be done as soon as practical (i.e., asynchronously from the point of view of the packetization layer instance), while the former may be delayed until a packetization layer instance wants to create a packet. `{{Retransmission should be done for only for those packets that are known to be dropped, as indicated by a Packet Too Big message}}`.

RQ_COR_1849 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time.

Requirement: [INFORMATIVE] The implementation's cached PMTU information becomes stale.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`.

RQ_COR_1850

RFC 1981 *Clause:* 5.3 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too large.

Requirement: [INFORMATIVE] The implemention will discover this almost immediately once a large enough packet is sent on the path.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. `{{If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path}}`. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values. When a PMTU value has not been decreased for a while (on the order of 10 minutes), the PMTU estimate should be set to the MTU of the first-hop link, and the packetization layers should be notified of the change. This will cause the complete Path MTU Discovery process to take place again.

RQ_COR_1851

RFC 1981 *Clause:* 5.3 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small.

Requirement: [INFORMATIVE] The implemention does not have a mechanism for realizing that a stale PMTU value is too small.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. `{{No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values}}`. When a PMTU value has not been decreased for a while (on the order of 10 minutes), the PMTU estimate should be set to the MTU of the first-hop link, and the packetization layers should be notified of the change. This will cause the complete Path MTU Discovery process to take place again.

RQ_COR_1852 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-2 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small.

Requirement: [INFORMATIVE] The implementation "ages" cached values.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. `{{No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values}}`. When a PMTU value has not been decreased for a while (on the order of 10 minutes), the PMTU estimate should be set to the MTU of the first-hop link, and the packetization layers should be notified of the change. This will cause the complete Path MTU Discovery process to take place again.

RQ_COR_1853 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-2 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small, then the implementation "ages" cached values. The implementation's PMTU value has not been decreased for a while (on the order of 10 minutes).

Requirement: [INFORMATIVE] The implementation's PMTU estimate is set to the MTU of the first-hop link, and the packetization layers are notified of the change.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values. `{{When a PMTU value has not been decreased for a while (on the order of 10 minutes), the PMTU estimate should be set to the MTU of the first-hop link, and the packetization layers should be notified of the change}}`. This will cause the complete Path MTU Discovery process to take place again.

RQ_COR_1854 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small, then the implementation "ages" cached values. The implementation's PMTU value has not been decreased for a while (on the order of 10 minutes). The implementation's PMTU estimate is set to the MTU of the first-hop link, and the packetization layers are notified of the change.

Requirement: [INFORMATIVE] The implementation starts again the complete Path MTU Discovery process. [INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values. `{{When a PMTU value has not been decreased for a while (on the order of 10 minutes), the PMTU estimate should be set to the MTU of the first-hop link, and the packetization layers should be notified of the change. This will cause the complete Path MTU Discovery process to take place again}}`.

RQ_COR_1855 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-3 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small, then the implementation "ages" cached values. The implementation's PMTU value has not been decreased for a while (on the order of 10 minutes), after that time the PMTU estimate ussually is set to the MTU of the first-hop link, and the packetization layers is notified of the change.

Requirement: [INFORMATIVE] The implementation has a means for changing the timeout duration, including setting it to "infinity". [INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values. When a PMTU value has not been decreased for a while (on the order of 10 minutes), the PMTU estimate should be set to the MTU of the first-hop link, and the packetization layers should be notified of the change. This will cause the complete Path MTU Discovery process to take place again. `{{Note: an implementation should provide a means for changing the timeout duration, including setting it to "infinity"}}`. For example, nodes attached to an FDDI link which is then attached to the rest of the Internet via a small MTU serial line are never going to discover a new non-local PMTU, so they should not have to put up with dropped packets every 10 minutes.

RQ_COR_1856 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-3, 5 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small, then the implementation "ages" cached values.

Requirement: [INFORMATIVE] The implementation associates a timestamp field with a PMTU value as one approach to implementing PMTU aging.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values}}. `{{...One approach to implementing PMTU aging is to associate a timestamp field with a PMTU value}}`. This field is initialized to a "reserved" value, indicating that the PMTU is equal to the MTU of the first hop link. Whenever the PMTU is decreased in response to a Packet Too Big message, the timestamp is set to the current time.

RQ_COR_1857 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-3, 5 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small, then the implementation "ages" cached values. The implementation associates a timestamp field with a PMTU value as one approach to implementing PMTU aging.

Requirement: [INFORMATIVE] The implementation initializes this timestamp field to a "reserved" value indicating that the PMTU is equal to the MTU of the first hop link.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values}}. ...One approach to implementing PMTU aging is to associate a timestamp field with a PMTU value. `{{This field is initialized to a "reserved" value, indicating that the PMTU is equal to the MTU of the first hop link}}`. Whenever the PMTU is decreased in response to a Packet Too Big message, the timestamp is set to the current time.

RQ_COR_1858 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-3, 5 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small, then the implementation "ages" cached values. The implementation associates a timestamp field with a PMTU value as one approach to implementing PMTU aging. The implementation initializes this timestamp field to a "reserved" value, indicating that the PMTU is equal to the MTU of the first hop link. The PMTU is decreased in response to a Packet Too Big message.

Requirement: [INFORMATIVE] The timestamp field is set to the current time. [INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values}}. One approach to implementing PMTU aging is to associate a timestamp field with a PMTU value. `{{This field is initialized to a "reserved" value, indicating that the PMTU is equal to the MTU of the first hop link. Whenever the PMTU is decreased in response to a Packet Too Big message, the timestamp is set to the current time}}`.

RQ_COR_1859 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.3 ¶1-3, 5-7 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. Internetwork topology is dynamic; routes change over time. The implementation's local representation of a path remains constant, but the actual path(s) in use changes and the PMTU information cached by a node can become stale. The implementation' stale PMTU value is too small. The implemention does not have a mechanism for realizing that a stale PMTU value is too small, then the implementation "ages" cached values. The implementation associates a timestamp field with a PMTU value as one approach to implementing PMTU aging. The implementation initializes this timestamp field to a "reserved" value, indicating that the PMTU is equal to the MTU of the first hop link. The PMTU is decreased in response to a Packet Too Big message. The timestamp field is set to the current time.

Requirement: [INFORMATIVE] The implementation, once a minute, runs a timer-driven procedure through all cached PMTU values, and for each PMTU whose timestamp is not "reserved" and is older than the timeout interval: - The PMTU estimate is set to the MTU of the first hop link. - The timestamp is set to the "reserved" value. - Packetization layers using this path are notified of the increase.[INFORMATIVE].

RFC text: Internetwork topology is dynamic; routes change over time. `{{While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale}}`. If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values}}. ...One approach to implementing PMTU aging is to associate a timestamp field with a PMTU value. This field is initialized to a "reserved" value, indicating that the PMTU is equal to the MTU of the first hop link. Whenever the PMTU is decreased in response to a Packet Too Big message, the timestamp is set to the current time. `{{Once a minute, a timer-driven procedure runs through all cached PMTU values, and for each PMTU whose timestamp is not "reserved" and is older than the timeout interval: - The PMTU estimate is set to the MTU of the first hop link. - The timestamp is set to the "reserved" value. - Packetization layers using this path are notified of the increase}}`.

RQ_COR_1860 PMTU: Selecting PMTU Discovery

RFC 1981 *Clause:* 5.6 ¶1-3 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery.

Requirement: [INFORMATIVE] The implementation provides a way for a system utility program to specify that Path MTU Discovery not be done on a given path.[INFORMATIVE].

RFC text: `{{It is suggested that an implementation provide a way for a system utility program to: - Specify that Path MTU Discovery not be done on a given path }}`. - Change the PMTU value associated with a given path. The former can be accomplished by associating a flag with the path; when a packet is sent on a path with this flag set, the IP layer does not send packets larger than the IPv6 minimum link MTU. These features might be used to work around an anomalous situation, or by a routing protocol implementation that is able to obtain Path MTU values. The implementation should also provide a way to change the timeout period for aging stale PMTU information.

RQ_COR_1861 PMTU: Selecting PMTU Discovery

RFC 1981 *Clause:* 5.6 ¶1-3 *Type:* MAY *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation provides a way for a system utility program to specify that Path MTU Discovery not be done on a given path.

Requirement: [INFORMATIVE] The implementation associates a flag with the path.[INFORMATIVE].

RFC text: `{{It is suggested that an implementation provide a way for a system utility program to: - Specify that Path MTU Discovery not be done on a given path }}`. - Change the PMTU value associated with a given path. `{{The former can be accomplished by associating a flag with the path}}`; when a packet is sent on a path with this flag set, the IP layer does not send packets larger than the IPv6 minimum link MTU. These features might be used to work around an anomalous situation, or by a routing protocol implementation that is able to obtain Path MTU values. The implementation should also provide a way to change the timeout period for aging stale PMTU information.

RQ_COR_1862 PMTU: Selecting PMTU Discovery

RFC 1981 *Clause:* 5.6 ¶1-3 *Type:* MUST *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation provides a way for a system utility program to specify that Path MTU Discovery not be done on a given path. The implementation associates a flag with the path. A packet is sent on a path with this flag set.

Requirement: [INFORMATIVE] The implementation's IP layer does not send packets larger than the IPv6 minimum link MTU.[INFORMATIVE].

RFC text: `{{It is suggested that an implementation provide a way for a system utility program to: - Specify that Path MTU Discovery not be done on a given path }}`. - Change the PMTU value associated with a given path. `{{The former can be accomplished by associating a flag with the path; when a packet is sent on a path with this flag set, the IP layer does not send packets larger than the IPv6 minimum link MTU}}`. These features might be used to work around an anomalous situation, or by a routing protocol implementation that is able to obtain Path MTU values. The implementation should also provide a way to change the timeout period for aging stale PMTU information.

RQ_COR_1863 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.6 ¶1-3 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery.

Requirement: [INFORMATIVE] The implementation provides a way for a system utility program to change the PMTU value associated with a given path.[INFORMATIVE].

RFC text: {{It is suggested that an implementation provide a way for a system utility program to}}: - Specify that Path MTU Discovery not be done on a given path. - {{Change the PMTU value associated with a given path}}. The former can be accomplished by associating a flag with the path; when a packet is sent on a path with this flag set, the IP layer does not send packets larger than the IPv6 minimum link MTU. These features might be used to work around an anomalous situation, or by a routing protocol implementation that is able to obtain Path MTU values.

RQ_COR_1864 PMTU: Updating Cache Information

RFC 1981 *Clause:* 5.6 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery.

Requirement: [INFORMATIVE] The implementation provides a way to change the timeout period for aging stale PMTU information.[INFORMATIVE].

RFC text: {{The implementation should also provide a way to change the timeout period for aging stale PMTU information}}.

RQ_COR_1865

RFC 1981 *Clause:* 6 ¶1-2 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives (from a malicious party) false Packet Too Big messages indicating a PMTU much smaller than reality.

Requirement: [INFORMATIVE] The implementation does not entirely stop data flow, since the implementation never set its PMTU estimate below the IPv6 minimum link MTU.[INFORMATIVE].

RFC text: {{This Path MTU Discovery mechanism makes possible two denial-of-service attacks, both based on a malicious party sending false Packet Too Big messages to a node}}. {{In the first attack, the false message indicates a PMTU much smaller than reality. This should not entirely stop data flow, since the victim node should never set its PMTU estimate below the IPv6 minimum link MTU. It will, however, result in suboptimal performance}}.

RQ_COR_1866

RFC 1981 *Clause:* 6 ¶1, 3 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Path MTU Discovery. The implementation receives (from a malicious party) false Packet Too Big messages indicating a PMTU larger than reality.

Requirement: [INFORMATIVE] The implementation never raises its estimate of the PMTU based on a Packet Too Big message, so it does not be vulnerable to this attack.[INFORMATIVE].

RFC text: `{{This Path MTU Discovery mechanism makes possible two denial-of-service attacks, both based on a malicious party sending false Packet Too Big messages to a node}}`. `{{In the second attack, the false message indicates a PMTU larger than reality. If believed, this could cause temporary blockage as the victim sends packets that will be dropped by some router. Within one round-trip time, the node would discover its mistake (receiving Packet Too Big messages from that router), but frequent repetition of this attack could cause lots of packets to be dropped. A node, however, should never raise its estimate of the PMTU based on a Packet Too Big message, so should not be vulnerable to this attack}}`.

4.6 Requirements extracted from RFC 2460

RQ_COR_1000 IPv6 Header [Generate]

RFC 2460 *Clause:* 3 ¶4 *Type:* recommended *applies to:* Node

Context: The implementation generates an IPv6 packet.

Requirement: The value of the Payload Length Field in the IPv6 Header is the packet's total length minus the IPv6 header's length.

RFC text: Payload Length 16-bit unsigned integer. `{{Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets}}`. (Note that any extension headers [section 4] present are considered part of the payload, i.e., included in the length count).

RQ_COR_1001 IPv6 Header [Generate]

RFC 2460 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation generates an IPv6 packet.

Requirement: The value of the Next Header Field in the IPv6 Header is the type of header immediately following the IPv6 header.

RFC text: Next Header 8-bit selector. `{{Identifies the type of header immediately following the IPv6 header}}`. Uses the same values as the IPv4 Protocol field [RFC-1700 et seq.].

RQ_COR_1002 Hop Limit [Process]

RFC 2460 *Clause:* 3 ¶6 *Type:* MUST *applies to:* Router

Context: The implementation receives an IPv6 packet with a Hop Limit greater than 0.

Requirement: The implementation decrements by 1 the Hop Limit field and forwards the packet.

RFC text: Hop Limit 8-bit unsigned integer. `{{Decremented by 1 by each node that forwards the packet}}`. The packet is discarded if Hop Limit is decremented to zero.

RQ_COR_1004 Extension Headers [Process]

RFC 2460 *Clause:* 4 ¶2-3 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with extension headers other than the Hop-by-Hop Options header and the implementation's address is not the Destination Address field of the IPv6 header.

Requirement: The implementation does not examine nor process the extension headers.

RFC text: `{{With one exception, extension headers are not examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. ...The exception referred to in the preceding paragraph is the Hop-by- Hop Options header, ...}}.`

RQ_COR_1005 Extension Headers [Process]

RFC 2460 *Clause:* 4 ¶2-3 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with extension headers other than the Hop-by-Hop Options header and the implementation's address is the Destination Address field of the IPv6 header.

Requirement: Implementation examines and processes the extension headers.

RFC text: `{{With one exception, extension headers are not examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. ...The exception referred to in the preceding paragraph is the Hop-by- Hop Options header, ...}}.`

RQ_COR_1006 Extension Headers [Process]

RFC 2460 *Clause:* 4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet containing several extension headers.

Requirement: The implementation processes the extension headers strictly in the order they appear in the packet.

RFC text: Therefore, `{{extension headers must be processed strictly in the order they appear in the packet}}`; a receiver must not, for example, scan through a packet looking for a particular kind of extension header and process that header prior to processing all preceding ones.

RQ_COR_1007 Hop by Hop Header [Process]

RFC 2460 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with a Hop-by-Hop Options header. The Destination Address in the IPv6 Header is not the implementation's address.

Requirement: The implementation examines and processes the extension header.

RFC text: `{{ Hop-by-Hop Options header, which carries information that must be examined and processed by every node along a packet's delivery path, including the source and destination nodes}}`.

RQ_COR_1008 Hop by Hop Header [Generate]

RFC 2460 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Hop-by-Hop Options extension header.

Requirement: The Hop-by-Hop Options extension header immediately follows the IPv6 header.

RFC text: The exception referred to in the preceding paragraph is the Hop-by- Hop Options header, which carries information that must be examined and processed by every node along a packet's delivery path, including the source and destination nodes. `{{The Hop-by-Hop Options header, when present, must immediately follow the IPv6 header}}`.

RQ_COR_1009 Hop by Hop Header [Generate]

RFC 2460 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Hop-by-Hop Options extension header.

Requirement: The Hop-by-Hop Options extension header presence is indicated by the value zero in the Next Header field of the IPv6 header.

RFC text: `{{Its presence is indicated by the value zero in the Next Header field of the IPv6 header}}`.

RQ_COR_1010 Extension Headers [Process]

RFC 2460 *Clause:* 4 ¶4 *Type:* SHOULD *applies to:* Node

Context: An implementation is processing an extension header that contains an unrecognizable Next Header value.

Requirement: The implementation discards the packet and sends an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1 ("unrecognized Next Header type encountered") and the ICMP Pointer field containing the offset of the unrecognized value within the original packet.

RFC text: `{{If, as a result of processing a header, a node is required to proceed to the next header but the Next Header value in the current header is unrecognized by the node, it should discard the packet and send an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1 ("unrecognized Next Header type encountered") and the ICMP Pointer field containing the offset of the unrecognized value within the original packet}}`.

RQ_COR_1011 Extension Headers [Process]

RFC 2460 *Clause:* 4 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a packet with a Next Header value of zero in any header other than the IPv6 header.

Requirement: The implementation discards the packet and sends an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1 ("unrecognized Next Header type encountered") and the ICMP Pointer field containing the offset of the unrecognized value within the original packet.

RFC text: The same action should be taken if a `{{node encounters a Next Header value of zero in any header other than an IPv6 header}}`.

RQ_COR_1012 Extension Headers [Process]

RFC 2460 *Clause:* 4 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation transmits a packet with one or several extension headers.

Requirement: Each extension header is an integer multiple of 8 octets long. Multi-octet fields of width n octets within each extension header are placed at an integer multiple of n octets from the start of the header, for n = 1, 2, 4, or 8

RFC text: `{{Each extension header is an integer multiple of 8 octets long}}`, in order to retain 8-octet alignment for subsequent headers. `{{Multi- octet fields within each extension header are aligned on their natural boundaries, i.e., fields of width n octets are placed at an integer multiple of n octets from the start of the header, for n = 1, 2, 4, or 8.}}`

RQ_COR_1013 Extension Headers [Generate]

RFC 2460 *Clause:* 4.1 ¶1 *Type:* Recommended *applies to:* Node

Context: The implementation transmits a packet with more than one extension header.

Requirement: The extension headers appear in the following order: IPv6 header, Hop-by-Hop Options header, Destination Options header, Routing header, Fragment header, Authentication header, Encapsulating Security Payload header, Destination Options header, upper-layer header.

RFC text: `{{When more than one extension header is used in the same packet, it is recommended that those headers appear in the following order: IPv6 header, Hop-by-Hop Options header, Destination Options header (note 1), Routing header, Fragment header, Authentication header (note 2), Encapsulating Security Payload header (note 2), Destination Options header (note 3), upper-layer header}}`

RQ_COR_1014 Extension Headers [Generate]

RFC 2460 *Clause:* 4.1 ¶6 *Type:* SHOULD *applies to:* Node

Context: The implementation transmits a packet with more than one extension header.

Requirement: Extension headers occur at most once, except for the Destination Options header which occurs at most twice (once before a Routing header and once before the upper-layer header).

RFC text: `{{Each extension header should occur at most once, except for the Destination Options header which should occur at most twice (once before a Routing header and once before the upper-layer header)}}`.

RQ_COR_1015 Extension Headers [Generate]

RFC 2460 *Clause:* 4.1 ¶7 *Type:* Recommended *applies to:* Node

Context: The implementation transmits a packet where an upper-layer header is another IPv6 header with its own extension headers; i.e. IPv6 tunneled over or encapsulated in IPv6.

Requirement: The upper-layer IPv6 extension headers appear in the following order: IPv6 header, Hop-by-Hop Options header, Destination Options header, Routing header, Fragment header, Authentication header, Encapsulating Security Payload header, Destination Options header, upper-layer header. This order is the same as that for IPv6 packets that do not tunnel another IPv6 packet.

RFC text: `{{If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its own extension headers, which are separately (sic) subject to the same ordering recommendations}}`.

RQ_COR_1016 Extension Headers [Process]

RFC 2460 *Clause:* 4.1 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with more than one extension header with duplicated extension headers and the headers not arranged in the recommended order. The Hop-by-Hop Options header is the first extension header in the packet.

Requirement: The implementation accepts and attempts to process the duplicated and out-of-order extension headers.

RFC text: `{{IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the Hop-by-Hop Options header which is restricted to appear immediately after an IPv6 header only}}`. Nonetheless, it is strongly advised that sources of IPv6 packets adhere to the above recommended order until and unless subsequent specifications revise that recommendation.

RQ_COR_1017 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with a Hop-by-Hop Options header and/or Destination Options header(s) that carry a variable number of type-length-value (TLV) encoded "options".

Requirement: The implementation processes the sequence of options within each options header strictly in the order as they appear in the header.

RFC text: `{{The sequence of options within a header must be processed strictly in the order they appear in the header}}`; a receiver must not, for example, scan through the header looking for a particular kind of option and process that option prior to processing all preceding ones.

RQ_COR_1018 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header and does not recognize the Option Type. The highest-order two bits of the Option Type in the extension header are 00.

Requirement: The implementation skips over this option and continues processing the header.

RFC text: `{{The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type}}`.

RQ_COR_1019 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header and does not recognize the Option Type. The highest-order two bits of the Option Type in the extension header are 01.

Requirement: The implementation discards the packet.

RFC text: `{{The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type}}`.

RQ_COR_1020 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header and does not recognize the Option Type. The highest-order two bits of the Option Type in the extension header are 10. The Destination Address of the packet is not a multicast address.

Requirement: The implementation discards the packet and sends an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

RFC text: `{{The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type}}`.

RQ_COR_1021 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header and does not recognize the Option Type. The highest-order two bits of the Option Type in the extension header are 11. The Destination Address of the packet is not a multicast address.

Requirement: Implementation discards the packet and sends an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

RFC text: `{{The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type}}`.

RQ_COR_1022 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header and the third-highest-order bit of the Option Type in the extension header is set to 1. This signifies that the Option Data can change en-route to the packet's final destination. In addition, an Authentication header is present in the packet.

Requirement: The implementation treats the entire Option Data field as zero-valued octets when computing or verifying the packet's authenticating value.

RFC text: The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination. `{{When an Authentication header is present in the packet, for any option whose data may change en-route, its entire Option Data field must be treated as zero-valued octets when computing or verifying the packet's authenticating value}}`.

RQ_COR_1023 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header. The third-highest-order bit of the Option Type in the extension header is set 0.

Requirement: Implementation leaves untouched the Option Data of that option.

RFC text: `{{The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination}}`.

RQ_COR_1024 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶10 *Type:* MAY *applies to:* Node

Context: The implementation processes aa Hop-by-Hop or Destination Options extension header. The third-highest-order bit of the Option Type in the extension header is 1.

Requirement: Implementation changes the Option Data of that option.

RFC text: `{{The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination}}`.

RQ_COR_1025 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶11 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header.

Requirement: Implementation identifies a particular option using the full 8-bit Option Type, not just the low-order 5 bits of an Option Type.

RFC text: `{{The three high-order bits described above are to be treated as part of the Option Type, not independent of the Option Type. That is, a particular option is identified by a full 8-bit Option Type, not just the low-order 5 bits of an Option Type}}`.

RQ_COR_1026 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶12 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header with options.

Requirement: The implementation uses both the Pad1 and PadN Option Type values for the Hop-by-Hop Options header and Destination Options header.

RFC text: `{{The same Option Type numbering space is used for both the Hop-by-Hop Options header and the Destination Options header. However, the specification of a particular option may restrict its use to only one of those two headers}}`.

RQ_COR_1027 Extension Header Options [Generate]

RFC 2460 *Clause:* 4.2 ¶13 *Type:* MUST *applies to:* Node

Context: The implementation transmits a packet with a Hop-by-Hop and/or Destination Options extension header where individual options have multi-octet values within Option Data fields.

Requirement: The implementation ensures that the multi-octet values within the Option Data files fall on natural boundaries. The Option Type is at an integer multiple of x octets from the start of the header plus y octets.

RFC text: Individual options may have specific alignment requirements, to ensure that multi-octet values within Option Data fields fall on natural boundaries. `{{The alignment requirement of an option is specified using the notation xn+y, meaning the Option Type must appear at an integer multiple of x octets from the start of the header, plus y octets.}}`.

RQ_COR_1028 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶15 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header. Pad1 is used to align subsequent options and to pad out the containing header to a multiple of 8 octets in length.

Requirement: Implementation recognizes the Pad1 option.

RFC text: There are two padding options which are used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length. `{{These padding options must be recognized by all IPv6 implementations}}`.

RQ_COR_1029 Extension Header Options [Generate]

RFC 2460 *Clause:* 4.2 ¶18 *Type:* MUST *applies to:* Node

Context: The implementation transmits a Hop-by-Hop or Destination Options extension header. The implementation needs to insert one octet of padding into the Options area of a header in order to align subsequent options and to pad out the containing header to a multiple of 8 octets in length.

Requirement: The implementation uses the Pad1 option for padding the Options area when only one octet of padding is necessary.

RFC text: `{{The Pad1 option is used to insert one octet of padding into the Options area of a header}}`.

RQ_COR_1030 Extension Header Options [Generate]

RFC 2460 *Clause:* 4.2 ¶18 *Type:* SHOULD *applies to:* Node

Context: The implementation transmits a Hop-by-Hop or Destination Options extension header. The implementation needs to insert more than one octet of padding into the Options area of a header in order to align subsequent options and to pad out the containing header to a multiple of 8 octets in length.

Requirement: Implementation uses PadN for padding the Options area when more than one octet of padding is necessary.

RFC text: `{{If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options}}`. `{{The PadN option is used to insert two or more octets of padding into the Options area of a header}}`.

RQ_COR_1031 Extension Header Options [Generate]

RFC 2460 *Clause:* 4.2 ¶18 *Type:* SHOULD *applies to:* Node

Context: The implementation transmits a packet with Hop-by-Hop or Destination Options extension header. The implementation needs to insert more than one octet of padding into the Options area of a header in order to align subsequent options and to pad out the containing header to a multiple of 8 octets in length.

Requirement: Implementation does not use multiple Pad1 options for padding the Options area when more than one octet of padding is necessary.

RFC text: `{{If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options}}`. `{{The PadN option is used to insert two or more octets of padding into the Options area of a header}}`.

RQ_COR_1032 Hop by Hop Header [Generate]

RFC 2460 *Clause:* 4.3 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Hop-by-Hop Options header.

Requirement: The value of the Next Header Field in the Hop-by-Hop Options header is the type of the header immediately following the Hop-by-Hop Options header.

RFC text: Next Header 8-bit selector. `{{Identifies the type of header inmmediately following the Hop-by-Hop Options header}}`. Uses the same values as the IPv4 Protocol field [RFC-1700 et seq.].

RQ_COR_1033 Hop by Hop Header [Generate]

RFC 2460 *Clause:* 4.3 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Hop-by-Hop Options header.

Requirement: The value of the Hdr Ext Len Field in the Hop-by-Hop Options header is its length in 8-octet units, not including the first 8 octets.

RFC text: Hdr Ext Len 8-bit unsigned integer. `{{Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets}}`.

RQ_COR_1034 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with one or more routers to be "visited" on the way to the packet's destination.

Requirement: The implementation lists in the Routing header the one or more nodes[routers] to be "visited" on the way to the packet's destination.

RFC text: `{{The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination}}`.

RQ_COR_1035 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Routing header.

Requirement: The Routing extension header presence is set to the value 43 in the header immediately preceding the Routing extension header.

RFC text: `{{The Routing header is identified by a Next Header value of 43 in the immediately preceding header}}`.

RQ_COR_1036 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Routing header.

Requirement: The value of the Next Header Field in the Routing header is the type of header immediately following the Routing header.

RFC text: Next Header 8-bit selector. `{{Identifies the type of header immediately following the Routing header}}`. Uses the same values as the IPv4 Protocol field [RFC-1700 et seq.].

RQ_COR_1037 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Routing header.

Requirement: The value of the Hdr Ext Len Field in the Routing header is its length in 8-octet units, not including the first 8 octets.

RFC text: Hdr Ext Len 8-bit unsigned integer. `{{Length of the Routing header in 8-octet units, not including the first 8 octets}}`.

RQ_COR_1038 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Routing header.

Requirement: The value of the Segments Left Field in the Routing header is the number of explicitly listed intermediate nodes still to be visited before reaching the final destination.

RFC text: Segments Left 8-bit unsigned integer. `{{ Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination}}`.

RQ_COR_1039 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Routing header.

Requirement: The value of the Segments Left Field in the Routing header is the number of explicitly listed intermediate nodes still to be visited before reaching the final destination.

RFC text: Segments Left 8-bit unsigned integer. `{{ Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination}}`.

RQ_COR_1040 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶7-8 *Type:* MUST *applies to:* Node

Context: The implementation processes a Routing header with an unrecognizable Routing Type value. The Segments Left Field value is set to zero.

Requirement: Implementation ignores the Routing header except its Next Header field and processes the next header in the packet, whose type is identified by the Next Header field in the Routing header.

RFC text: `{{If, while processing a received packet, a node encounters a Routing header with an unrecognized Routing Type value, the required behavior of the node depends on the value of the Segments Left field}}`. `{{If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing header}}`.

RQ_COR_1041 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶7,9 *Type:* MUST *applies to:* Router

Context: The implementation processes a Routing header and with an unrecognizable Routing Type value. The Segments Left Field value is set to a non-zero value.

Requirement: The implementation discards the packet and sends an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

RFC text: {{If, while processing a received packet, a node encounters a Routing header with an unrecognized Routing Type value, the required behavior of the node depends on the value of the Segments Left field}}. {{If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type}}.

RQ_COR_1042 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶10 *Type:* MUST *applies to:* Router

Context: The implementation processes a Routing header. The packet is to be forwarded onto a link whose link MTU is less than the size of the packet.

Requirement: The implementation discards the packet and sends an ICMP Packet Too Big message to the packet's Source Address.

RFC text: {{If, after processing a Routing header of a received packet, an intermediate node determines that the packet is to be forwarded onto a link whose link MTU is less than the size of the packet, the node must discard the packet and send an ICMP Packet Too Big message to the packet's Source Address}}.

RQ_COR_1043 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶13-14,16 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Type 0 Routing header.

Requirement: The value of the Hdr Ext Len Field in the Type 0 Routing header is equal to two times the number of addresses in the Routing header. The value of the Routing Type Field is set to zero. The value of the Reserved Field in the Type 0 Routing header is initialized to zero

RFC text: Hdr Ext Len 8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets. {{For the Type 0 Routing header, Hdr Ext Len is equal to two times the number of addresses in the header.}}.

RQ_COR_1046 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶16 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with a Type 0 Routing header.

Requirement: The implementation ignores the value of the Reserved Field in the Type 0 Routing header.

RFC text: Reserved 32-bit reserved field. Initialized to zero for transmission; {{ignored on reception.}}.

RQ_COR_1047 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶18 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Type 0 Routing header.

Requirement: The Address[n] Fields in the Type 0 does not contain Multicast addresses.

RFC text: {{Multicast addresses must not appear in a Routing header of Type 0}}.

RQ_COR_1048 Routing Header [Generate]

RFC 2460 *Clause:* 4.4 ¶18 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Type 0 Routing header.

Requirement: The IPv6 Destination Address field of a packet carrying a Routing header of Type 0 is not a Multicast address.

RFC text: Multicast addresses must not appear in a Routing header of Type 0, `{{or in the IPv6 Destination Address field of a packet carrying a Routing header of Type 0}}`.

RQ_COR_1049 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with a Routing header. The Destination Address in the IPv6 Header is not the implementation's address.

Requirement: The implementation does not examine nor process the Routing Header.

RFC text: `{{A Routing header is not examined or processed until it reaches the node identified in the Destination Address field of the IPv6 header}}`.

RQ_COR_1050 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with a Routing header. The Destination Address in the IPv6 Header is the implementation's address.

Requirement: The implementation examines and processes the Routing Header.

RFC text: `{{A Routing header is not examined or processed until it reaches the node identified in the Destination Address field of the IPv6 header}}`.

RQ_COR_1051 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19-20 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Type 0 Routing header. The Segments Left Field value is set to zero.

Requirement: The implementation ignores the Routing header except its Next Header field and processes the next header in the packet, whose type is identified by the Next Header field in the Routing header.

RFC text: ...In that node, dispatching on the Next Header field of the immediately preceding header causes the Routing header module to be invoked, which, `{{in the case of Routing Type 0}}`, performs the following algorithm: `{{...if Segments Left = 0 {proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing header.}}`

RQ_COR_1052 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19,21 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Type 0 Routing header. The Segments Left Field value is larger than zero. The Hdr Ext Len is odd.

Requirement: The implementation sends an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Hdr Ext Len field, and discards the packet.

RFC text: ...In that node, dispatching on the Next Header field of the immediately preceding header causes the Routing header module to be invoked, which, `{{in the case of Routing Type 0}}`, performs the following algorithm: `{{...else if Hdr Ext Len is odd {send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Hdr Ext Len field, and discard the packet}}`.

RQ_COR_1053 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19,22-23 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Type 0 Routing header. The Hdr Ext Len value is even. The Segments Left field value is larger than zero and greater than the number of addresses in the Routing header.

Requirement: The implementation sends an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field, and discards the packet.

RFC text: ...In that node, dispatching on the Next Header field of the immediately preceding header causes the Routing header module to be invoked, which, `{{in the case of Routing Type 0}}`, performs the following algorithm: .....else {compute n, the number of addresses in the Routing header, by dividing Hdr Ext Len by 2. `{{...if Segments Left is greater than n {send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field, and discard the packet}}`}.

RQ_COR_1055 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19,22-25 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Type 0 Routing header. The Hdr Ext Len is even. The Segments Left Field value is larger than zero and not greater than the number of Addresses in the Routing header address vector. The next address to be visited in the address vector is a multicast address.

Requirement: The implementation discards the packet.

RFC text: ...In that node, dispatching on the Next Header field of the immediately preceding header causes the Routing header module to be invoked, which, `{{in the case of Routing Type 0}}`, performs the following algorithm: ...else {compute n, the number of addresses in the Routing header, by dividing Hdr Ext Len by 2... ...if Segments Left is greater than n {send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field, and discard the packet}... ...else {decrement Segments Left by 1; compute i, the index of the next address to be visited in the address vector, by subtracting Segments Left from n... `{{...if Address [i] or the IPv6 Destination Address is multicast {discard the packet...}}`.

RQ_COR_1056 Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19,22-25 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Type 0 Routing header. The Hdr Ext Len is even. The Segments Left Field value is larger than zero and is not greater than than the number of Addresses in the Routing header address vector. The Destination Address is a multicast address.

Requirement: The implementation discards the packet.

RFC text: ...In that node, dispatching on the Next Header field of the immediately preceding header causes the Routing header module to be invoked, which, `{{in the case of Routing Type 0}}`, performs the following algorithm:... ...else {compute n, the number of addresses in the Routing header, by dividing Hdr Ext Len by 2... ...if Segments Left is greater than n {send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field, and discard the packet}... ...else {decrement Segments Left by 1; compute i, the index of the next address to be visited in the address vector, by subtracting Segments Left from n... `{{...if Address [i] or the IPv6 Destination Address is multicast discard the packet...}}`.

RQ_COR_1058 Hop Limit [Process] Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19,22-27 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Type 0 Routing header. The Hdr Ext Len is even. The Segments Left Field value is larger than zero and not greater than the number of addresses in the Routing header. Neither the next address to be visited nor the IPv6 Destination Address are multicast addresses. The Hop Limit is less than or equal to 1.

Requirement: The implementation sends an ICMP Time Exceeded -- Hop Limit Exceeded in Transit message to the Source Address and discards the packet.

RFC text: ...In that node, dispatching on the Next Header field of the immediately preceding header causes the Routing header module to be invoked, which, `{{in the case of Routing Type 0}}`, performs the following algorithm:... ...else {compute n, the number of addresses in the Routing header, by dividing Hdr Ext Len by 2... if Segments Left is greater than n {send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field, and discard the packet}... ...else {decrement Segments Left by 1; compute i, the index of the next address to be visited in the address vector, by subtracting Segments Left from n. ...if Address [i] or the IPv6 Destination Address is multicast {discard the packet... ...else {swap the IPv6 Destination Address and Address[i]... `{{...if the IPv6 Hop Limit is less than or equal to 1 {send an ICMP Time Exceeded -- Hop Limit Exceeded in Transit message to the Source Address and discard the packet}}}`.

RQ_COR_1059 Hop Limit [Process] Routing Header [Process]

RFC 2460 *Clause:* 4.4 ¶19,22-29 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Type 0 Routing header. The Hdr Ext Len is even. The Segments Left Field value is larger than zero and not greater than the number of addresses in the Routing header. Neither the next address to be visited nor the IPv6 Destination Address are multicast addresses. The Hop Limit is larger than 1.

Requirement: The implementation decrements Segments Left Field by 1, swaps the IPv6 Destination Address and next address to be visited, decrements the IPv6 Hop Limit by 1, and forwards the packet to the new destination.

RFC text: In that node, dispatching on the Next Header field of the immediately preceding header causes the Routing header module to be invoked, which, `{{in the case of Routing Type 0}}`, performs the following algorithm:... ...else {compute n, the number of addresses in the Routing header, by dividing Hdr Ext Len by 2... ...if Segments Left is greater than n {send an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field, and discard the packet}... ...else {decrement Segments Left by 1; compute i, the index of the next address to be visited in the address vector, by subtracting Segments Left from n. ...if Address [i] or the IPv6 Destination Address is multicast {discard the packet... ...else {swap the IPv6 Destination Address and Address[i]... `{{...if the IPv6 Hop Limit is less than or equal to 1 {send an ICMP Time Exceeded -- Hop Limit Exceeded in Transit message to the Source Address and discard the packet}... else {decrement the Hop Limit by 1... resubmit the packet to the IPv6 module for transmission to the new destination}}`.

RQ_COR_1064 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶1, 5 ¶5 *Type:* MAY *applies to:* Node

Context: The implementation generates a packet larger than the path MTU to the packet's destination.

Requirement: The implementation fragments the packet and uses the Fragment header.

RFC text: `{{The Fragment header is used by an IPv6 source to send a packet larger than would fit in the path MTU to its destination. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path -- see section 5.)}}`. `{{In order to send a packet larger than a path's MTU, a node may use the IPv6 Fragment header to fragment the packet at the source and have it reassembled at the destination(s)}}`. However, the use of such fragmentation is discouraged in any application that is able to adjust its packets to fit the measured path MTU (i.e., down to 1280 octets).

RQ_COR_1065 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶1, 5 ¶5 *Type:* SHOULD *applies to:* Node

Context: An application has the choice of fragmenting a packet larger than the path MTU or adjusting the packet size to fit the path MTU.

Requirement: The implementation adjusts the packet size to fit the path MTU.

RFC text: `{{The Fragment header is used by an IPv6 source to send a packet larger than would fit in the path MTU to its destination. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path -- see section 5.)}}`. In order to send a packet larger than a path's MTU, a node may use the IPv6 Fragment header to fragment the packet at the source and have it reassembled at the destination(s). `{{However, the use of such fragmentation is discouraged in any application that is able to adjust its packets to fit the measured path MTU (i.e., down to 1280 octets)}}`.

RQ_COR_1066 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶1, 5 ¶5 *Type:* MUST *applies to:* Router

Context: The implementation processes a packet larger than would fit in the path MTU to its destination.

Requirement: The implementation does not use the Fragment header to fragment the packet.

RFC text: {{The Fragment header is used by an IPv6 source to send a packet larger than would fit in the path MTU to its destination. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path -- see section 5.)}}.
NOTE:see RQ_COR_1042

RQ_COR_1067 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Fragment header.

Requirement: The Fragment header presence is set to the value 44 in the Next Header field immediately preceding the Fragment header.

RFC text: {{The Fragment header is identified by a Next Header value of 44 in the immediately preceding header}}.

RQ_COR_1068 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶2-3,5 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Fragment header.

Requirement: The value of the Next Header Field in the Fragment header is the initial header type of the Fragmentable Part of the original packet. The value of the Reserved Field in the Fragment header is initialized to zero. The value of the Res Field in the Fragment header is initialized to zero.

RFC text: Next Header 8-bit selector. {{Identifies the initial header type of the Fragmentable Part of the original packet (defined below)}}. Uses the same values as the IPv4 Protocol field [RFC-1700 et seq.]. ...Reserved 8-bit reserved field. {{Initialized to zero for transmission}}; ignored on reception. Res 2-bit reserved field. {{Initialized to zero for transmission}}; ignored on reception.

RQ_COR_1069 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶3,5 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet with a Fragment header.

Requirement: The implementation ignores the value of the Reserved Field in the Fragment Header.

RFC text: Reserved 8-bit reserved field. Initialized to zero for transmission;{{ignored on reception}}. Res 2-bit reserved field. Initialized to zero for transmission;{{ignored on reception}}.

RQ_COR_1070 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Fragment header.

Requirement: The value of the Fragment Offset Field in the Fragment header is the offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part of the original packet.

RFC text: Fragment Offset 13-bit unsigned integer. {{The offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part of the original packet}}.

RQ_COR_1071 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation generates a fragment other than the "rightmost" fragment of a fragmented packet.

Requirement: The Fragment header of the fragment contains an M Flag value set to 1 indicating that there are more fragments following this fragment.

RFC text: M flag`{{1 = more fragments}}`; 0 = last fragment.

RQ_COR_1072 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation generates the "rightmost" fragment of a fragmented packet.

Requirement: The Fragment header of the fragment contains an M Flag value set to 0 indicating that this fragment is the last.

RFC text: M flag 1 = more fragments; `{{0 = last fragment}}`.

RQ_COR_1073 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶9-10 *Type:* MUST *applies to:* Node

Context: The implementation generates fragmented packets.

Requirement: The implementation generates a unique Identification value for packet to be fragmented different than that for any other fragmented packet sent recently with the same Source Address and Destination Address.

RFC text: `{{The Identification must be different than that of any other fragmented packet sent recently* with the same Source Address and Destination Address}}`. ...* "recently" means within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet. However, it is not required that a source node know the maximum packet lifetime. Rather, it is assumed that the requirement can be met by maintaining the Identification value as a simple, 32-bit, "wrap-around" counter, incremented each time a packet must be fragmented. It is an implementation choice whether to maintain a single counter for the node or multiple counters, e.g., one for each of the node's possible source addresses, or one for each active (source address, destination address) combination.

RQ_COR_1074 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶9-10 *Type:* MUST *applies to:* Node

Context: The implementation generates fragmented packets. Along this, the implementation generates an Identification value for every packet that is to be fragmented.

Requirement: The implementation generates each time different Identification value than that of any other fragmented packet sent recently* with the same Source Address and Destination Address. Identification value is obtined from a MULTIPLE (one for each of the node's possible source addresses) simple, 32-bit, "wrap-around" counters, incremented each time a packet must be fragmented.

RFC text: `{{The Identification must be different than that of any other fragmented packet sent recently* with the same Source Address and Destination Address}}`. ...* "recently" means within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet. However, it is not required that a source node know the maximum packet lifetime. Rather, it is assumed that the requirement can be met by maintaining the Identification value as a simple, 32-bit, "wrap-around" counter, incremented each time a packet must be fragmented. It is an implementation choice whether to maintain a single counter for the node or multiple counters, e.g., one for each of the node's possible source addresses, or one for each active (source address, destination address) combination.

RQ_COR_1075 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶9-10 *Type:* MUST *applies to:* Node

Context: The implementation generates fragmented packets. Along this, the implementation generates an Identification value for every packet that is to be fragmented.

Requirement: The implementation generates each time different Identification value than that of any other fragmented packet sent recently* with the same Source Address and Destination Address. Identification value is obtined from a MULTIPLE (one for each active (source address, destination address) combination) simple, 32-bit, "wrap-around" counters, incremented each time a packet must be fragmented.

RFC text: `{{The Identification must be different than that of any other fragmented packet sent recently* with the same Source Address and Destination Address}}`. ...* "recently" means within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet. However, it is not required that a source node know the maximum packet lifetime. Rather, it is assumed that the requirement can be met by maintaining the Identification value as a simple, 32-bit, "wrap-around" counter, incremented each time a packet must be fragmented. It is an implementation choice whether to maintain a single counter for the node or multiple counters, e.g., one for each of the node's possible source addresses, or one for each active (source address, destination address) combination.

RQ_COR_1076 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶9-10 *Type:* MUST *applies to:* Node

Context: The implementation generates fragmented packets. Along this, the implementation generates an Identification value for every packet that is to be fragmented. A Routing header is present.

Requirement: The Destination Address of the fragemented packets is the final destination.

RFC text: The Identification must be different than that of any other fragmented packet sent recently* with the same Source Address and Destination Address`{{. If a Routing header is present, the Destination Address of concern is that of the final destination}}`.

RQ_COR_1079 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶13-14 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet too long for the path MTU to its destination.

Requirement: The implementation does not fragment the IPv6 header plus any extension headers that must be processed by nodes en route to the destination. The implementation fragments any extension headers that need to be processed only by the final destination node(s), plus the upper-layer header and data.

RFC text: `{{The Unfragmentable Part consists of the IPv6 header plus any extension headers that must be processed by nodes en route to the destination, that is, all headers up to and including the Routing header if present, else the Hop-by-Hop Options header if present, else no extension headers}}`. `{{The Fragmentable Part consists of the rest of the packet, that is, any extension headers that need be processed only by the final destination node(s), plus the upper-layer header and data}}`.

RQ_COR_1080 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet too long for the path MTU to its destination.

Requirement: The implementation divides the Fragmentable Part of original packet into fragments. Each fragment, except possibly the last ("rightmost") one, is an integer multiple of 8 octets long. The lengths of the fragments are such that the resulting fragment packets fit within the path MTU to the packets' destination(s).

RFC text: `{{The Fragmentable Part of the original packet is divided into fragments, each, except possibly the last ("rightmost") one, being an integer multiple of 8 octets long}}. {{The lengths of the fragments must be chosen such that the resulting fragment packets fit within the MTU of the path to the packets' destination(s)}}.`

RQ_COR_1081 Fragment Packets [Generate]

RFC 2460 *Clause:* 4.5 ¶15,18-25 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet too long for the path MTU to its destination.

Requirement: The implementation divides the Fragmentable Part of original packet into fragments. The implementation transmits these fragments in separate "fragment packets". Each fragment packet is composed of: (1) The Unfragmentable Part of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the IPv6 header itself), and the Next Header field of the last header of the Unfragmentable Part changed to 44. (2) A Fragment header containing: a. The Next Header value that identifies the first header of the Fragmentable Part of the original packet. b. A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable Part of the original packet. The Fragment Offset of the first ("leftmost") fragment is 0. c. An M flag value of 0 if the fragment is the last ("rightmost") one, else an M flag value of 1. d. The Identification value generated for the original packet. (3) The fragment itself.

RFC text: `{{The fragments are transmitted in separate "fragment packets"...}}. {{Each fragment packet is composed of: (1) The Unfragmentable Part of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the IPv6 header itself), and the Next Header field of the last header of the Unfragmentable Part changed to 44. (2) A Fragment header containing: The Next Header value that identifies the first header of the Fragmentable Part of the original packet. A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable Part of the original packet. The Fragment Offset of the first ("leftmost") fragment is 0. An M flag value of 0 if the fragment is the last ("rightmost") one, else an M flag value of 1. The Identification value generated for the original packet. (3) The fragment itself}}.`

RQ_COR_1082 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶27-38 *Type:* MUST *applies to:* Node

Context: The implementation receives "Fragment packets".

Requirement: The implementation reassembles into their original, unfragmented form, the received fragments following these rules: An original packet is reassembled only from fragment packets that have the same Source Address, Destination Address, and Fragment Identification. The Unfragmentable Part of the reassembled packet consists of all headers up to, but not including, the Fragment header of the first fragment packet (that is, the packet whose Fragment Offset is zero), with the following two changes: a. The Next Header field of the last header of the Unfragmentable Part is obtained from the Next Header field of the first fragment's Fragment header. b. The Payload Length of the reassembled packet is computed from the length of the Unfragmentable Part and the length and offset of the last fragment. The Fragmentable Part of the reassembled packet is constructed from the fragments following the Fragment headers in each of the fragment packets. The length of each fragment is computed by subtracting from the packet's Payload Length the length of the headers between the IPv6 header and fragment itself; its relative position in Fragmentable Part is computed from its Fragment Offset value. Finnally, the Fragment header is not present in the final, reassembled packet.

RFC text: `{{The following rules govern reassembly: An original packet is reassembled only from fragment packets that have the same Source Address, Destination Address, and Fragment Identification. The Unfragmentable Part of the reassembled packet consists of all headers up to, but not including, the Fragment header of the first fragment packet (that is, the packet whose Fragment Offset is zero), with the following two changes: The Next Header field of the last header of the Unfragmentable Part is obtained from the Next Header field of the first fragment's Fragment header. The Payload Length of the reassembled packet is computed from the length of the Unfragmentable Part and the length and offset of the last fragment. For example, a formula for computing the Payload Length of the reassembled original packet is: PL.orig = PL.first - FL.first - 8 + (8 * FO.last) + FL.last}}`. `{{The Fragmentable Part of the reassembled packet is constructed from the fragments following the Fragment headers in each of the fragment packets. The length of each fragment is computed by subtracting from the packet's Payload Length the length of the headers between the IPv6 header and fragment itself; its relative position in Fragmentable Part is computed from its Fragment Offset value. The Fragment header is not present in the final, reassembled packet}}`.

RQ_COR_1083 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶39-40 *Type:* MUST *applies to:* Node

Context: The implementation receives "Fragment packets". Insufficient fragments are received to complete reassembly of a packet within 60 seconds of the reception of the first-arriving fragment of that packet.

Requirement: The implementation abandons the reassembly of that packet and all the fragments that have been received for that packet are discarded.

RFC text: The following error conditions may arise when reassembling fragmented packets... `{{...If insufficient fragments are received to complete reassembly of a packet within 60 seconds of the reception of the first-arriving fragment of that packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded}}`.

RQ_COR_1084 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶39-40 *Type:* SHOULD *applies to:* Node

Context: The implementation receives "Fragment packets". Insufficient fragments are received to complete reassembly of a packet within 60 seconds of the reception of the first-arriving fragment of that packet. The first fragment packet(i.e., the one with a Fragment Offset of zero) is in the set of received fragments. The implementation abandons the reassembly of that packet and all the fragments that have been received for that packet are discarded.

Requirement: The implementation sends an ICMP Time Exceeded -- Fragment Reassembly Time Exceeded message to the source of that fragment.

RFC text: The following error conditions may arise when reassembling fragmented packets... `{{...If insufficient fragments are received to complete reassembly of a packet within 60 seconds of the reception of the first-arriving fragment of that packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded. If the first fragment (i.e., the one with a Fragment Offset of zero) has been received, an ICMP Time Exceeded -- Fragment Reassembly Time Exceeded message should be sent to the source of that fragment}}`.

RQ_COR_1085 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶39,41 *Type:* MUST *applies to:* Node

Context: The implementation receives "Fragment packets". The length of a fragment, as derived from the fragment packet's Payload Length field, is not a multiple of 8 octets and the M flag of the same fragment is 1.

Requirement: The implementation discards the fragment whose Payload Length field is not a multiple of 8 octets and whose M flag is set to 1.

RFC text: The following error conditions may arise when reassembling fragmented packets:... `{{...If the length of a fragment, as derived from the fragment packet's Payload Length field, is not a multiple of 8 octets and the M flag of that fragment is 1, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Payload Length field of the fragment packet}}`.

RQ_COR_1086 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶39,42 *Type:* MUST *applies to:* Node

Context: The implementation receives "Fragment packets". The length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets.

Requirement: The implementation discards that fragment.

RFC text: The following error conditions may arise when reassembling fragmented packets... `{{...If the length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Fragment Offset field of the fragment packet}}`.

RQ_COR_1087 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶43-44 *Type:* MUST *applies to:* Node

Context: The implementation receives "Fragment packets". The number and content of the headers preceding the Fragment header of different fragments of the same original packet differ.

Requirement: The implementation processes the headers present that precede the Fragment header in each fragment packet prior to queueing the fragments for reassembly. Only the headers in the Offset zero fragment packet are retained in the reassembled packet.

RFC text: The following conditions are not expected to occur, but are not considered errors if they do. `{{The number and content of the headers preceding the Fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the Fragment header in each fragment packet, are processed when the packets arrive, prior to queueing the fragments for reassembly. Only those headers in the Offset zero fragment packet are retained in the reassembled packet}}`.

RQ_COR_1088 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶43,45 *Type:* MUST *applies to:* Node

Context: The implementation receives "Fragment packets". The Next Header values in the Fragment headers of different fragments of the same original packet differ.

Requirement: The implementation uses only the Next Header value from the Offset zero fragment packet for reassembly.

RFC text: The following conditions are not expected to occur, but are not considered errors if they do. `{{The Next Header values in the Fragment headers of different fragments of the same original packet may differ. Only the value from the Offset zero fragment packet is used for reassembly}}`.

RQ_COR_1089 Destination Options Header [Generate]

RFC 2460 *Clause:* 4.6 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Destination Options header.

Requirement: The Destination Options extension header presence is set to the value 60 in the Next Header field in the header immediately preceding the Destination Options header.

RFC text: `{{The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header}}`.

RQ_COR_1090 Destination Options Header [Generate]

RFC 2460 *Clause:* 4.6 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Destination Options header.

Requirement: The value of the Next Header Field in the Destination Options header is the type of header immediately following the Destination Options header.

RFC text: Next Header 8-bit selector. `{{Identifies the type of header immediately following the Destination Options header}}`. Uses the same values as the IPv4 Protocol field [RFC-1700 et seq.].

RQ_COR_1091 Destination Options Header [Generate]

RFC 2460 *Clause:* 4.6 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with a Destination Options header.

Requirement: The value of the Hdr Ext Len Field in the Destination Options header is its length in 8-octet units, not including the first 8 octets.

RFC text: Hdr Ext Len 8-bit unsigned integer. `{{Length of the Destination Options header in 8-octet units, not including the first 8 octets}}`.

RQ_COR_1092 Extension Headers [Generate]

RFC 2460 *Clause:* 4.7 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet with an IPv6 header or any extension header indicating that there is nothing following that header.

Requirement: The implementation sets the value 59 in the last header of the IPv6 packet to indicate that nothing follows that header.

RFC text: `{{The value 59 in the Next Header field of an IPv6 header or any extension header indicates that there is nothing following that header}}`.

RQ_COR_1093 Extension Headers [Process]

RFC 2460 *Clause:* 4.7 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet for forwarding with an IPv6 header or any extension header containing a value 59 in the Next Header field. The Payload Length field of the IPv6 header indicates the presence of octets past the end of a header whose Next Header field contains 59.

Requirement: The implementation ignores and passes on unchanged the octets past the end of a header whose Next Header field contains 59.

RFC text: The value 59 in the Next Header field of an IPv6 header or any extension header indicates that there is nothing following that header. `{{If the Payload Length field of the IPv6 header indicates the presence of octets past the end of a header whose Next Header field contains 59, those octets must be ignored, and passed on unchanged if the packet is forwarded}}`.

RQ_COR_1094 IPv6 Packet [Generate]

RFC 2460 *Clause:* 5 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is on a link that cannot transmit a 1280-octet packet in one piece.

Requirement: The implementation provides link-specific fragmentation and reassembly for packets of size 1280 octets or more at a layer below IPv6.

RFC text: IPv6 requires that every link in the internet have an MTU of 1280 octets or greater. `{{On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6}}`.

RQ_COR_1095 MTU [Determine Default]

RFC 2460 *Clause:* 5 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is on a link with a configurable MTU.

Requirement: The configured MTU is 1280 octets or more.

RFC text: `{{Links that have a configurable MTU (for example, PPP links [RFC-1661]) must be configured to have an MTU of at least 1280 octets}}`.

RQ_COR_1096 MTU [Determine Default]

RFC 2460 *Clause:* 5 ¶2 *Type:* RECOMMENDED *applies to:* Node

Context: The implementation is on a link with a configurable MTU.

Requirement: The implementation provides with a configured MTU of 1500 octets or more.

RFC text: `{{...it is recommended that they be configured with an MTU of 1500 octets or greater, to accommodate possible encapsulations (i.e., tunneling) without incurring IPv6-layer fragmentation}}`.

RQ_COR_1097 IPv6 Packet [Process]

RFC 2460 *Clause:* 5 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation is directly attached to one or more links.

Requirement: From each link to which the implementation is attached, the implementation accepts packets as large as that link's MTU.

RFC text: `{{From each link to which a node is directly attached, the node must be able to accept packets as large as that link's MTU}}`.

RQ_COR_1098 PMTU Discovery

RFC 2460 *Clause:* 5 ¶4 *Type:* Strongly *applies to:* Node

Context: The implementation is attached to one or more links.

Requirement: The implementation implements Path MTU Discovery.

RFC text: `{{It is strongly recommended that IPv6 nodes implement Path MTU Discovery [RFC-1981], in order to discover and take advantage of path MTUs greater than 1280 octets}}`.

RQ_COR_1099 IPv6 Packet [Generate]

RFC 2460 *Clause:* 5 ¶4 *Type:* MAY *applies to:* Node

Context: A minimal IPv6 implementation is attached to one or more links.

Requirement: The implementation sends packets no larger than 1280 octets.

RFC text: `{{However, a minimal IPv6 implementation (e.g., in a boot ROM) may simply restrict itself to sending packets no larger than 1280 octets, and omit implementation of Path MTU Discovery}}`.

RQ_COR_1100 Fragment Packets [Process]

RFC 2460 *Clause:* 5 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation receives fragmented packets. The original fragment size is 1500 octets.

Requirement: The implementation reassembles the fragments into the original unfragmented packet of 1500 octets.

RFC text: `{{A node must be able to accept a fragmented packet that, after reassembly, is as large as 1500 octets}}`.

RQ_COR_1101 Fragment Packets [Process]

RFC 2460 *Clause:* 5 ¶6 *Type:* MAY *applies to:* Node

Context: The implementation receives fragmented packets. The original fragment size is larger than 1500 octets.

Requirement: The implementation reassembles the fragments into the original unfragmented packet of size larger than 1500 octets.

RFC text: `{{A node is permitted to accept fragmented packets that reassemble to more than 1500 octets}}`.

RQ_COR_1102 Fragment Packets [Generate]

RFC 2460 *Clause:* 5 ¶6 *Type:* SHOULD *applies to:* Node

Context: The implementation provides IPv6 fragmentation service to an upper-layer protocol or application in order to send packets larger than the path MTU. The upper-layer protocol or application can not assure that the destination is capable of reassembling packets larger than 1500 octets.

Requirement: The implementation does not send packets larger than 1500 octets.

RFC text: `{{An upper-layer protocol or application that depends on IPv6 fragmentation to send packets larger than the MTU of a path should not send packets larger than 1500 octets unless it has assurance that the destination is capable of reassembling packets of that larger size}}`.

RQ_COR_1103 Fragment Packets [Generate]

RFC 2460 *Clause:* 5 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation sends an IPv6 packet to an IPv4 destination. The implementation receives an ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280 octets.

Requirement: The implementation includes a Fragment header in following packets with a suitable Identification value to use in resulting IPv4 fragments.

RFC text: In response to an IPv6 packet that is sent to an IPv4 destination (i.e., a packet that undergoes translation from IPv6 to IPv4), the originating IPv6 node may receive an ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280. In that case, `{{the IPv6 node is not required to reduce the size of subsequent packets to less than 1280, but must include a Fragment header in those packets so that the IPv6-to-IPv4 translating router can obtain a suitable Identification value to use in resulting IPv4 fragments}}`. Note that this means the payload may have to be reduced to 1232 octets (1280 minus 40 for the IPv6 header and 8 for the Fragment header), and smaller still if additional extension headers are used.

RQ_COR_1104 Fragment Packets [Generate]

RFC 2460 *Clause:* 5 ¶7 *Type:* MAY *applies to:* Node

Context: The implementation sends an IPv6 packet to an IPv4 destination. The implementation receives an ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280 octets.

Requirement: The implementation does not reduce the size of subsequent packets to less than 1280 octets.

RFC text: In response to an IPv6 packet that is sent to an IPv4 destination (i.e., a packet that undergoes translation from IPv6 to IPv4), the originating IPv6 node may receive an ICMP Packet Too Big message reporting a Next-Hop MTU less than 1 280. In that case, `{{the IPv6 node is not required to reduce the size of subsequent packets to less than 1 280, but must include a Fragment header in those packets so that the IPv6-to-IPv4 translating router can obtain a suitable Identification value to use in resulting IPv4 fragments}}`. Note that this means the payload may have to be reduced to 1 232 octets (1 280 minus 40 for the IPv6 header and 8 for the Fragment header), and smaller still if additional extension headers are used.

RQ_COR_1105 Fragment Packets [Generate]

RFC 2460 *Clause:* 5 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation sends an IPv6 packet to an IPv4 destination. The implementation receive an ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280 octets. The implementation reduces the size of subsequent packets to less than 1 280 octets.

Requirement: The implementation reduces the payload to at least 1 232 octets (1 280 minus 40 for the IPv6 header and 8 for the Fragment header) and smaller still if additional extension headers are used.

RFC text: In response to an IPv6 packet that is sent to an IPv4 destination (i.e., a packet that undergoes translation from IPv6 to IPv4), the originating IPv6 node may receive an ICMP Packet Too Big message reporting a Next-Hop MTU less than 1 280. In that case, the IPv6 node is not required to reduce the size of subsequent packets to less than 1 280, but must include a Fragment header in those packets so that the IPv6-to-IPv4 translating router can obtain a suitable Identification value to use in resulting IPv4 fragments. `{{Note that this means the payload may have to be reduced to 1 232 octets (1 280 minus 40 for the IPv6 header and 8 for the Fragment header), and smaller still if additional extension headers are used}}`.

RQ_COR_1106 Flow Label [Generate]

RFC 2460 *Clause:* 6 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation generates IPv6 packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service.

Requirement: The implementation, acting as a source, uses Flow Label field in the IPv6 header to label sequences of packets for which it requests special handling by the IPv6 routers.

RFC text: `{{The 20-bit Flow Label field in the IPv6 header may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service. This aspect of IPv6 is, at the time of writing, still experimental and subject to change as the requirements for flow support in the Internet become clearer}}`.

RQ_COR_1107 Flow Label [Generate]

RFC 2460 *Clause:* 6 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates IPv6 packets. The implementation does not support the functions of the Flow Label field.

Requirement: The implementation sets the field to zero.

RFC text: `{{Hosts or routers that do not support the functions of the Flow Label field are required to set the field to zero when originating a packet, pass the field on unchanged when forwarding a packet, and ignore the field when receiving a packet}}`.

RQ_COR_1108 Flow Label [Process]

RFC 2460 *Clause:* 6 ¶1 *Type:* SHOULD *applies to:* Router

Context: The implementation processes IPv6 packets. The implementation does not support the functions of the Flow Label field.

Requirement: The implementation passes the field on unchanged.

RFC text: `{{Hosts or routers that do not support the functions of the Flow Label field are required to set the field to zero when originating a packet, pass the field on unchanged when forwarding a packet, and ignore the field when receiving a packet}}`.

RQ_COR_1109 Flow Label [Process]

RFC 2460 *Clause:* 6 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation receives IPv6 packets. The implementation does not support the functions of the Flow Label field.

Requirement: The implementation ignores the field when receiving a packet.

RFC text: `{{Hosts or routers that do not support the functions of the Flow Label field are required to set the field to zero when originating a packet, pass the field on unchanged when forwarding a packet, and ignore the field when receiving a packet}}`.

RQ_COR_1110 Traffic Class [Generate]

RFC 2460 *Clause:* 7 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation generates IPv6 packets for which it needs to identify and distinguish between different classes or priorities of IPv6 packets.

Requirement: The implementation uses Traffic Class field in the IPv6 header.

RFC text: `{{The 8-bit Traffic Class field in the IPv6 header is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets}}`.

RQ_COR_1111 Traffic Class [Process]

RFC 2460 *Clause:* 7 ¶1 *Type:* MAY *applies to:* Router

Context: The implementation processes IPv6 packets for which it needs to identify and distinguish between different classes or priorities of IPv6 packets.

Requirement: The implementation uses Traffic Class field in the IPv6 header.

RFC text: `{{The 8-bit Traffic Class field in the IPv6 header is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets}}`.

RQ_COR_1112 Traffic Class [Generate]

RFC 2460 *Clause:* 7 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet using the Traffic Class field in the IPv6 header.

Requirement: The implementation provides a means for an upper-layer protocol to supply the value of the Traffic Class bits to the IPv6 service for packets originated by that upper-layer protocol.

RFC text: `{{The service interface to the IPv6 service within a node must provide a means for an upper-layer protocol to supply the value of the Traffic Class bits in packets originated by that upper-layer protocol}}`.

RQ_COR_1113 Traffic Class [Generate]

RFC 2460 *Clause:* 7 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation processes a packet using the Traffic Class field in the IPv6 header.

Requirement: The implementation provides a means an upper-layer protocol to supply the value of the Traffic Class bits in packets originated by that upper-layer protocol.

RFC text: `{{The service interface to the IPv6 service within a node must provide a means for an upper-layer protocol to supply the value of the Traffic Class bits in packets originated by that upper-layer protocol}}`.

RQ_COR_1114 Traffic Class [Generate]

RFC 2460 *Clause:* 7 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation generates a packet.

Requirement: The implementation uses zero as default value for all 8 bits in the Traffic Class field.

RFC text: `{{The default value must be zero for all 8 bits}}`.

RQ_COR_1115 Traffic Class [Process]

RFC 2460 *Clause:* 7 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation processes a packet.

Requirement: The implementation uses zero as default value for all 8 bits in the Traffic Class field.

RFC text: `{{The default value must be zero for all 8 bits}}`.

RQ_COR_1116 Traffic Class [Generate]

RFC 2460 *Clause:* 7 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation generates a packet. The Implementation supports a specific (experimental or eventual standard) use of some or all of the Traffic Class bits.

Requirement: The implementation changes the value of those Traffic Class bits in packets as is required for that specific use.

RFC text: `{{Nodes that support a specific (experimental or eventual standard) use of some or all of the Traffic Class bits are permitted to change the value of those bits in packets that they originate, forward, or receive, as required for that specific use. }}`.

RQ_COR_1117 Traffic Class [Process]

RFC 2460 *Clause:* 7 ¶3 *Type:* MAY *applies to:* Router

Context: The implementation processes a packet. The Implementation supports a specific (experimental or eventual standard) use of some or all of the Traffic Class bits.

Requirement: The implementation changes the value of those Traffic Class bits in packets as is required for that specific use.

RFC text: `{{Nodes that support a specific (experimental or eventual standard) use of some or all of the Traffic Class bits are permitted to change the value of those bits in packets that they originate, forward, or receive, as required for that specific use. }}`.

RQ_COR_1118 Traffic Class [Process]

RFC 2460 *Clause:* 7 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation receives a packet. The Implementation supports a specific (experimental or eventual standard) use of some or all of the Traffic Class bits.

Requirement: The implementation changes the value of those bits in packets as is required for that specific use.

RFC text: `{{Nodes that support a specific (experimental or eventual standard) use of some or all of the Traffic Class bits are permitted to change the value of those bits in packets that they originate, forward, or receive, as required for that specific use. }}`.

RQ_COR_1119 Traffic Class [Process]

RFC 2460 *Clause:* 7 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation does not support a specific (experimental or eventual standard) use of some or all of the Traffic Class bits.

Requirement: The implementation ignores and leaves unchanged any bits of the Traffic Class field for which it does not support a specific use.

RFC text: `{{Nodes should ignore and leave unchanged any bits of the Traffic Class field for which they do not support a specific use}}`.

RQ_COR_1120 Checksum [Compute]

RFC 2460 *Clause:* 8.1 ¶1,5 *Type:* MUST *applies to:* Node

Context: The implementation sends an IPv6 packet with a UDP "pseudo-header".

Requirement: The implementation computes a UDP checksum over the packet and the UDP pseudo-header.

RFC text: Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration shows the TCP and UDP "pseudo-header" for IPv6:... `{{...Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header}}`. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

RQ_COR_1121 Checksum [Compute]

RFC 2460 *Clause:* 8.1 ¶1,5 *Type:* MUST *applies to:* Node

Context: The implementation sends an IPv6 packet with a UDP "pseudo-header". The implementation computes a UDP checksum over the packet and the UDP pseudo-header. That computation yields a result of zero.

Requirement: The implementation changes the UDP checksum to hex FFFF for placement in the UDP header.

RFC text: Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration shows the TCP and UDP "pseudo-header" for IPv6:... `{{...Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header}}`. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

RQ_COR_1122 Checksum [Process]

RFC 2460 *Clause:* 8.1 ¶1,5 *Type:* MUST *applies to:* Node

Context: The implementation receives an IPv6 packet with a UDP "pseudo-header". The UDP checksum is zero.

Requirement: The implementation discards the UDP packet.

RFC text: Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration shows the TCP and UDP "pseudo-header" for IPv6:... ...Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. `{{IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error}}`.

RQ_COR_1123 Checksum [Process]

RFC 2460 *Clause:* 8.1 ¶1,5 *Type:* SHOULD *applies to:* Node

Context: The implementation receives an IPv6 packet with a UDP "pseudo-header". The implementation discards the UDP packet because the UDP checksum is zero.

Requirement: The implementation logs the error.

RFC text: Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration shows the TCP and UDP "pseudo-header" for IPv6:... ...Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. `{{IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error}}`.

RQ_COR_1124 IPv6 Packet [Generate]

RFC 2460 *Clause:* 8.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation sends an IPv6 packet.

Requirement: The implementation does not enforce maximum packet lifetime.

RFC text: `{{Unlike IPv4, IPv6 nodes are not required to enforce maximum packet lifetime. That is the reason the IPv4 "Time to Live" field was renamed "Hop Limit" in IPv6}}`.

RQ_COR_1125 Flow Label [Generate]

RFC 2460 *Clause:* A ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation generates flows requiring special handling by the intervening routers.

Requirement: The implementation uniquely identifies the flows by the combination of a source address and a non-zero flow label. Packets that do not belong to a flow are identified with a flow label of zero.

RFC text: A flow is a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers. There may be multiple active flows from a source to a destination, as well as traffic that is not associated with any flow. `{{A flow is uniquely identified by the combination of a source address and a non-zero flow label. Packets that do not belong to a flow carry a flow label of zero}}`.

RQ_COR_1126 Flow Label [Generate]

RFC 2460 *Clause:* A ¶3-5 *Type:* MUST *applies to:* Node

Context: The implementation generates flows for requiring special handling by the intervening routers. The implementation uniquely identifies the flows by the combination of a source address and a non-zero flow label.

Requirement: The implementation chooses new flow labels (pseudo-)randomly and uniformly from the range 1 to FFFFF hex. The implementation sends all packets belonging to the same flow with the same source address, destination address, and flow label. The implementation does not reuse a flow label for a new flow within the maximum lifetime of any flow-handling state that might have been established for the prior use of that flow label.

RFC text: `{{A flow label is assigned to a flow by the flow's source node. New flow labels must be chosen (pseudo-)randomly and uniformly from the range 1 to FFFFF hex. The purpose of the random allocation is to make any set of bits within the Flow Label field suitable for use as a hash key by routers, for looking up the state associated with the flow}}`. `{{...All packets belonging to the same flow must be sent with the same source address, destination address, and flow label}}`. `{{...A source must not re- use a flow label for a new flow within the maximum lifetime of any flow-handling state that might have been established for the prior use of that flow label}}`.

RQ_COR_1127 Flow Label [Generate]

RFC 2460 *Clause:* A ¶4 *Type:* MUST *applies to:* Node

Context: The implementation generates flows requiring desires special handling by the intervening routers. The implementation uniquely identifies the flows by the combination of a source address and a non-zero flow label. All packets belonging to the same flow are sent with the same source address, destination address, and flow label. The packets includes a Hop-by-Hop Options header.

Requirement: The implementation generates all flow packets with the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header).

RFC text: `{{All packets belonging to the same flow must be sent with the same source address, destination address, and flow label. If any of those packets includes a Hop-by-Hop Options header, then they all must be originated with the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header)}}`.

RQ_COR_1128 Flow Label [Generate]

RFC 2460 *Clause:* A ¶4 *Type:* MUST *applies to:* Node

Context: The implementation generates flows requiring desires special handling by the intervening routers. The implementation uniquely identifies the flows by the combination of a source address and a non-zero flow label. All packets belonging to the same flow are sent with the same source address, destination address, and flow label. The packets includes a Routing header.

Requirement: The implementation generates all the flow packets containing the same contents in all extension headers up to and including the Routing header (excluding the Next Header field in the Routing header).

RFC text: `{{All packets belonging to the same flow must be sent with the same source address, destination address, and flow label. If any of those packets includes a Routing header, then they all must be originated with the same contents in all extension headers up to and including the Routing header (excluding the Next Header field in the Routing header)}}`.

RQ_COR_1129 Flow Label [Process]

RFC 2460 *Clause:* A ¶4 *Type:* MAY *applies to:* Node

Context: The implementation receives flows.

Requirement: The implementation verifies that the following conditions are satisfied. a. All packets belonging to the same flow arrive with the same source address, destination address, and flow label. b. If any of those packets includes a Hop-by-Hop Options header, then they all have the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header). c. If any of those packets includes a Routing header, then they all have the same contents in all extension headers up to and including the Routing header (excluding the Next Header field in the Routing header).

RFC text: All packets belonging to the same flow must be sent with the same source address, destination address, and flow label. If any of those packets includes a Hop-by-Hop Options header, then they all must be originated with the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header). If any of those packets includes a Routing header, then they all must be originated with the same contents in all extension headers up to and including the Routing header (excluding the Next Header field in the Routing header). `{{The routers or destinations are permitted, but not required, to verify that these conditions are satisfied}}`. If a violation is detected, it should be reported to the source by an ICMP Parameter Problem message, Code 0, pointing to the high-order octet of the Flow Label field (i.e., offset 1 within the IPv6 packet).

RQ_COR_1130 Flow Label [Process]

RFC 2460 *Clause:* A ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation receives flows. The implementation checks that the following conditions are satisfied. a. All packets belonging to the same flow arrive with the same source address, destination address, and flow label. b. If any of those packets includes a Hop-by-Hop Options header, then they all have the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header). c. If any of those packets includes a Routing header, then they all have the same contents in all extension headers up to and including the Routing header (excluding the Next Header field in the Routing header). One or more of these conditions is not met

Requirement: The implementation reports the violation to the source by an ICMP Parameter Problem message, Code 0, pointing to the high-order octet of the Flow Label field (i.e., offset 1 within the IPv6 packet).

RFC text: All packets belonging to the same flow must be sent with the same source address, destination address, and flow label. If any of those packets includes a Hop-by-Hop Options header, then they all must be originated with the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header). If any of those packets includes a Routing header, then they all must be originated with the same contents in all extension headers up to and including the Routing header (excluding the Next Header field in the Routing header). The routers or destinations are permitted, but not required, to verify that these conditions are satisfied. `{{If a violation is detected, it should be reported to the source by an ICMP Parameter Problem message, Code 0, pointing to the high-order octet of the Flow Label field (i.e., offset 1 within the IPv6 packet)}}`.

RQ_COR_1131 Flow Label [Process]

RFC 2460 *Clause:* A ¶6 *Type:* MUST *applies to:* Node

Context: The implementation generates flows requiring special handling by the intervening routers. The implementation stops and restarts.

Requirement: The implementation does not use a flow label value that was active before the stop and restart and whose lifetime has not expired.

RFC text: `{{When a node stops and restarts (e.g., as a result of a "crash"), it must be careful not to use a flow label that it might have used for an earlier flow whose lifetime may not have expired yet}}`.

RQ_COR_9000 Hop by Hop Header [Process]

RFC 2460 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with a Hop-by-Hop Options header. The Destination Address in the IPv6 Header is the implementation's address.

Requirement: The implementation examines and processes the extension header.

RFC text: `{{ Hop-by-Hop Options header, which carries information that must be examined and processed by every node along a packet's delivery path, including the source and destination nodes}}`.

RQ_COR_9001 Hop by Hop Header [Process]

RFC 2460 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet with a Hop-by-Hop Options header. The Source Address in the IPv6 Header is the implementation's address.

Requirement: The implementation examines and processes the extension header.

RFC text: `{{ Hop-by-Hop Options header, which carries information that must be examined and processed by every node along a packet's delivery path, including the source and destination nodes}}`.

RQ_COR_9002 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header and does not recognize the Option Type. The highest-order two bits of the Option Type in the extension header are 10. The Destination Address of the packet is a multicast address.

Requirement: The implementation discards the packet and sends an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

RFC text: `{{The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type}}`.

RQ_COR_9003 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header and does not recognize the Option Type. The highest-order two bits of the Option Type in the extension header are 11. The Destination Address of the packet is a multicast address.

Requirement: Implementation discards the packet.

RFC text: `{{The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type}}`.

RQ_COR_9004 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶15 *Type:* MUST *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header. PadN is used to align subsequent options and to pad out the containing header to a multiple of 8 octets in length.

Requirement: Implementation recognizes the PadN option.

RFC text: There are two padding options which are used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length. `{{These padding options must be recognized by all IPv6 implementations}}`.

RQ_COR_9005 Extension Header Options [Process]

RFC 2460 *Clause:* 4.2 ¶12 *Type:* MAY *applies to:* Node

Context: The implementation processes a Hop-by-Hop or Destination Options extension header with any other Option Type besides the Pad1 and PadN.

Requirement: The implementation uses unique Option Type value for either Hop-by-Hop Options header or Destination Options header.

RFC text: `{{The same Option Type numbering space is used for both the Hop-by-Hop Options header and the Destination Options header. However, the specification of a particular option may restrict its use to only one of those two headers}}`.

RQ_COR_9006 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶39,41 *Type:* SHOULD *applies to:* Node

Context: The implementation receives "Fragment packets". The length of a fragment, as derived from the fragment packet's Payload Length field, is not a multiple of 8 octets and the M flag of that fragment is 1. The implementation discards that fragment.

Requirement: The implementation sends an ICMP Parameter Problem, Code 0, message to the source of the fragment, pointing to the Payload Length field of the fragment packet.

RFC text: The following error conditions may arise when reassembling fragmented packets:... `{{...If the length of a fragment, as derived from the fragment packet's Payload Length field, is not a multiple of 8 octets and the M flag of that fragment is 1, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Payload Length field of the fragment packet}}`.

RQ_COR_9007 Fragment Packets [Process]

RFC 2460 *Clause:* 4.5 ¶39,42 *Type:* SHOULD *applies to:* Node

Context: The implementation receives "Fragment packets". The length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets. The implementation discards that fragment.

Requirement: The implementation sends an ICMP Parameter Problem, Code 0, message to the source of the fragment pointing to the Fragment Offset field of the fragment packet.

RFC text: The following error conditions may arise when reassembling fragmented packets... `{{...If the length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Fragment Offset field of the fragment packet}}`.

RQ_COR_9008 PMTU Discovery

RFC 2460 *Clause:* 5 ¶4 *Type:* MAY *applies to:* Node

Context: A minimal IPv6 implementation is attached to one or more links.

Requirement: The implementation does not implement Path MTU Discovery.

RFC text: `{{...However, a minimal IPv6 implementation (e.g., in a boot ROM) may simply restrict itself to sending packets no larger than 1280 octets, and omit implementation of Path MTU Discovery}}`.

4.7 Requirements extracted from RFC 2461

RQ_COR_8100 Neighbor Discovery

RFC 2461 *Clause:* 2.1 "on-link" *Type:* MUST *applies to:* Node

Context: The implementation's address is assigned to an interface on a specified link and is covered by one of the link's prefixes.

Requirement: The implementation considers the address to be on-link.

RFC text: on-link - an address that is assigned to an interface on a specified link.
`{{A node considers an address to be on-link if: ... - it is covered by one of the link's prefixes}}`

RQ_COR_8101 Neighbor Discovery by Redirect Message

RFC 2461 *Clause:* 2.1 "on-link" *Type:* MUST *applies to:* Node

Context: A neighboring router specifies an address assigned to the implementation's interface on a link as the target/Destination Address of a Redirect message.

Requirement: The implementation considers the target/Destination Address to be on-link.

RFC text: on-link - an address that is assigned to an interface on a specified link.
`{{A node considers an address to be on-link if: ... a neighboring router specifies the address as the target of a Redirect message}}`, or

RQ_COR_8102 Neighbor Discovery by NA

RFC 2461 *Clause:* 2.1 "on-link" *Type:* MUST *applies to:* Node

Context: The implementations receives a valid Neighbor Advertisement.

Requirement: The implementation considers the address in the advertisement's Target Address field to be on-link.

RFC text: on-link - an address that is assigned to an interface on a specified link.
`{{A node considers an address to be on-link if: ... a Neighbor Advertisement message is received for the (target) address}}`, or

RQ_COR_8103 Neighbor Discovery by Redirect

RFC 2461 *Clause:* 2.1 "on-link" *Type:* MUST *applies to:* Node

Context: The implementations receives a valid Neighbor Discovery message.

Requirement: The implementation considers the Source Address of the Neighbor Discovery message to be on-link.

RFC text: on-link - an address that is assigned to an interface on a specified link.
`{{A node considers an address to be on-link if: ... any Neighbor Discovery message is received from the address.}}`

RQ_COR_8104 Neighbor Discovery

RFC 2461 *Clause:* 2.1 "longest *Type:* MUST *applies to:* Node

Context: The implementation has multiple prefixes covering the same target address.

Requirement: The implementation uses the longest prefix as the one that matches.

RFC text: longest prefix match
- The process of determining which prefix (if any) in a set of prefixes covers a target address. A target address is covered by a prefix if all of the bits in the prefix match the left-most bits of the target address.
`{{When multiple prefixes cover an address, the longest prefix is the one that matches.}}`

RQ_COR_8105 Neighbor Discovery Messages [Generate]

RFC 2461 *Clause:* 2.1 "random *Type:* MUST *applies to:* Node

Context: The implementation computes random component delays.

Requirement: The implmentation generates uniformly-distributed random values that fall between the specified minimum and maximum delay times.

RFC text: random delay
when sending out messages, it is sometimes necessary to delay a transmission for a random amount of time in order to prevent multiple nodes from transmitting at exactly the same time, or to prevent long-range periodic transmissions from synchronizing with each other [SYNC]. `{{When a random component is required, a node calculates the actual delay in such a way that the computed delay forms a uniformly-distributed random value that falls between the specified minimum and maximum delay times.}}` The implementor take care to insure that the granularity of the calculated random component and the resolution of the timer used are both high enough to insure that the probability of multiple nodes delaying the same amount of time is small.

RQ_COR_8106 Neighbor Discovery Messages [Generate]

RFC 2461 *Clause:* 2.1 "random *Type:* SHOULD *applies to:* Node

Context: The implementation uses a pseudo-random number generator to calculate random delay components.

Requirement: The implementation initializes the generator with a unique seed that prevents successive generation of the same random numbers sequences.

RFC text: random delay seed
-If a pseudo-random number generator is used in calculating a random delay component, `{{the generator should be initialized with a unique seed prior to being used.}}` Note that it is not sufficient to use the interface token alone as the seed, since interface tokens will not always be unique. To reduce the probability that duplicate interface tokens cause the same seed to be used, the seed should be calculated from a variety of input sources (e.g., machine components) that are likely to be different even on identical "boxes". For example, the seed could be formed by combining the CPU's serial number with an interface token.

RQ_COR_8107 address: Link-local [Form]

RFC 2461 *Clause:* 2.3 "link-local *Type:* MUST *applies to:* Router

Context: The implementation is generating addresses for its interfaces.

Requirement: The implementation assigns a valid link-local address to each of its interfaces.

RFC text: link-local address
- a unicast address having link-only scope that can be used to reach neighbors. `{{All interfaces on routers MUST have a link-local address.}}` Also, [ADDRCONF] requires that interfaces on hosts have a link-local address.

RQ_COR_8108 address: Link-local [Form]

RFC 2461 *Clause:* 2.3 "link-local *Type:* MUST *applies to:* Host

Context: The implementation is generating addresses for its interfaces.

Requirement: The implementation assigns a valid link-local address to each of its interfaces.

RFC text: link-local address
- a unicast address having link-only scope that can be used to reach neighbors. All interfaces on routers MUST have a link-local address. `{{Also, [ADDRCONF] requires that interfaces on hosts have a link-local address.}}`

RQ_COR_8109 address: Destination Address [Generate]

RFC 2461 *Clause:* 2.3 *Type:* MUST *applies to:*

Context: The implementation is generating a Destination Address

Requirement: The implementation does not use the Unspecificied Address (0::0) as a Destination Address.

RFC text: unspecified address
- a reserved address value that indicates the lack of an address (e.g., the address is unknown). `{{It is never used as a destination address}}`, but may be used as a source address if the sender does not (yet) know its own address (e.g., while verifying an address is unused during address autoconfiguration [ADDRCONF]). The unspecified address has a value of 0:0:0:0:0:0:0:0.

RQ_COR_8110 Router Solicitation [Generate]

RFC 2461 *Clause:* 3 "Router *Type:* MAY *applies to:* Host

Context: The implementation's interface becomes enabled and immediately wants a Router Advertisement.

Requirement: The implementation transmits a Router Solicitation.

RFC text: Router Solicitation: When an interface becomes enabled, `{{hosts may send out Router Solicitations that request routers to generate Router Advertisements immediately }}`rather than at their next scheduled time.

RQ_COR_8111 Router Advertisement [Generate]

RFC 2461 *Clause:* 3 "Router *Type:* MUST *applies to:* Router

Context: The implementation is running on a multicast-capable link.

Requirement: The implementation periodically transmits a Router Advertisement to advertise its presence together with various link and Internet parameters.

RFC text: Router Advertisement: `{{Routers advertise their presence together with various link and Internet parameters either periodically}}`, or in response to a Router Solicitation message. Router Advertisements contain prefixes that are used for on-link determination and/or address configuration, a suggested hop limit value, etc.

RQ_COR_8112 Router Solicitation [Process]

RFC 2461 *Clause:* 3 "Router *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation message.

Requirement: The implementation transmits a Router Advertisement message to indicate its presence together with various link and Internet parameters.

RFC text: Router Advertisement: `{{Routers advertise their presence together with various link and Internet parameters}}` either periodically, or `{{in response to a Router Solicitation message}}`. Router Advertisements contain prefixes that are used for on-link determination and/or address configuration, a suggested hop limit value, etc.

RQ_COR_8113 Neighbor Solicitation [Generate]

RFC 2461 *Clause:* 3 "Neighbor *Type:* MUST *applies to:* Node

Context: The implementation needs to determine the link-layer address of a neighbor.

Requirement: The implementation generates and transmits a Neighbor Solicitation message.

RFC text: Neighbor Solicitation: `{{Sent by a node to determine the link-layer address of a neighbor}}`, or to verify that a neighbor is still reachable via a cached link-layer address. Neighbor Solicitations are also used for Duplicate Address Detection.

RQ_COR_8114 Neighbor Solicitation [Generate]

RFC 2461 *Clause:* 3 "Neighbor *Type:* MUST *applies to:* Node

Context: The implementation needs to verify if a neighbor is still reachable for a given address.

Requirement: The implementation generates and transmits a Neighbor Solicitation message.

RFC text: Neighbor Solicitation: `{{Sent by a node}}` to determine the link-layer address of a neighbor, or `{{to verify that a neighbor is still reachable via a cached link-layer address}}`. Neighbor Solicitations are also used for Duplicate Address Detection.

RQ_COR_8115

RFC 2461 *Clause:* 3 "Neighbor *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message.

Requirement: The implementation transmits a Neighbor Advertisement message in response.

RFC text: Neighbor Advertisement: `{{A response to a Neighbor Solicitation message.}}` A node may also send unsolicited Neighbor Advertisements to announce a link-layer address change.

RQ_COR_8116 Neighbor Advertisement: Solicited NA [Generate]

RFC 2461 *Clause:* 3 "Neighbor *Type:* MAY *applies to:* Node

Context: The implementation has changed one of its link-layer addresses.

Requirement: The implementation transmits an unsolicited Neighbor Advertisement message to announce the link-layer address change.

RFC text: Neighbor Advertisement: A response to a Neighbor Solicitation message. `{{A node may also send unsolicited Neighbor Advertisements to announce a link-layer address change.}}`

RQ_COR_8117

RFC 2461 *Clause:* 3 "Redirect" *Type:* MUST *applies to:* Router

Context: The implementation has determined a better first-hop node to reach a particular destination.

Requirement: The implementation transmits a Redirect message to the node transmitting packets to the particular destination.

RFC text: Redirect: How a router informs a host of a better first-hop node to reach a particular destination.

RQ_COR_8118 Router Advertisement [Process]

RFC 2461 *Clause:* 3 "Inbound *Type:* MUST *applies to:* Node

Context: The implementation has received a Router Advertisement with a missing source link-layer address.

Requirement: The implementation transmits a Neighbor Solicitation message to learn the link-layer address of the router transmitting the Router Advertisement.

RFC text: Load balancing is handled by allowing routers to omit the source link-layer address from Router Advertisement packets, thereby `{{forcing neighbors to use Neighbor Solicitation messages to learn link-layer addresses of routers.}}` Returned Neighbor Advertisement messages can then contain link-layer addresses that differ depending on who issued the solicitation.

RQ_COR_8119 Neighbor Discovery

RFC 2461 *Clause:* 3.2 "point-to- *Type:* SHOULD *applies to:* Node

Context: The implementation is on a point-to-point link.

Requirement: The implementation implements Neighbor Discovery as described in RFC 2461.

RFC text: point-to-point - Neighbor Discovery handles such links just like multicast links. (Multicast can be trivially provided on point to point links, and interfaces can be assigned link-local addresses.) `{{Neighbor Discovery should be implemented as described in this document.}}`

RQ_COR_8120 Neighbor Discovery

RFC 2461 *Clause:* 3.2 "multicast" *Type:* SHOULD *applies to:* Node

Context: The implementation is on a multicast link.

Requirement: The implementation implements Neighbor Discovery as described in RFC 2461.

RFC text: `{{multicast - Neighbor Discovery should be implemented as described in this document.}}`

RQ_COR_8121 Neighbor Discovery

RFC 2461 *Clause:* 3.2 "non- *Type:* SHOULD *applies to:* Node

Context: The implementation is on a non-broadcast multiple access (NBMA)link.

Requirement: The implementation implements Redirect, Neighbor Unreachability Detection and next-hop determination as described in RFC 2461.

RFC text: `{{non-broadcast multiple access (NBMA) - Redirect, Neighbor Unreachability Detection and next-hop determination should be implemented as described in this document.}}` Address resolution, and the mechanism for delivering Router Solicitations and Advertisements on NBMA links is not specified in this document. Note that if hosts support manual configuration of a list of default routers, hosts can dynamically acquire the link-layer addresses for their neighbors from Redirect messages.

RQ_COR_8122 PMTU: Multicast PMTU [Discover]

RFC 2461 *Clause:* 3.2 "variable *Type:* MUST *applies to:* Node

Context: The implementation is on a multicast link.

Requirement: The implementation uses the same MTU as all other nodes on the multicast link.

RFC text: variable MTU - Neighbor Discovery allows routers to specify a MTU for the link, which all nodes then use. `{{All nodes on a link must use the same MTU (or Maximum Receive Unit) in order for multicast to work properly.}}` Otherwise when multicasting a sender, which can not know which nodes will receive the packet, could not determine a minimum packet size all receivers can process.

RQ_COR_8123 Neighbor Unreachability Detection

RFC 2461 *Clause:* 3.2 *Type:* SHOULD *applies to:* Node

Context: The node detects the absence of symmetric reachability (half-links) using Neighbor Unreachability Detection.

Requirement: The node does not use paths that have asymmetric reachability.

RFC text: asymmetric reachability - Neighbor Discovery detects the absence of symmetric reachability; a node avoids paths to a neighbor with which it does not have symmetric connectivity.
`{{The Neighbor Unreachability Detection will typically identify such half-links and the node will refrain from using them.}}`
The protocol can presumably be extended in the future to find viable paths in environments that lack reflexive and transitive connectivity.

RQ_COR_8124 Router Solicitation [Generate]

RFC 2461 *Clause:* 4.1 ¶1 *Type:* SHOULD *applies to:* Host

Context: Tbe implementation wants to prompt routers to send Router Advertisements quickly.

Requirement: The implementation transmits a Router Solicitation.

RFC text: `{{Hosts send Router Solicitations in order to prompt routers to generate Router Advertisements quickly.}}`

RQ_COR_8125 Router Solicitation Header [Generate]

RFC 2461 *Clause:* 4.1 "Source *Type:* MUST *applies to:* Host

Context: The implementation is generating a Router Solicitation. An IP address is assigned to the implementation's address.

Requirement: The implementation places the IP address assigned to the sending interface in the Source Address field of the IP header of the Router Solicitation.

RFC text: Source Address
`{{An IP address assigned to the sending interface,}}` or the unspecified address if no address is assigned to the sending interface.

RQ_COR_8126 Router Solicitation Header [Generate]

RFC 2461 *Clause:* 4.1 "Source *Type:* MUST *applies to:* Host

Context: The implementation is generating a Router Solicitation. No address is assigned to the implementation's address.

Requirement: The implementation places the Unspecified Address (0::0) in the Source Address field of the IP header of the Router Solicitation.

RFC text: Source Address
An IP address assigned to the sending interface, or `{{the unspecified address if no address is assigned to the sending interface}}`.

RQ_COR_8127 Router Solicitation Header [Generate]

RFC 2461 *Clause:* 4.1 *Type:* SHOULD *applies to:* Host

Context: The implementation is generating a Router Solicitation.

Requirement: The implementation places the all-routers multicast address in the Destination Address field of the IP header of the Router Solicitation.

RFC text:
```
{{Destination Address
 Typically the all-routers multicast address.}}
```

RQ_COR_8128 Router Solicitation Header [Generate]

RFC 2461 *Clause:* 4.1 "Hop Limit" *Type:* MUST *applies to:* Host

Context: The implementation is generating a Router Solicitation.

Requirement: The implementation sets the Hop Limit field of the IP header of the Router Solicitation to 255.

RFC text: `{{Hop Limit 255 }}`

RQ_COR_8129 Router Solicitation Header [Generate]

RFC 2461 *Clause:* 4.1 *Type:* SHOULD *applies to:* Host

Context: The implementation is generating a Router Solicitation. A Security Association exists between the implementation and the destination address.

Requirement: The implementation includes the Authentication Header in the Router Solicitation.

RFC text: Authentication Header
If a Security Association for the IP Authentication Header exists between the sender and the destination address, then the sender SHOULD include this header.

RQ_COR_8130 Router Solicitation Header [Generate]

RFC 2461 *Clause:* 4.1 "ICMP *Type:* MUST *applies to:* Host

Context: The implementation is generating a Router Solicitation.

Requirement: The implementation sets the following ICMP Fields: Type is set to 133, Code is set to 0, Checksum is set to the ICMP checksum, and the Reserved field is set to zero.

RFC text:
```
{{ICMP Fields:
Type            133
Code            0
Checksum        The ICMP checksum.  See [ICMPv6].
Reserved        This field is unused. It MUST be initialized to zero
by the sender}}
```
and MUST be ignored by the receiver.

RQ_COR_8131 Router Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 4.1 "ICMP *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation and the ICMP Reserved field is set to any value.

Requirement: The implementation ignores any value in the Reserved field.

RFC text: ICMP Fields:
Type 133
Code 0
Checksum The ICMP checksum. See [ICMPv6].
Reserved This field is unused. It MUST be initialized to zero by the sender and `{{MUST be ignored by the receiver}}`.

RQ_COR_8132 Router Solicitation: Source Link-Layer Address

RFC 2461 *Clause:* 4.1 "Options: *Type:* SHOULD *applies to:* Host

Context: The implementation is generating a Router Solicitation and has a known source link-layer address.

Requirement: The implementation includes the Source link-layer address option in the Router Solicitation.

RFC text: `{{Source link-layer address`
`The link-layer address of the sender, if known.}}` MUST NOT be included if the Source Address is the unspecified address. `{{Otherwise it SHOULD be included on link layers that have addresses.}}`

RQ_COR_8133 Router Solicitation: Source Link-Layer Address

RFC 2461 *Clause:* 4.1 "Options: *Type:* MUST *applies to:* Host

Context: The implementation is generating a Router Solicitation. The solicitation's Source Address is the Unspecified Address (0::0).

Requirement: The implementation does not include the Source link-layer address option in the Router Advertisement.

RFC text: Source link-layer address
The link-layer address of the sender, if known. `{{MUST NOT be included if the Source Address is the unspecified address.}}` Otherwise it SHOULD be included on link layers that have addresses.3GrqmtTxt:

RQ_COR_8134 Router Solicitation [Process]

RFC 2461 *Clause:* 4.1 "Valid *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation with an unrecognizable option type.

Requirement: The implementation ignores the unrecognizable option type and continues processing the Router Solicitation.

RFC text: Valid Options:
Future versions of this protocol may define new option types. `{{Receivers MUST silently ignore any options they do not recognize and continue processing the message.}}`

RQ_COR_8135 Router Advertisement Header [Form]

RFC 2461 *Clause:* 4.2 *Type:* MUST *applies to:* Router

Context: The implementation is functioning on a link.

Requirement: The implementation periodically transmits Router Advertisement messages. The advertisement's destination address is set to the all-nodes multicast address. The advertisement's Source Address in the IPv6 Header is set to the link-local address assigned to the advertisement's sending interface. The Hop Limit in the IPv6 Header is set to 255.

RFC text: `{{Routers send out Router Advertisement message periodically}}`, or in response to a Router Solicitation.
`{{Destination Address}}`
Typically the Source Address of an invoking Router Solicitation or `{{the all-nodes multicast addressDestination Address.}}`
`{{Source Address`
`MUST be the link-local address assigned to the interface from which this message is sent}}`.
`{{Hop Limit      255}}`

RQ_COR_8136 Router Solicitation [Process]

RFC 2461 *Clause:* 4.2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation.

Requirement: The implementation transmits a Router Advertisement message. The advertisement's destination address is set to the Source Address of the received solicitation. The advertisement's Source Address in the IPv6 Header is set to the link-local address assigned to the advertisement's sending interface. The Hop Limit in the IPv6 Header is set to 255.

RFC text: {{Routers}} send out Router Advertisement message periodically, or in response to a Router Solicitation{{...
Destination Address
Typically {{the Source Address of an invoking Router Solicitation}} or the all-nodes multicast address.
{{Source Address
MUST be the link-local address assigned to the interface from which this message is sent}}.
{{Hop Limit 255}}

RQ_COR_8137 Router Advertisement Header [Form]

RFC 2461 *Clause:* 4.2 *Type:* SHOULD *applies to:* Router

Context: The implementation is generating a Router Advertisement. A Security Association exists between the implementation and the destination address.

Requirement: The implementation includes the Authentication Header in the Router Advertisement packet.

RFC text: {{Authentication Header
If a Security Association for the IP Authentication Header exists between the sender and the destination address, then the sender SHOULD include this header}}.

RQ_COR_8138 Router Advertisement Header [Form]

RFC 2461 *Clause:* 4.2 "ICMP *Type:* MUST *applies to:* Router

Context: The implementation is generating a Router Advertisement.

Requirement: The implementation sets the following ICMP field values: Type is set to 134. Code is set to 0. Checksum is set to the calculated checksum. The 6-bit Reserved field is set all zeros.

RFC text:
```
{{ICMP Fields:
Type           134
Code           0
Checksum       The ICMP checksum
Reserved       A 6-bit unused field. It MUST be initialized to zero
by the sender and MUST be ignored by the receiver.}}
```

RQ_COR_8139 Router Advertisement: [Process] Field Anomalies

RFC 2461 *Clause:* 4.2 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement with the ICMP Reserved field set to a value other than 0.

Requirement: The implmentation ignores the ICMP Reserved field value.

RFC text: Reserved
A 6-bit unused field. It MUST be initialized to zero by the sender and {{MUST be ignored by the receiver}}.

RQ_COR_8140

RFC 2461 *Clause:* 4.2 "ICMP *Type:* SHOULD *applies to:* Hosts

Context: The implementation has a received a Router Advertisement with Router Lifetime set to 0.

Requirement: The implementation does not use the router sending that advertisement as one of its default routers.

RFC text: Router Lifetime
16-bit unsigned integer. The lifetime associated with the default router in units of seconds. The maximum value corresponds to 18.2 hours. {{A Lifetime of 0 indicates that the router is not a default router and SHOULD NOT appear on the default router list.}} The Router Lifetime applies only to the router's usefulness as a default router; it does not apply to information contained in other message fields or options. Options that need time limits for their information include their own lifetime fields.

RQ_COR_8141

RFC 2461 *Clause:* 4.2 "Options - *Type:* MUST *applies to:* Router

Context: The implementation enables inbound load sharing across multiple link-layer addresses.

Requirement: The implementation omits the Source link-layer address option in Router Advertisement messages for the multiple link-layer addresses.

RFC text: Source link-layer address
The link-layer address of the interface from which the Router Advertisement is sent. Only used on link layers that have addresses. {{A router MAY omit this option in order to enable inbound load sharing across multiple link-layer addresses}}.

RQ_COR_8142 Router Advertisement: MTU Option

RFC 2461 *Clause:* 4.2 "Options - *Type:* SHOULD *applies to:* Router

Context: The implementation is on a link that has a variable MTU, e.g. Ethernet.

Requirement: The implementation includes the MTU option in Router Advertisements that it transmits.

RFC text: {{MTU
SHOULD be sent on links that have a variable MTU }}(as specified in the document that describes how to run IP over the particular link type). MAY be sent on other links.

RQ_COR_8143 Router Advertisement: MTU Option

RFC 2461 *Clause:* 4.2 "Options - *Type:* MAY *applies to:* Router

Context: The implementation is on a link does not have a variable MTU.

Requirement: The implementation includes the MTU option in Router Advertisements that it transmits.

RFC text: MTU
SHOULD be sent on links that have a variable MTU (as specified in the document that describes how to run IP over the particular link type). {{MAY be sent on other links}}.

RQ_COR_8144 Router Advertisement: Prefix Option

RFC 2461 *Clause:* 4.2 "Options - *Type:* SHOULD *applies to:* Router

Context: The implmentation is generating a Router Advertisement that includes the Prefix Information option.

Requirement: The implementation include all its on-link prefixes (except the link-local prefix) in the Prefix Information option.

RFC text: Prefix Information
These options specify the prefixes that are on-link and/or are used for address autoconfiguration. `{{A router SHOULD include all its on-link prefixes (except the link-local prefix) so that multihomed hosts have complete prefix information about on-link destinations for the links to which they attach}}`. If complete information is lacking, a multihomed host may not be able to choose the correct outgoing interface when sending traffic to its neighbors.

RQ_COR_8145

RFC 2461 *Clause:* 4.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement containing options that it cannot recognize.

Requirement: The implementation ignores the unrecognizable options in the advertisement and continues processing the advertisement.

RFC text: Future versions of this protocol may define new option types. `{{Receivers MUST silently ignore any options they do not recognize and continue processing the message.}}`

RQ_COR_8146 Address Resolution

RFC 2461 *Clause:* 4.3 ¶1 and *Type:* MUST *applies to:* Node

Context: The node needs to resolve an IP address with a corresponding link-layer address.

Requirement: The node transmits a Neighbor Solicitation with the solicited-node multicast address corresponding to the target address as the Destination Address.

RFC text: `{{Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target. Neighbor Solicitations are multicast when the node needs to resolve an address}}` and unicast when the node seeks to verify the reachability of a neighbor.
...
Destination Address
Either `{{the solicited-node multicast address corresponding to the target address}}`, or the target address.

RQ_COR_8147 Neighbor Reachability Determination

RFC 2461 *Clause:* 4.3 ¶1 and *Type:* MUST *applies to:*

Context: The node seeks to verify the reachability of a neighbor on the link.

Requirement: The node transmits a Neighbor Solicitation with the neighbor's address as the Destination Address.

RFC text: `{{Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target}}`. Neighbor Solicitations are multicast when the node needs to resolve an address and `{{unicast when the node seeks to verify the reachability of a neighbor.}}`
...
Destination Address
Either the solicited-node multicast address corresponding to the target address, or `{{the target address}}`.

RQ_COR_8148 address: Duplicate Address Detection (DAD)

RFC 2461 *Clause:* 4.3 "IP Fields - *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation for use in Duplicate Address Detection.

Requirement: The Source Address for the Neighbor Solicitation is set to the Unspecified Address (0::0).

RFC text: Source Address
Either an address assigned to the interface from which this message is sent or `{{(if Duplicate Address Detection is in progress [ADDRCONF]) the unspecified address.}}`

RQ_COR_8149 Neighbor Solicitation Header [Generate]

RFC 2461 *Clause:* 4.3 "IP Fields - *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation for any use other than in Duplicate Address Detection.

Requirement: The Source Address for the Neighbor Solicitation is set to an address assigned to the interface from with the solicitation is sent.

RFC text: Source Address
`{{Either an address assigned to the interface from which this message is sent}}` or (if Duplicate Address Detection is in progress [ADDRCONF]) the unspecified address.

RQ_COR_8150 Neighbor Solicitation Header [Generate]

RFC 2461 *Clause:* 4.3 "IP Fields - *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation.

Requirement: The implementation sets the Hop Limit value in the IPv6 Header of the solicitation to 255.

RFC text: `{{Hop Limit      255}}`

RQ_COR_8151 Neighbor Solicitation Header [Generate]

RFC 2461 *Clause:* 4.3 "IP Fields - *Type:* SHOULD *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation. A security association exists between the implementation and the destination address.

Requirement: The implementation includes the Authentication Header in the Neighbor Solicitation packet.

RFC text:
```
{{Authentication Header
 If a Security Association for the IP Authentication Header exists
between the sender and the destination address, then the sender
SHOULD include this header}}.
```

RQ_COR_8152 Neighbor Solicitation Header [Generate]

RFC 2461 *Clause:* 4.3 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation.

Requirement: The implementation sets the following ICMP field values: Type is set to 135. Code is set to 0. Checksum is set to the calculated checksum. Reserved is set to 0. The Target Address field is set to the IP address of the solicitation's target.

RFC text: ICMP Fields:

```
 {{Type           135
  Code           0
  Checksum       The ICMP checksum.
  Reserved       This field is unused. It MUST be initialized to zero
by the sender and MUST be ignored by the receiver.
  Target Address
  The IP address of the target of the solicitation.}}
```

It MUST NOT be a multicast address.

RQ_COR_8153

RFC 2461 *Clause:* 4.3 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation with the Reserved field set to a value other than zero.

Requirement: The implementation ignores the value in the Reserved field.

RFC text: Reserved
This field is unused. It MUST be initialized to zero by the sender and `{{MUST be ignored by the receiver}}`.

RQ_COR_8154 Neighbor Solicitation Header [Generate]

RFC 2461 *Clause:* 4.3 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation.

Requirement: The implementation does not place a multicast IP address in the Target field of the solicitation.

RFC text: Target Address
The IP address of the target of the solicitation. `{{It MUST NOT be a multicast address.}}`

RQ_COR_8155 Neighbor Solicitation Option [Generate]

RFC 2461 *Clause:* 4.3 "Options: *Type:* MUST *applies to:* Node

Context: The node is generating a Neighbor Solicitation with the Source Address set to the Unspecified Address (0::0).

Requirement: The implementation omits the source link-layer address option in the solicitation.

RFC text: Source link-layer address
The link-layer address for the sender. `{{MUST NOT be included when the source IP address is the unspecified address}}`. Otherwise, on link layers that have addresses this option MUST be included in multicast solicitations and SHOULD be included in unicast solicitations.

RQ_COR_8156 Neighbor Solicitation Option [Generate]

RFC 2461 *Clause:* 4.3 "Options: *Type:* MUST *applies to:* Node

Context: The node is generating a Neighbor Solicitation on a link layer that has addresses. The Source Address is set to a value other than the Unspecified Address. The Destination Address of the solicitation is a multicast address.

Requirement: The implementation includes the source link-layer address option in the solicitation.

RFC text: Source link-layer address
The link-layer address for the sender. MUST NOT be included when the source IP address is the unspecified address. Otherwise, `{{on link layers that have addresses this option MUST be included in multicast solicitations}}` and SHOULD be included in unicast solicitations.

RQ_COR_8157 Neighbor Solicitation Option [Generate]

RFC 2461 *Clause:* 4.3 "Options: *Type:* SHOULD *applies to:* Node

Context: The node is generating a Neighbor Solicitation on a link layer that has addresses. The Source Address is set to a value other than the Unspecified Address. The Destination Address of the solicitation is a unicast address.

Requirement: The implementation omits the source link-layer address option in the solicitation.

RFC text: Source link-layer address
The link-layer address for the sender. MUST NOT be included when the source IP address is the unspecified address. Otherwise, on link layers that have addresses this option MUST be included in multicast solicitations and `{{SHOULD be included in unicast solicitations}}`.

RQ_COR_8158

RFC 2461 *Clause:* 4.3 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation with unrecognizable options.

Requirement: The implementation silently ignores the unrecognizable options and continues process the solicitation

RFC text: Future versions of this protocol may define new option types. `{{Receivers MUST silently ignore any options they do not recognize and continue processing the message}}`.

RQ_COR_8159 Neighbor Solicitation [Process]

RFC 2461 *Clause:* 4.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation.

Requirement: The implementation sends a Neighbor Advertisement message in response to the solicitation.

RFC text: `{{A node sends Neighbor Advertisements in response to Neighbor Solicitations}}` and sends unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly.

RQ_COR_8160 Neighbor Advertisement: Solicited NA [Generate]

RFC 2461 *Clause:* 4.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation decides to propagate new information quickly.

Requirement: The implementation sends an unsolicited Neighbor Advertisement message.

RFC text: A node sends Neighbor Advertisements in response to Neighbor Solicitations and `{{sends unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly}}`.

RQ_COR_8161 Neighbor Advertisement Header [Generate]

RFC 2461 *Clause:* 4.4 "IP Fields: *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Advertisement message.

Requirement: The implementations sets the following IP field values: The Source Address field is set to an address assigned to the interface sending the advertisement. The Hop Limit field is set to 255.

RFC text: IP Fields:
```
{{Source Address - An address assigned to the interface from which
the advertisement is sent.
...
Hop Limit      255}}
```

RQ_COR_8162 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 4.4 "IP Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation message with a Source Address set to a value other than the Unspecified Address.

Requirement: The implementation transmits a Neighbor Advertisement with the Destination Address set to the Source Address of the received solicitation.

RFC text: Destination Address
`{{For solicited advertisements, the Source Address of an invoking Neighbor Solicitation}}` or, if the solicitation's Source Address is the unspecified address, the all-nodes multicast address.

RQ_COR_8163 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 4.4 "IP Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation message with a Source Address set to the Unspecified Address (0::0).

Requirement: The implementation transmits a Neighbor Advertisement with the Destination Address set to the all-nodes multicast address.

RFC text: Destination Address
For solicited advertisements, the Source Address of an invoking Neighbor Solicitation or, `{{if the solicitation's Source Address is the unspecified address, the all-nodes multicast address}}`.

RQ_COR_8164

RFC 2461 *Clause:* 4.4 "IP Fields: *Type:* SHOULD *applies to:* Node

Context: The implementation is generating an unsolicited Neighbor Advertisement.

Requirement: The implementation sets the Destination Address of the advertisement's IP header to the all-nodes multicast address.

RFC text: Destination Address
For solicited advertisements, the Source Address of an invoking Neighbor Solicitation or, if the solicitation's Source Address is the unspecified address, the all-nodes multicast address.
```
{{For unsolicited advertisements typically the all-nodes multicast
address.}}
```

RQ_COR_8165 Neighbor Advertisement [Generate]

RFC 2461 *Clause:* 4.4 "IP Fields: *Type:* SHOULD *applies to:* Node

Context: The implementation is generating a Neighbor Advertisement. A Security Association exists between the implementation and the destination address.

Requirement: The implementation includes the IP Authentication Header in the Neighbor Advertisement's packet.

RFC text: Authentication Header
`{{If a Security Association for the IP Authentication Header exists between the sender and the destination address, then the sender SHOULD include this header}}`.

RQ_COR_8166 Neighbor Advertisement Header [Form]

RFC 2461 *Clause:* 4.4 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Advertisement.

Requirement: The implementation sets the following ICMP Fields in the Neighbor Advertisement: Type is set to 136, Code is set to 0, Checksum is set to the ICMP checksum, and the Reserved field is a 29-bit field set to zero.

RFC text:

```
{{ICMP Fields:
 Type          136
 Code          0
 Checksum      The ICMP checksum.
...
 Reserved      29-bit unused field. It MUST be initialized to zero
by the sender}}
```
and MUST be ignored by the receiver.

RQ_COR_8167 Process Field Anomalies in NA

RFC 2461 *Clause:* 4.4 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement with the ICMP Reserved field set to a value other than zero.

Requirement: The implementation ignores the value in the ICMP Reserved field of the advertisement.

RFC text: Reserved
29-bit unused field. It MUST be initialized to zero by the sender and `{{MUST be ignored by the receiver}}`.

RQ_COR_8168 Neighbor Advertisement Header [Form]

RFC 2461 *Clause:* 4.4 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Advertisement with a multicast address in the Destination Address of the IP header.

Requirement: The implementation sets the S-bit of the ICMP Field of the advertisement to zero.

RFC text: S
Solicited flag. When set, the S-bit indicates that the advertisement was sent in response to a Neighbor Solicitation from the Destination address. The S-bit is used as a reachability confirmation for Neighbor Unreachability Detection. `{{It MUST NOT be set in multicast advertisements}}` or in unsolicited unicast advertisements.

RQ_COR_8169 Neighbor Advertisement: Unsolicited NA Header

RFC 2461 *Clause:* 4.4 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation is generating an unsolicited Neighbor Advertisement with a unicast address in the Destination Address of the IP header.

Requirement: The implementation sets the S-bit of the ICMP Field of the advertisement to zero.

RFC text: S
Solicited flag. When set, the S-bit indicates that the advertisement was sent in response to a Neighbor Solicitation from the Destination address. The S-bit is used as a reachability confirmation for Neighbor Unreachability Detection. `{{It MUST NOT be set}}` in multicast advertisements or `{{in unsolicited unicast advertisements}}`.

RQ_COR_8170 Neighbor Advertisement [Process]

RFC 2461 *Clause:* 4.4 "ICMP *Type:* SHOULD *applies to:* Node

Context: The implementation receives a Neighbor Advertisement with the O-bit set 1, a Target Address field, and a Target Link-layer Address option.

Requirement: The implementation changes the link-layer address associated to the IP address in the Target Address field to the new link-layer address shown in the Advertisements's Target Link-layer Address option.

RFC text: O
Override flag. `{{When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address}}`. When it is not set the advertisement will not update a cached link-layer address though it will update an existing Neighbor Cache entry for which no link-layer address is known. It SHOULD NOT be set in solicited advertisements for anycast addresses and in solicited proxy advertisements. It SHOULD be set in other solicited advertisements and in unsolicited advertisements.

RQ_COR_8171 Neighbor Advertisement [Process]

RFC 2461 *Clause:* 4.4 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation has an IP address already associated with a link-layer address. The implementation then receives a Neighbor Advertisement with the O-bit set 0, a Target Address field, and a Target Link-layer Address option. The IP address in the Target Address field is the same as IP address already associated to a link-layer address. However, the link-layer address in the Target Link-layer Address option is different than that already associated with the IP address.

Requirement: The implementation does not change the association of the link-layer address to the IP address in the Target Address field.

RFC text: O
Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. `{{When it is not set the advertisement will not update a cached link-layer address}}` though it will update an existing Neighbor Cache entry for which no link-layer address is known. It SHOULD NOT be set in solicited advertisements for anycast addresses and in solicited proxy advertisements. It SHOULD be set in other solicited advertisements and in unsolicited advertisements.

RQ_COR_8172 Neighbor Advertisement [Process]

RFC 2461 *Clause:* 4.4 "ICMP *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement with the O-bit set to 0, a Target Address field, and a Target Link-layer Address option. The implementation has no link-layer address associated to the IP address in the Target Address field.

Requirement: The implementation associates the new link-layer address shown in the Advertisements's Target Link-layer Address option to the IP address in the Target Address field.

RFC text: O
Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. When it is not set the advertisement will not update a cached link-layer address `{{though it will update an existing Neighbor Cache entry for which no link-layer address is known}}`. It SHOULD NOT be set in solicited advertisements for anycast addresses and in solicited proxy advertisements. It SHOULD be set in other solicited advertisements and in unsolicited advertisements.

RQ_COR_8173 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 4.4 "ICMP *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation with a unicast address in the Source Address field of the solicitations IP Header.

Requirement: The implementation transmits a Neighbor Advertisement with the O-bit set to zero.

RFC text: O
Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. When it is not set the advertisement will not update a cached link-layer address though it will update an existing Neighbor Cache entry for which no link-layer address is known. `{{It SHOULD NOT be set in solicited advertisements for anycast addresses}}` and in solicited proxy advertisements. It SHOULD be set in other solicited advertisements and in unsolicited advertisements.

RQ_COR_8174 Neighbor Advertisement: Solicited NA [Process]

RFC 2461 *Clause:* 4.4 "ICMP *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. The implementation is acting as a proxy for the address in the Destination Address field of the solicitation's IP Header.

Requirement: The implementation transmits a Neighbor Advertisement with the O-bit set to zero.

RFC text: O
Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. When it is not set the advertisement will not update a cached link-layer address though it will update an existing Neighbor Cache entry for which no link-layer address is known. `{{It SHOULD NOT be set}}` in solicited advertisements for anycast addresses and `{{in solicited proxy advertisements}}`. It SHOULD be set in other solicited advertisements and in unsolicited advertisements.

RQ_COR_8175 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 4.4 "ICMP *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation that does not have a unicast address in the Source Address field of the solicitations IP Header nor is the implementation is acting as a proxy for the address in the Destination Address field of the solicitation's IP Header.

Requirement: The implementation transmits a Neighbor Advertisement with the O-bit set to one.

RFC text: O
Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. When it is not set the advertisement will not update a cached link-layer address though it will update an existing Neighbor Cache entry for which no link-layer address is known. It SHOULD NOT be set in solicited advertisements for anycast addresses and in solicited proxy advertisements. `{{It SHOULD be set in other solicited advertisements}}` and in unsolicited advertisements.

RQ_COR_8176 Neighbor Advertisement: Unsolicited NA Header

RFC 2461 *Clause:* 4.4 "ICMP *Type:* SHOULD *applies to:* Node

Context: The implementation is generating an unsolicited Neighbor Advertisement.

Requirement: The implementation transmits the Neighbor Advertisement with the O-bit set to one.

RFC text: O
Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. When it is not set the advertisement will not update a cached link-layer address though it will update an existing Neighbor Cache entry for which no link-layer address is known. It SHOULD NOT be set in solicited advertisements for anycast addresses and in solicited proxy advertisements. `{{It SHOULD be set}}` in other solicited advertisements and `{{in unsolicited advertisements}}`.

RQ_COR_8177 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 4.4 "Target *Type:* MUST *applies to:* Node

Context: The implementation has received a valid Neighbor Solicitation with a non-multicast address in the Target Address field. The implementation must respond to the solicitation.

Requirement: The implementation transmits a Neighbor Advertisement with its Target Address field set to the same address in the solicitation's Target Address field.

RFC text: Target Address
`{{For solicited advertisements, the Target Address field in the Neighbor Solicitation message that prompted this advertisement.}}` For an unsolicited advertisement, the address whose link-layer address has changed. `{{The Target Address MUST NOT be a multicast address}}`.

RQ_COR_8178

RFC 2461 *Clause:* 4.4 "Target *Type:* MUST *applies to:* Node

Context: The node is generating an unsolicited Neighbor Advertisement to notify neighbors that one of its link-layer addresses has changed.

Requirement: The implementation sets the advertisement's Target Address field to the IP address whose link-layer has changed. This IP address is not a muticast address.

RFC text: Target Address
For solicited advertisements, the Target Address field in the Neighbor Solicitation message that prompted this advertisement. `{{For an unsolicited advertisement, the address whose link-layer address has changed. The Target Address MUST NOT be a multicast address}}`.

RQ_COR_8179 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 4.4 "Options: *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation with a multicast Destination Address to which it must respond.

Requirement: The implementation transmits a Neighbor Advertisement with a Target Link-layer Address option.

RFC text: Target link-layer address
The link-layer address for the target, i.e., the sender of the advertisement. {{This option MUST be included on link layers that have addresses when responding to multicast solicitations.}} When responding to a unicast Neighbor Solicitation this option SHOULD be included.

RQ_COR_8180 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 4.4 "Options: *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation with a unicast Destination Address to which it must respond.

Requirement: The implementation transmits a Neighbor Advertisement with a Target Link-layer Address option.

RFC text: Target link-layer address
The link-layer address for the target, i.e., the sender of the advertisement. This option MUST be included on link layers that have addresses when responding to multicast solicitations. {{When responding to a unicast Neighbor Solicitation this option SHOULD be included.}}

RQ_COR_8181

RFC 2461 *Clause:* 4.4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement with an unrecognizable Option fields.

Requirement: The implementation silently ignores the unrecognizable Option fields and continue processing the advertisement.

RFC text: Future versions of this protocol may define new option types. {{Receivers MUST silently ignore any options they do not recognize and continue processing the message.}}

RQ_COR_8182 Redirect Message [Generate]

RFC 2461 *Clause:* 4.5 ¶1 *Type:* MUST *applies to:* Router

Context: The implemention is to inform a host of a better first-hop node on the path to a destination.

Requirement: The implementation sends a Redirect packet.

RFC text: {{Routers send Redirect packets to inform a host of a better first-hop node on the path to a destination.}} Hosts can be redirected to a better first-hop router but can also be informed by a redirect that the destination is in fact a neighbor. The latter is accomplished by setting the ICMP Target Address equal to the ICMP Destination Address.

RQ_COR_8183 Redirect Message [Generate]

RFC 2461 *Clause:* 4.5 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation is to inform a host that the destination address is, in fact, a neighbor on the same link.

Requirement: The implementation sends a Redirect packet with the Target Address field equal to the neighbor's Destination Address.

RFC text: Routers send Redirect packets to inform a host of a better first-hop node on the path to a destination. Hosts can be redirected to a better first-hop router but {{can also be informed by a redirect that the destination is in fact a neighbor. The latter is accomplished by setting the ICMP Target Address equal to the ICMP Destination Address}}.

RQ_COR_8184 Redirect Message [Generate]

RFC 2461 *Clause:* 4.5 "IP Fields:" *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect message.

Requirement: The implementation sets the following IP Header fields in the Redirect message: The Source Address is set to the link-local address assigned to the sending message's interface. The Destination Address field is set to the Source Address of the packet triggering the Redirect. The Hop Limit field is set to 255.

RFC text: IP Fields:
{{Source Address - MUST be the link-local address assigned to the interface from which this message is sent.
Destination Address - The Source Address of the packet that triggered the redirect.
Hop Limit - 255}}

RQ_COR_8185 Redirect Message [Generate]

RFC 2461 *Clause:* 4.5 "IP Fields: *Type:* SHOULD *applies to:* Router

Context: The implementation is generating a Redirect. A Security Association exists between the implementation and the destination address.

Requirement: The implementation includes the Authentication Header in the Redirect.

RFC text: Authentication Header
{{If a Security Association for the IP Authentication Header exists between the sender and the destination address, then the sender SHOULD include this header}}.

RQ_COR_8186 Redirect Message [Generate]

RFC 2461 *Clause:* 4.5 "ICMP *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect.

Requirement: The implementation sets the following ICMP fields: Type is set to 137. Code is set to 0. Checksum is set to the calculated checksum. Reserved is set to 0. The Destination Address is set to the IP address of the destination that is redirected.

RFC text: ICMP Fields:
{{Type - 137
Code - 0
Checksum - The ICMP checksum.
Reserved - This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
Destination Address - The IP address of the destination which is redirected to the target}}.

RQ_COR_8187

RFC 2461 *Clause:* 4.5 "ICMP *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message with the ICMP Reserved field set to any value other than 0.

Requirement: The implementation ignores the Reserved field of the the Redirect.

RFC text: `{{Reserved`
`This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.}}`

RQ_COR_8188 Redirect Target Address Field [Determine]

RFC 2461 *Clause:* 4.5 "ICMP *Type:* MUST *applies to:* Router

Context: The implementation informs a host that there is a better first hop router to use for a given Destination Address.

Requirement: The implementation sets the Redirect's ICMP Target Address field to the implementation's local-link address.

RFC text: Target Address
`{{An IP address that is a better first hop to use for the ICMP Destination Address}}`. When the target is the actual endpoint of communication, i.e., the destination is a neighbor, the Target Address field MUST contain the same value as the ICMP Destination Address field. `{{Otherwise the target is a better first-hop router and the Target Address MUST be the router's link-local address so that hosts can uniquely identify routers}}`.

RQ_COR_8189

RFC 2461 *Clause:* 4.5 "ICMP *Type:* MUST *applies to:* Router

Context: The implementation is to inform a host that the destination address is, in fact, a neighbor on the same link.

Requirement: The implementation sets the Redirect's ICMP Target Address field to the same value as the Redirect's IP header Destination Address field.

RFC text: Target Address
An IP address that is a better first hop to use for the ICMP Destination Address. `{{When the target is the actual endpoint of communication, i.e., the destination is a neighbor, the Target Address field MUST contain the same value as the ICMP Destination Address field}}`. Otherwise the target is a better first-hop router and the Target Address MUST be the router's link-local address so that hosts can uniquely identify routers.

RQ_COR_8190 Redirect Options [Generate]

RFC 2461 *Clause:* 4.5 "Possible *Type:* SHOULD *applies to:* Router

Context: The implementation is generating a Redirect message on a non-NBMA link with a known link-layer address for the target.

Requirement: The implementation includes the Target Link-layer Address option in the Redirect

RFC text: Target link-layer address
The link-layer address for the target. `{{It SHOULD be included (if known)}}`. Note that on NBMA links, hosts may rely on the presence of the Target Link-Layer Address option in Redirect messages as the means for determining the link-layer addresses of neighbors. In such cases, the option MUST be included in Redirect messages.

RQ_COR_8191 Redirect Options [Generate]

RFC 2461 *Clause:* 4.5 "Possible *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect message on an NBMA link with a known link-layer address for the target.

Requirement: The implementation includes the Target Link-layer Address option in the Redirect

RFC text: Target link-layer address
The link-layer address for the target. It SHOULD be included (if known). {{Note that on NBMA links, hosts may rely on the presence of the Target Link-Layer Address option in Redirect messages as the means for determining the link-layer addresses of neighbors. In such cases, the option MUST be included in Redirect messages}}.

RQ_COR_8192

RFC 2461 *Clause:* 4.5 "Possible *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect message that includes the Redirected Header option.

Requirement: The implementation places as much as possible of the IP packet that triggered the sending of the Redirect without making the Redirect packet exceed 1280 octets.

RFC text: Redirected Header
{{As much as possible of the IP packet that triggered the sending of the Redirect without making the redirect packet exceed 1280 octets}}.

RQ_COR_8193

RFC 2461 *Clause:* 4.6 "Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation packet with an option whose Length field is zero.

Requirement: The implementation silently discards the Neighbor Solicitation.

RFC text: Length
8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 octets. The value 0 is invalid. {{Nodes MUST silently discard an ND packet that contains an option with length zero}}.

RQ_COR_8194

RFC 2461 *Clause:* 4.6 "Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement packet with an option whose Length field is zero.

Requirement: The implementation silently discards the Neighbor Advertisement.

RFC text: Length
8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 octets. The value 0 is invalid. {{Nodes MUST silently discard an ND packet that contains an option with length zero}}.

RQ_COR_8195 Router Solicitation [Process]

RFC 2461 *Clause:* 4.6 "Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Solicitation packet with an option whose Length field is zero.

Requirement: The implementation silently discards the Router Solicitation.

RFC text: Length
8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 octets. The value 0 is invalid. `{{Nodes MUST silently discard an ND packet that contains an option with length zero}}`.

RQ_COR_8196

RFC 2461 *Clause:* 4.6 "Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement packet with an option whose Length field is zero.

Requirement: The implementation silently discards the Router Advertisement.

RFC text: Length
8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 octets. The value 0 is invalid. `{{Nodes MUST silently discard an ND packet that contains an option with length zero}}`.

RQ_COR_8197

RFC 2461 *Clause:* 4.6 "Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a Redirect packet with an option whose Length field is zero.

Requirement: The implementation silently discards the Redirect.

RFC text: Length
8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 octets. The value 0 is invalid. `{{Nodes MUST silently discard an ND packet that contains an option with length zero}}`.

RQ_COR_8198

RFC 2461 *Clause:* 4.6.1 "Fields:" *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation containing a Source Link-Layer Address option.

Requirement: The implementation sets the following fields of the solicitation's Source Link-Layer Address option: The Type field is set to 1. The field Length is to the length of the option (including the type and length fields) in units of 8 octets. The Link-Layer Address field is set to the link-layer address.

RFC text: Fields:

```
{{Type
1 for Source Link-layer Address
2 for Target Link-layer Address
Length
         The length of the option (including the type and length
fields) in units of 8 octets. For example, the length for IEEE 802
addresses is 1 [IPv6- ETHER]. Link-Layer Address
The variable length link-layer address.
The content and format of this field (including byte and bit
ordering) is expected to be specified in specific documents that
describe how IPv6 operates over different link layers. For instance,
[IPv6-ETHER].
Description
The Source Link-Layer Address option contains the link-layer address
of the sender of the packet. It is used in the Neighbor Solicitation,
Router Solicitation, and Router Advertisement packets}}.
```

The Target Link-Layer Address option contains the link-layer address of the target. It is used in Neighbor Advertisement and Redirect packets.

RQ_COR_8199 Router Solicitation Source Link-Layer Address

RFC 2461 *Clause:* 4.6.1 "Fields:" *Type:* MUST *applies to:* Node

Context: The implementation is generating a Router Solicitation containing a Source Link-Layer Address option.

Requirement: The implementation sets the following fields of the solicitation's Source Link-Layer Address option: The Type field is set to 1. The field Length is to the length of the option (including the type and length fields) in units of 8 octets. The Link-Layer Address field is set to the link-layer address.

RFC text: Fields:

```
{{Type
1 for Source Link-layer Address
2 for Target Link-layer Address
Length
         The length of the option (including the type and length
fields) in units of 8 octets. For example, the length for IEEE 802
addresses is 1 [IPv6- ETHER]. Link-Layer Address
The variable length link-layer address.
The content and format of this field (including byte and bit
ordering) is expected to be specified in specific documents that
describe how IPv6 operates over different link layers. For instance,
[IPv6-ETHER].
Description
The Source Link-Layer Address option contains the link-layer address
of the sender of the packet. It is used in the Neighbor Solicitation,
Router Solicitation, and Router Advertisement packets}}.
```

The Target Link-Layer Address option contains the link-layer address of the target. It is used in Neighbor Advertisement and Redirect packets.

RQ_COR_8200 Router Advertisement Source Link-Layer

RFC 2461 *Clause:* 4.6.1 "Fields:" *Type:* MUST *applies to:* Router

Context: The implementation is generating a Router Advertisement containing a Source Link-Layer Address option.

Requirement: The implementation sets the following fields of the advertisement's Source Link-Layer Address option: The Type field is set to 1. The field Length is to the length of the option (including the type and length fields) in units of 8 octets. The Link-Layer Address field is set to the link-layer address.

RFC text: Fields:

```
{{Type
1 for Source Link-layer Address
2 for Target Link-layer Address
Length
         The length of the option (including the type and length
fields) in units of 8 octets. For example, the length for IEEE 802
addresses is 1 [IPv6- ETHER]. Link-Layer Address
The variable length link-layer address.
The content and format of this field (including byte and bit
ordering) is expected to be specified in specific documents that
describe how IPv6 operates over different link layers. For instance,
[IPv6-ETHER]}}.
```

Description

The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.

```
{{The Target Link-Layer Address option contains the link-layer
address of the target. It is used in Neighbor Advertisement and
Redirect packets}}.
```

RQ_COR_8201 Neighbor Advertisement [Generate]

RFC 2461 *Clause:* 4.6.1 "Fields:" *Type:* MUST *applies to:* Router

Context: The implementation is generating a Neighbor Advertisement containing a Target Link-Layer Address option.

Requirement: The implementation sets the following fields of the advertisement's Target Link-Layer Address option: The Type field is set to 2. The field Length is to the length of the option (including the type and length fields) in units of 8 octets. The Link-Layer Address field is set to the link-layer address.

RFC text: Fields:

```
{{Type
1 for Source Link-layer Address
2 for Target Link-layer Address
Length
         The length of the option (including the type and length
fields) in units of 8 octets. For example, the length for IEEE 802
addresses is 1 [IPv6- ETHER]. Link-Layer Address
 The variable length link-layer address.
The content and format of this field (including byte and bit
ordering) is expected to be specified in specific documents that
describe how IPv6 operates over different link layers. For instance,
[IPv6-ETHER]}}.
```

Description

The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.

```
{{The Target Link-Layer Address option contains the link-layer
address of the target. It is used in Neighbor Advertisement and
Redirect packets}}.
```

RQ_COR_8202 Redirect Options [Generate]

RFC 2461 *Clause:* 4.6.1 "Fields:" *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect containing a Target Link-Layer Address option.

Requirement: The implementation sets the following fields of the redirect's Target Link-Layer Address option: The Type field is set to 2. The field Length is to the length of the option (including the type and length fields) in units of 8 octets. The Link-Layer Address field is set to the link-layer address.

RFC text: Fields:

```
{{Type
1 for Source Link-layer Address
2 for Target Link-layer Address
Length
         The length of the option (including the type and length
fields) in units of 8 octets. For example, the length for IEEE 802
addresses is 1 [IPv6- ETHER]. Link-Layer Address
The variable length link-layer address.
The content and format of this field (including byte and bit
ordering) is expected to be specified in specific documents that
describe how IPv6 operates over different link layers. For instance,
[IPv6-ETHER].}}
```

Description
The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.

```
{{The Target Link-Layer Address option contains the link-layer
address of the target. It is used in Neighbor Advertisement and
Redirect packets.}}
```

RQ_COR_8203

RFC 2461 *Clause:* 4.6.1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message containing a Target Link-Layer Address option.

Requirement: The implementation ignores the Target Link-Layer Address option [and processes the remainder of the solicition].

RFC text: Description
The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.
The Target Link-Layer Address option contains the link-layer address of the target. It is used in Neighbor Advertisement and Redirect packets.

```
{{These options MUST be silently ignored for other Neighbor Discovery
messages.}}
```

RQ_COR_8204 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 4.6.1 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation message containing a Target Link-Layer Address option.

Requirement: The implementation ignores the Target Link-Layer Address option [and processes the remainder of the solicition].

RFC text: Description
The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.
The Target Link-Layer Address option contains the link-layer address of the target. It is used in Neighbor Advertisement and Redirect packets.

```
{{These options MUST be silently ignored for other Neighbor Discovery
messages.}}
```

RQ_COR_8205 Router Advertisement - Option Anomalies

RFC 2461 *Clause:* 4.6.1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement message containing a Target Link-Layer Address option.

Requirement: The implementation ignores the Target Link-Layer Address option [and processes the remainder of the advertisement].

RFC text: Description
The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.
The Target Link-Layer Address option contains the link-layer address of the target. It is used in Neighbor Advertisement and Redirect packets.
`{{These options MUST be silently ignored for other Neighbor Discovery messages.}}`

RQ_COR_8206

RFC 2461 *Clause:* 4.6.1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message containing a Source Link-Layer Address option.

Requirement: The implementation ignores the Source Link-Layer Address option [and processes the remainder of the advertisement].

RFC text: Description
The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.
The Target Link-Layer Address option contains the link-layer address of the target. It is used in Neighbor Advertisement and Redirect packets.
`{{These options MUST be silently ignored for other Neighbor Discovery messages.}}`

RQ_COR_8207

RFC 2461 *Clause:* 4.6.1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message containing a Source Link-Layer Address option.

Requirement: The implementation ignores the Source Link-Layer Address option [and processes the remainder of the Redirect].

RFC text: Description
The Source Link-Layer Address option contains the link-layer address of the sender of the packet. It is used in the Neighbor Solicitation, Router Solicitation, and Router Advertisement packets.
The Target Link-Layer Address option contains the link-layer address of the target. It is used in Neighbor Advertisement and Redirect packets.
`{{These options MUST be silently ignored for other Neighbor Discovery messages.}}`

RQ_COR_8208 Router Advertisement Prefix Option

RFC 2461 *Clause:* 4.6.2 "Fields" *Type:* MUST *applies to:* Router

Context: The implementation is generating a Router Advertisement containing containing a Prefix Information option.

Requirement: The implementations sets the following fields of the Prefix Information option: The Type field is set to 3. The Length field is set to 4. The Prefix Length field is set to the number of leading bits in the Prefix field that are valid. The value can range from 0..128 (sic). The Reserved 1 field is set to 0. The Valid Lifetime field is set to the time in seconds relative to the time the advertisement is sent that the prefix is valid for on-link determination. The Preferred Lifetime field is set to the time in seconds relative to the time the advertisement is sent that addresses generated from the prefix by stateless autoconfiguration remain preferred. The Reserved2 field is set to 0. The Prefix field is set to address or the prefix of the IP address. The Prefix Length field is set to the number of valid leading bits in the Prefix field. The remaining bits in the Prefix field that are not part of the Prefix as indicated by the Prefix Length field are set to 0.

RFC text: Fields:

```
{{Type          3
Length        4
Prefix Length
8-bit unsigned integer. The number of leading bits in the Prefix that
are valid. The value ranges from 0 to 128.
...
Reserved1
6-bit unused field. It MUST be initialized to zero by the sender and
MUST be ignored by the receiver.
Valid Lifetime
32-bit unsigned integer. The length of time in seconds (relative to
the time the packet is sent) that the prefix is valid for the purpose
of on-link determination. A value of all one bits (0xffffffff)
represents infinity. The Valid Lifetime is also used by [ADDRCONF].
Preferred Lifetime
32-bit unsigned integer. The length of time in seconds (relative to
the time the packet is sent) that addresses generated from the prefix
via stateless address autoconfiguration remain preferred [ADDRCONF].
A value of all one bits (0xffffffff) represents infinity. See
[ADDRCONF].
Reserved2
This field is unused. It MUST be initialized to zero by the sender
and MUST be ignored by the receiver.
Prefix
An IP address or a prefix of an IP address. The Prefix Length field
contains the number of valid leading bits in the prefix. The bits in
the prefix after the prefix length are reserved and MUST be
initialized to zero by the sender and ignored by the receiver}}.
```

A router SHOULD NOT send a prefix option for the link-local prefix and a host SHOULD ignore such a prefix option.

RQ_COR_8209 Router Advertisement - Option Anomalies

RFC 2461 *Clause:* 4.6.2 "Fields" *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisment message with a Prefix Information option's Reserved1 and Reserved 2 fields set to any value other than zero. The option also has a Prefix field value with bits set to 1 after the number of bits shown in the Prefix Length field.

Requirement: The implementation ignores the values in the Reserved1 and Reserved fields and the additional bits in the Prefix field that are not included in the prefix as indicated by the Prefix Length field. [The implementation processes the remainder the advertisement.}

RFC text: Fields:
Type 3
Length 4
Prefix Length
8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128.
...
Reserved1
{{6-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver}}.
Valid Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity. The Valid Lifetime is also used by [ADDRCONF].
Preferred Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred [ADDRCONF]. A value of all one bits (0xffffffff) represents infinity. See [ADDRCONF].
{{Reserved2
This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver}}.
Prefix
An IP address or a prefix of an IP address. The Prefix Length field contains the number of valid leading bits in the prefix. {{The bits in the prefix after the prefix length are }}reserved and MUST be initialized to zero by the sender and {{ignored by the receiver}}. A router SHOULD NOT send a prefix option for the link-local prefix and a host SHOULD ignore such a prefix option.

RQ_COR_8210 **Router Advertisement [Process]**

RFC 2461 *Clause:* 4.6.2 "Fields: *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement message with a Prefix Information option's Valid and Preferred Lifetime fields each set to 0x0xffffffff.

Requirement: The implementation treats the both the valid and preferred lifetimes as infinite.

RFC text: Fields:
Type 3
Length 4
Prefix Length
8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128.
...
Reserved1
6-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
{{Valid Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity. The Valid Lifetime is also used by [ADDRCONF].
Preferred Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred [ADDRCONF]. A value of all one bits (0xffffffff) represents infinity.}} See [ADDRCONF].
Reserved2
This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
Prefix
An IP address or a prefix of an IP address. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length are reserved and MUST be initialized to zero by the sender and ignored by the receiver. A router SHOULD NOT send a prefix option for the link-local prefix and a host SHOULD ignore such a prefix option.

RQ_COR_8211

RFC 2461 *Clause:* 4.6.2 "Fields: *Type:* SHOULD *applies to:* Router

Context: The implementation is generating a Router Advertisement message.

Requirement: The Prefix Information option of the Router Advertisement message does not contain the link-local prefix.

RFC text: Fields:
Type 3
Length 4
Prefix Length
8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128.
...
Reserved1
6-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
Valid Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity. The Valid Lifetime is also used by [ADDRCONF].
Preferred Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred [ADDRCONF]. A value of all one bits (0xffffffff) represents infinity. See [ADDRCONF].
Reserved2
This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
Prefix
An IP address or a prefix of an IP address. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length are reserved and MUST be initialized to zero by the sender and ignored by the receiver. `{{A router SHOULD NOT send a prefix option for the link-local prefix }}`and a host SHOULD ignore such a prefix option.

RQ_COR_8212

RFC 2461 *Clause:* 4.6.2 "Fields: *Type:* SHOULD *applies to:* Host

Context: The implementation has received a Router Advertisement message with a Prefix Information option for a link-local prefix.

Requirement: The implementation ignores the Prefix Information option containing the link-local prefix [and processes the remainder of the advertisement].

RFC text: Fields:
Type 3
Length 4
Prefix Length
8-bit unsigned integer. The number of leading bits in the Prefix that are valid. The value ranges from 0 to 128.
...
Reserved1
6-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
Valid Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination. A value of all one bits (0xffffffff) represents infinity. The Valid Lifetime is also used by [ADDRCONF].
Preferred Lifetime
32-bit unsigned integer. The length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred [ADDRCONF]. A value of all one bits (0xffffffff) represents infinity. See [ADDRCONF].
Reserved2
This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
Prefix
An IP address or a prefix of an IP address. The Prefix Length field contains the number of valid leading bits in the prefix. The bits in the prefix after the prefix length are reserved and MUST be initialized to zero by the sender and ignored by the receiver. A router SHOULD NOT send a prefix option for the link-local prefix and `{{a host SHOULD ignore such a prefix option}}`.

RQ_COR_8213

RFC 2461 *Clause:* 4.6.2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message with a Prefix Information option.

Requirement: The implementation silently ignores the solicitation's Prefix Information option [and processes the remainder of the solicitation].

RFC text: Description
The Prefix Information option provide hosts with on-link prefixes and prefixes for Address Autoconfiguration.
The Prefix Information option appears in Router Advertisement packets and `{{MUST be silently ignored for other messages}}`.

RQ_COR_8214

RFC 2461 *Clause:* 4.6.2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message with a Prefix Information option.

Requirement: The implementation silently ignores the advertisement's Prefix Information option [and processes the remainder of the advertisement].

RFC text: Description
The Prefix Information option provide hosts with on-link prefixes and prefixes for Address Autoconfiguration.
The Prefix Information option appears in Router Advertisement packets and `{{MUST be silently ignored for other messages}}`.

RQ_COR_8215 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 4.6.2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation message with a Prefix Information option.

Requirement: The implementation silently ignores the solicitation's Prefix Information option [and processes the remainder of the solicitation].

RFC text: Description
The Prefix Information option provide hosts with on-link prefixes and prefixes for Address Autoconfiguration.
The Prefix Information option appears in Router Advertisement packets and `{{MUST be silently ignored for other messages}}`.

RQ_COR_8216

RFC 2461 *Clause:* 4.6.2 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message with a Prefix Information option.

Requirement: The implementation silently ignores the redirect's Prefix Information option [and processes the remainder of the redirect].

RFC text: Description
The Prefix Information option provide hosts with on-link prefixes and prefixes for Address Autoconfiguration.
The Prefix Information option appears in Router Advertisement packets and `{{MUST be silently ignored for other messages}}`.

RQ_COR_8217 Redirect Options [Generate]

RFC 2461 *Clause:* 4.6.3 "Fields:" *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect packet containing a Redirected Header option.

Requirement: The implementation sets the fields in the redirect's Redirected Header option to the following values: The Type field is set to 4. The Length field is set to option's length in units of 8 octets. The Reserved field is set to 0. The IP Header + Data field is a truncated copy of the original packet prompting the redirect. This last field is truncated to ensure that the total size of the Redirect is not greater than 1280 octets.

RFC text: Fields:

```
{{Type          4
Length         The length of the option in units of 8 octets.
Reserved
These fields are unused. They MUST be initialized to zero by the
sender and MUST be ignored by the receiver.
IP header + data
The original packet truncated to ensure that the size of the redirect
message does not exceed 1280 octets}}.
```

RQ_COR_8218 Redirect Message - Option Anomalies[Process]

RFC 2461 *Clause:* 4.6.3 "Fields: *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect packet with the Reserved field of the Redirected Header option set to a value other than 0.

Requirement: The implementation ignores the Redirected Header option having its Reserved field set a value other than zero [and processes the remainder of the Redirect packet].

RFC text: Fields:
Type 4
Length The length of the option in units of 8 octets.
Reserved
These fields are unused. They MUST be initialized to zero by the sender and `{{MUST be ignored by the receiver.}}`
IP header + data
The original packet truncated to ensure that the size of the redirect message does not exceed 1280 octets.

RQ_COR_8219

RFC 2461 *Clause:* 4.6.3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message containing a Redirected Header option.

Requirement: The implementation silently ignores the Redirected Header option [and processes the remainder of the solicitation].

RFC text: Description
The Redirected Header option is used in Redirect messages and contains all or part of the packet that is being redirected.
`{{This option MUST be silently ignored for other Neighbor Discovery messages}}`.

RQ_COR_8220

RFC 2461 *Clause:* 4.6.3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message containing a Redirected Header option.

Requirement: The implementation silently ignores the Redirected Header option [and processes the remainder of the Advertisement].

RFC text: Description
The Redirected Header option is used in Redirect messages and contains all or part of the packet that is being redirected.
`{{This option MUST be silently ignored for other Neighbor Discovery messages}}`.

RQ_COR_8221 Router Advertisement - Option Anomalies

RFC 2461 *Clause:* 4.6.3 *Type:* MUST *applies to:* Host

Context: The implementation receives a Router Advertisement message containing a Redirected Header option.

Requirement: The implementation silently ignores the Redirected Header option [and processes the remainder of the Advertisement].

RFC text: Description
The Redirected Header option is used in Redirect messages and contains all or part of the packet that is being redirected.
`{{This option MUST be silently ignored for other Neighbor Discovery messages}}`.

RQ_COR_8222 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 4.6.3 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation message containing a Redirected Header option.

Requirement: The implementation silently ignores the Redirected Header option [and processes the remainder of the Solicitation].

RFC text: Description
The Redirected Header option is used in Redirect messages and contains all or part of the packet that is being redirected.
`{{This option MUST be silently ignored for other Neighbor Discovery messages}}`.

RQ_COR_8223 RA MTU Option

RFC 2461 *Clause:* 4.6.4 "Fields:" *Type:* MUST *applies to:* Router

Context: The implementation is generating a Router Advertisement containing an MTU option.

Requirement: The implementation sets the following fields in the MTU option: The Type field is set to 5. The Length field is set to 1. The Reserved field is set to 0. The MTU field is sent to the recommended link MTU.

RFC text: Fields:

```
 {{Type          5
  Length        1
  Reserved
This field is unused. It MUST be initialized to zero by the sender
and MUST be ignored by the receiver.
  MTU
32-bit unsigned integer. The recommended MTU for the link.}}
```

RQ_COR_8224 Router Advertisement - Option Anomalies

RFC 2461 *Clause:* 4.6.4 "Fields: *Type:* MUST *applies to:* Host

Context: The implementation receives a Router Advertisement containing an MTU option whose Reserved field is set to a value other than 0.

Requirement: The implementation ignores the Reserved field of the advertisement's option [and processes the remainder of the option].

RFC text: Reserved
This field is unused. It MUST be initialized to zero by the sender and `{{MUST be ignored by the receiver}}`.

RQ_COR_8225

RFC 2461 *Clause:* 4.6.4 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement packet containing an MTU option.

Requirement: The implemention silently ignores the MTU option [and processes the remainder of the advertisement].

RFC text: Description
The MTU option is used in Router Advertisement messages to insure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known.
`{{This option MUST be silently ignored for other Neighbor Discovery messages}}`.

RQ_COR_8226

RFC 2461 *Clause:* 4.6.4 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation packet containing an MTU option.

Requirement: The implemention silently ignores the MTU option [and processes the remainder of the Solicitation].

RFC text: Description
The MTU option is used in Router Advertisement messages to insure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known.
{{This option MUST be silently ignored for other Neighbor Discovery messages}}.

RQ_COR_8227 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 4.6.4 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet containing an MTU option.

Requirement: The implemention silently ignores the MTU option [and processes the remainder of the Solicitation].

RFC text: Description
The MTU option is used in Router Advertisement messages to insure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known.
{{This option MUST be silently ignored for other Neighbor Discovery messages}}.

RQ_COR_8228

RFC 2461 *Clause:* 4.6.4 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect packet containing an MTU option.

Requirement: The implemention silently ignores the MTU option [and processes the remainder of the Redirect].

RFC text: Description
The MTU option is used in Router Advertisement messages to insure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known.
{{This option MUST be silently ignored for other Neighbor Discovery messages}}.

RQ_COR_8229

RFC 2461 *Clause:* 4.6.4 "Fields: *Type:* MUST *applies to:* Router

Context: The implementation is part of a network configuration where heterogeneous technologies are bridged together. The bridges in this network do not generate ICMP Packet Too Big messages.

Requirement: The implementation uses the MTU option to specify the maximum MTU value supported by all segments.

RFC text: Description
The MTU option is used in Router Advertisement messages to insure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known.
This option MUST be silently ignored for other Neighbor Discovery messages.
{{In configurations in which heterogeneous technologies are bridged together, the maximum supported MTU may differ from one segment to another. If the bridges do not generate ICMP Packet Too Big messages, communicating nodes will be unable to use Path MTU to dynamically determine the appropriate MTU on a per-neighbor basis. In such cases, routers use the MTU option to specify the maximum MTU value that is supported by all segments}}.

RQ_COR_8230 Router Advertisement [Process]

RFC 2461 *Clause:* 5.1 "Prefix List" *Type:* SHOULD *applies to:* Router

Context: The implementation is generating a Router Advertisement packet with a Prefix Information option for the link-local prefix.

Requirement: The advertisement's Prefix Information option does not change the link-local prefix's invalidation timer.

RFC text: Prefix List -
A list of the prefixes that define a set of addresses that are on-link. Prefix List entries are created from information received in Router Advertisements. Each entry has an associated invalidation timer value (extracted from the advertisement) used to expire prefixes when they become invalid. A special "infinity" timer value specifies that a prefix remains valid forever, unless a new (finite) value is received in a subsequent advertisement.
The link-local prefix is considered to be on the prefix list with an infinite invalidation timer regardless of whether routers are advertising a prefix for it. `{{Received Router Advertisements SHOULD NOT modify the invalidation timer for the link-local prefix}}`.

RQ_COR_8231 Router Advertisement: Host Processing of ...

RFC 2461 *Clause:* 5.4 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation is on a link with two or more routers transmitting Router Advertisement packets.

Requirement: The implementation uses at least two of the routers as its default routers.

RFC text: A node should retain entries in the Default Router List and the Prefix List until their lifetimes expire. However, a node may garbage collect entries prematurely if it is low on memory. If not all routers are kept on the Default Router list, `{{a node should retain at least two entries in the Default Router List (and preferably more) in order to maintain robust connectivity for off-link destinations}}`.

RQ_COR_8232 Next Hop Determination

RFC 2461 *Clause:* 5.4 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is on a link and transmitting packets to a destination through one of the default routers. This default router then stops forwarding packets to the destination. The node detects that packets are no longer arriving at the destination

Requirement: The implementation performs next-hop determination to select a new default router.

RFC text: When removing an entry from the Prefix List there is no need to purge any entries from the Destination or Neighbor Caches. Neighbor Unreachability Detection will efficiently purge any entries in these caches that have become invalid. `{{When removing an entry from the Default Router List, however, any entries in the Destination Cache that go through that router must perform next-hop determination again to select a new default router.}}`

RQ_COR_8233 Router Solicitation [Process]

RFC 2461 *Clause:* 6.1.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Router Solicitation packet.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{Hosts MUST silently discard any received Router Solicitation Messages}}`.

RQ_COR_8234 Router Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 6.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet with the IP Header Hop Limit field set to 255.

Requirement: The implementation silently discards the invalid solicitation.

RFC text: A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:
- {{The IP Hop Limit field has a value of 255}}, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 8 or more octets.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8235 Router Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 6.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet containing an IP Authentication Header. The packet fails authentication.

Requirement: The implementation silently discards the invalid solicitation.

RFC text: A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
{{- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.}}
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 8 or more octets.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8236 Router Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 6.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet. The packet fails the checksum comparison.

Requirement: The implementation silently discards the invalid solicitation.

RFC text: A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
{{- ICMP Checksum is valid. }}
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 8 or more octets.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8237 Router Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 6.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet. The ICMP code is set a value other than 0.

Requirement: The implementation silently discards the invalid solicitation.

RFC text: A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
`{{- ICMP Code is 0.}}`
- ICMP length (derived from the IP length) is 8 or more octets.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8238 Router Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 6.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet. The length of the ICMP part of the packet is less than 8 octets.

Requirement: The implementation silently discards the invalid solicitation.

RFC text: A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
`{{- ICMP length (derived from the IP length) is 8 or more octets}}`.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8239 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 6.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet containing an option whose Length field is set to 0.

Requirement: The implementation silently discards the invalid solicitation.

RFC text: A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 8 or more octets.
`{{- All included options have a length that is greater than zero}}`.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8240 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 6.1.1 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet with the Unspecified Address in the IP header Source Address field. There is no Source Link-layer Address option in the solicitation.

Requirement: The implementation silently discards the invalid solicitation.

RFC text: A router MUST silently discard any received Router Solicitation messages that do not satisfy all of the following validity checks:
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 8 or more octets.
- All included options have a length that is greater than zero.
`{{- If the IP source address is the unspecified address, there is no source link-layer address option in the message}}`.

RQ_COR_8241 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 6.1.1 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet containing an option that the implementation does not recognize.

Requirement: The implementation ignores the contents of the unrecognized option [and continues processing the remainder of the solicitation].

RFC text: `{{The contents}}` of the Reserved field, and `{{of any unrecognized options, MUST be ignored}}`. Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8242 Router Solicitation - Option Anomalies[Process]

RFC 2461 *Clause:* 6.1.1 ¶4 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation packet containing an option that it recognizes. This option is not specified to be used with the Router Solicitation; i.e any other option other than the Source Link-layer option.

Requirement: The implementation ignores the option that is not specified for use in the solicitation and processes the remainder of the packet.

RFC text: `{{The contents of any defined options that are not specified to be used with Router Solicitation messages MUST be ignored and the packet processed as normal}}`. The only defined option that may appear is the Source Link-Layer Address option.

RQ_COR_8243 Router Solicitation: Source Link-Layer Address

RFC 2461 *Clause:* 6.1.1 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation generates a Router Solicitation packet.

Requirement: The implementation includes only the Source Link-layer Address option in the packet.

RFC text: The contents of any defined options that are not specified to be used with Router Solicitation messages MUST be ignored and the packet processed as normal. `{{The only defined option that may appear is the Source Link-Layer Address option}}`.

RQ_COR_8244 Router Advertisement - Field Anomalies

RFC 2461 *Clause:* 6.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement packet with other than a link-local address in IP Header Source Address field.

Requirement: The implementation silently discards the invalid advertisement.

RFC text: A node MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:
`{{- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 16 or more octets.
- All included options have a length that is greater than zero.

RQ_COR_8245 Router Advertisement - Field Anomalies

RFC 2461 *Clause:* 6.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement packet with the IP Header Hop Limit field is set to a value other than 255.

Requirement: The implementation silently discards the invalid advertisement.

RFC text: A node MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
`{{- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router}}`.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 16 or more octets.
- All included options have a length that is greater than zero.

RQ_COR_8246 Router Advertisement - Field Anomalies

RFC 2461 *Clause:* 6.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement packet containing an IP Authentication Header. The packet fails authentication.

Requirement: The implementation silently discards the invalid advertisement.

RFC text: A node MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
`{{- If the message includes an IP Authentication Header, the message authenticates correctly}}`.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 16 or more octets.
- All included options have a length that is greater than zero.

RQ_COR_8247 Router Advertisement - Field Anomalies

RFC 2461 *Clause:* 6.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement. The calculated checksum does not match the checksum in the advertisement.

Requirement: The implementation silently discards the invalid advertisement.

RFC text: A node MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
`{{- ICMP Checksum is valid.}}`
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 16 or more octets.
- All included options have a length that is greater than zero.

RQ_COR_8248 Router Advertisement - Field Anomalies

RFC 2461 *Clause:* 6.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement containing an ICMP Code field set to a value other than 0.

Requirement: The implementation silently discards the invalid advertisement.

RFC text: A node MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
`{{- ICMP Code is 0}}`.
- ICMP length (derived from the IP length) is 16 or more octets.
- All included options have a length that is greater than zero.

RQ_COR_8249 Router Advertisement - Field Anomalies

RFC 2461 *Clause:* 6.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement. The length of the ICMP part of the packet is less than 16 octets.

Requirement: The implementation silently discards the invalid advertisement.

RFC text: A node MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
`{{- ICMP length (derived from the IP length) is 16 or more octets}}`.
- All included options have a length that is greater than zero.

RQ_COR_8250 Router Advertisement - Option Anomalies

RFC 2461 *Clause:* 6.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement containing an option whose Length field is set to 0.

Requirement: The implementation silently discards the invalid advertisement.

RFC text: A node MUST silently discard any received Router Advertisement messages that do not satisfy all of the following validity checks:
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 16 or more octets.
`{{- All included options have a length that is greater than zero}}`.

RQ_COR_8251 Router Advertisement - Option Anomalies

RFC 2461 *Clause:* 6.1.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement containing an option that the implementation does not recognize.

Requirement: The implementation ignores the contents of the unrecognized option [and continues processing the remainder of the advertisement].

RFC text: `{{The contents}}` of the Reserved field, and `{{of any unrecognized options, MUST be ignored}}`. Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8252 Router Advertisement - Option Anomalies

RFC 2461 *Clause:* 6.1.2 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement packet containing an option that it recognizes. This option is not specified to be used with the Router Advertisement; i.e any other option other than the Source Link-layer Address, Prefix Information, or MTU options.

Requirement: The implementation ignores the option that is not specified for use in the advertisement and processes the remainder of the packet.

RFC text: `{{The contents of any defined options that are not specified to be used with Router Advertisement messages MUST be ignored and the packet processed as normal}}`. The only defined options that may appear are the Source Link-Layer Address, Prefix Information and MTU options.

RQ_COR_8253 Router Advertisement Options [Form]

RFC 2461 *Clause:* 6.1.2 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation generates a Router Advertisement packet.

Requirement: The implementation includes only the Source Link-layer Address, Prefix Information and MTU options in the advertisement packet.

RFC text: The contents of any defined options that are not specified to be used with Router Advertisement messages MUST be ignored and the packet processed as normal. `{{The only defined options that may appear are the Source Link-Layer Address, Prefix Information and MTU options}}`.

RQ_COR_8254

RFC 2461 *Clause:* 6.1.2 ¶3 *Type: applies to:*

Context: The implementation generates a Router Advertisement packet.

Requirement: The implementation includes only the Source Link-layer Address, Prefix Information and MTU options in the advertisement packet.

RFC text: The contents of any defined options that are not specified to be used with Router Advertisement messages MUST be ignored and the packet processed as normal. `{{The only defined options that may appear are the Source Link-Layer Address, Prefix Information and MTU options}}`.

RQ_COR_8255 Router Advertisement Behavior Configuration

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface a flag to prohibit the implementation from both sending periodic Router Advertisements and responding to Router Solicitations.

RFC text:
```
{{A router MUST allow for the following conceptual variables to be configured by system management.
...
AdvSendAdvertisements
A flag indicating whether or not the router sends periodic Router Advertisements and responds to Router Solicitations.}}
```
Default: FALSE
Note that AdvSendAdvertisements MUST be FALSE by default so that a node will not accidentally start acting as a router unless it is explicitly configured by system management to send Router Advertisements.

RQ_COR_8256 Router Advertisement Behavior Configuration

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to prohibit the implementation from both sending periodic Router Advertisements and responding to Router Solicitations. This flag is left at its default value.

Requirement: The implementation does not periodically generate Router Advertisements.

RFC text: A router MUST allow for the following conceptual variables to be configured by system management.
...
AdvSendAdvertisements
A flag indicating whether or not the router sends periodic Router Advertisements and responds to Router Solicitations.
```
{{Default: FALSE
Note that AdvSendAdvertisements MUST be FALSE by default so that a node will not accidentally start acting as a router unless it is explicitly configured by system management to send Router Advertisements}}.
```

RQ_COR_8257 Router Advertisement Behavior Configuration

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to prohibit the implementation from both sending periodic Router Advertisements and responding to Router Solicitations. This flag is set to allow both periodic generation of Router Advertisements and responding to Router Solicitations.

Requirement: The implementation periodically generates Router Advertisements.

RFC text: A router MUST allow for the following conceptual variables to be configured by system management.
...
AdvSendAdvertisements
A flag indicating whether or not the router sends periodic Router Advertisements and responds to Router Solicitations.

```
{{Default: FALSE
Note that AdvSendAdvertisements MUST be FALSE by default so that a
node will not accidentally start acting as a router unless it is
explicitly configured by system management to send Router
Advertisements}}.
```

RQ_COR_8258 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface a timer to set in seconds for controlling the maximum time in seconds between sending unsolicited multicast Router Advertisements from the interface. The maximum time is no less than 4 seconds and no greater than 1800 seconds.

RFC text:

```
{{A router MUST allow for the following conceptual variables to be
configured by system management.
...
MaxRtrAdvInterval
The maximum time allowed between sending unsolicited multicast Router
Advertisements from the interface, in seconds. MUST be no less than 4
seconds and no greater than 1800 seconds.}}
```

Default: 600 seconds

RQ_COR_8259 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a timer to set in seconds for controlling the maximum time between sending unsolicited multicast Router Advertisements from the interface. The timer is not set during configuration thereby remaining at its default value.

Requirement: The implementation transmits unsolicited multicast Router Advertisements from the interface with periods whose maximum interval is 600 seconds.

RFC text:

```
{{A router MUST allow for the following conceptual variables to be
configured by system management.
...
MaxRtrAdvInterval
The maximum time allowed between sending unsolicited multicast Router
Advertisements from the interface, in seconds. MUST be no less than 4
seconds and no greater than 1800 seconds.
Default: 600 seconds}}
```

RQ_COR_8260 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a timer to set in seconds for controlling the maximum time between sending unsolicited multicast Router Advertisements from the interface. The timer is set during configuration to a value other than the default value.

Requirement: The implementation transmits unsolicited multicast Router Advertisements from the interface with periods whose maximum interval is the value set by system management during configuration.

RFC text:
```
{{A router MUST allow for the following conceptual variables to be
configured by system management.
...
MaxRtrAdvInterval
The maximum time allowed between sending unsolicited multicast Router
Advertisements from the interface, in seconds. MUST be no less than 4
seconds and no greater than 1800 seconds}}.
```
Default: 600 seconds

RQ_COR_8261 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface a timer to set in seconds for controlling the minimum time in seconds between sending unsolicited multicast Router Advertisements from the interface. The minimum time is no less than 3 seconds and no greater .75 times the maximum time for sending the same message (cf MaxRtrAdvInterval).

RFC text:
```
{{MinRtrAdvInterval
The minimum time allowed between sending unsolicited multicast Router
Advertisements from the interface, in seconds. MUST be no less than 3
seconds and no greater than .75 * MaxRtrAdvInterval.}}
```
Default: 0.33 * MaxRtrAdvInterval

RQ_COR_8262 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a timer to set in seconds for controlling the minimum time in seconds between sending unsolicited multicast Router Advertisements from the interface. The timer is not set during configuration thereby remaining at its default value.

Requirement: The implementation transmits unsolicited multicast Router Advertisements from the interface with periods whose minimum interval is is 1/3 of the maximum interval (cf MaxRtrAdvInterval).

RFC text: MinRtrAdvInterval
The minimum time allowed between sending unsolicited multicast Router Advertisements from the interface, in seconds. MUST be no less than 3 seconds and no greater than .75 * MaxRtrAdvInterval.
```
{{Default: 0.33 * MaxRtrAdvInterval}}
```

RQ_COR_8263 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a timer to set in seconds for controlling the minimum time in seconds between sending unsolicited multicast Router Advertisements from the interface. The timer is set by system management during configuration at a value other than the default value, no less than 3 seconds, and no more than 3/4 of the maximum interval value (cf MaxRtrAdvInterval).

Requirement: The implementation transmits unsolicited multicast Router Advertisements from the interface with periods whose minimum interval is the value set by system management during configuration.

RFC text: MinRtrAdvInterval
`{{The minimum time allowed between sending unsolicited multicast Router Advertisements from the interface, in seconds. MUST be no less than 3 seconds and no greater than .75}}` * MaxRtrAdvInterval.
Default: 0.33 * MaxRtrAdvInterval

RQ_COR_8264 Router Advertisement Config: AdvManagedFlag

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface a flag to indicate the value to be placed in the "Managed address configuration" flag field in the Router Advertisement.

RFC text: `{{AdvManagedFlag`
`The TRUE/FALSE value to be placed in the "Managed address configuration" flag field in the Router Advertisement.}}` See [ADDRCONF].
Default: FALSE

RQ_COR_8265 Router Advertisement Config: AdvManagedFlag

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to indicate the value to be placed in the "Managed address configuration" flag field in the Router Advertisement. The flag is not set during configuration by system management thereby remaining at its default value.

Requirement: The implementation places False in the "Managed address configuration" flag field when generating Router Advertisements.

RFC text:

```
{{AdvManagedFlag
The TRUE/FALSE value to be placed in the "Managed address
configuration" flag field in the Router Advertisement. See
[ADDRCONF].
Default: FALSE}}
```

RQ_COR_8266 Router Advertisement Config: AdvManagedFlag

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to indicate the value to be placed in the "Managed address configuration" flag field in the Router Advertisement. The flag is set to TRUE during configuration by system management.

Requirement: The implementation places True in the "Managed address configuration" flag field when generating Router Advertisements.

RFC text: `{{AdvManagedFlag`
`The TRUE/FALSE value to be placed in the "Managed address configuration" flag field in the Router Advertisement.}}` See [ADDRCONF].
Default: FALSE

RQ_COR_8267 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface a flag to indicate the value to be placed in the "Other stateful configuration" flag field in the Router Advertisement.

RFC text: `{{AdvOtherConfigFlag`
`The TRUE/FALSE value to be placed in the "Other stateful configuration" flag field in the Router Advertisement}}`. See [ADDRCONF].
Default: FALSE

RQ_COR_8268 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to indicate the value to be placed in the "Other stateful configuration" flag field in the Router Advertisement. The flag is not set during configuration by system management thereby remaining at its default value.

Requirement: The implementation places False in the "Other stateful configuration" flag field when generating Router Advertisements.

RFC text:

```
{{AdvOtherConfigFlag
The TRUE/FALSE value to be placed in the "Other stateful
configuration" flag field in the Router Advertisement. See
[ADDRCONF].
Default: FALSE}}
```

RQ_COR_8269 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to indicate the value to be placed in the "Other stateful configuration" flag field in the Router Advertisement. The flag is set to TRUE during configuration by system management.

Requirement: The implementation places True in the "Other stateful configuration" flag field when generating Router Advertisements.

RFC text: `{{AdvOtherConfigFlag`
`The TRUE/FALSE value to be placed in the "Other stateful configuration" flag field in the Router Advertisement}}`. See [ADDRCONF].
Default: FALSE

RQ_COR_8270 Router Advertisement Config: MTU Option

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface the value to set in the MTU field of the MTU options to be sent by the implementation.

RFC text: `{{AdvLinkMTU`
`The value to be placed in MTU options sent by the router}}`. A value of zero indicates that no MTU options are sent.
Default: 0

RQ_COR_8271 Router Advertisement Config: MTU Option

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the value to set in the MTU field of the MTU options to be sent by the implementation. The value is not set during configuration by system management thereby remaining at its default value.

Requirement: The implementation does not generate nor send MTU options.

RFC text:

```
{{AdvLinkMTU
The value to be placed in MTU options sent by the router. A value of
zero indicates that no MTU options are sent.
Default: 0}}
```

RQ_COR_8272 Router Advertisement Config: MTU Option

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the value to set in the MTU field of the MTU options to be sent by the implementation. The value is set during configuration by system management to a value other than its default of 0.

Requirement: The implementation generates and sends MTU options whose MTU field is set to the configuration management value.

RFC text: `{{AdvLinkMTU`
`The value to be placed in MTU options sent by the router.}}` A value of zero indicates that no MTU options are sent.
Default: 0

RQ_COR_8273 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface the time to be set in the Reachable Time field of Router Advertisment packets sent by the implementation. This time is no greater than 3,600,000 milliseconds (1 hour).

RFC text: `{{AdvReachableTime`
`The value to be placed in the Reachable Time field in the Router Advertisement messages sent by the router}}`. The value zero means unspecified (by this router). MUST be no greater than 3,600,000 milliseconds (1 hour)}}.
Default: 0

RQ_COR_8274 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the time to be set in the Reachable Time field of Router Advertisment packets sent by the implementation. This time is not set during configuration by system management thereby remaining at its default value of 0.

Requirement: The implementation sets the Reachable Time field to 0 in generated Router Advertisement messages indicating that its reachable time is unspecified.

RFC text: {{AdvReachableTime
The value to be placed in the Reachable Time field in the Router Advertisement messages sent by the router. The value zero means unspecified (by this router). MUST be no greater than 3,600,000 milliseconds (1 hour)}}.
Default: 0}}

RQ_COR_8275 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the time to be set in the Reachable Time of Router Advertisment packets sent by the implementation. This time is set during configuration by system management to a value other than 0.

Requirement: The implementation sets the Reachable Time field in Router Advertisement messages to the value set by system management during configuration.

RFC text: {{AdvReachableTime
The value to be placed in the Reachable Time field in the Router Advertisement messages sent by the router}}. The value zero means unspecified (by this router). MUST be no greater than 3,600,000 milliseconds (1 hour)}}.
Default: 0

RQ_COR_8276 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface the value to be placed in the Retrans Timer field of Router Advertisement messages sent by the implementation.

RFC text: {{AdvRetransTimer
The value to be placed in the Retrans Timer field in the Router Advertisement messages sent by the router}}. The value zero means unspecified (by this router).
Default: 0

RQ_COR_8277 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the value to be placed in the Retrans Timer field of Router Advertisement messages sent by the implementation. This time is not set during configuration by system management thereby remaining at its default value of 0.

Requirement: The implementation sets the Retrans Timer field to 0 in generated Router Advertisement messages indicating that the Retrans Timer is unspecified.

RFC text:
```
{{AdvRetransTimer
The value to be placed in the Retrans Timer field in the Router
Advertisement messages sent by the router. The value zero means
unspecified (by this router).
Default: 0}}
```

RQ_COR_8278 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the value to be placed in the Retrans Timer field of Router Advertisement messages sent by the implementation. This time is set to a value other than its default value of 0.

Requirement: The implementation sets the Retrans Timer field in generated Router Advertisement messages to the value set by system management during configuration.

RFC text:
```
{{AdvRetransTimer
The value to be placed in the Retrans Timer field in the Router
Advertisement messages sent by the router. The value zero means
unspecified (by this router).
Default: 0}}
```

RQ_COR_8279 Router Advertisement Config: AdvCurHopLimit

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface the value to be placed in the Cur Hop Limit field of Router Advertisement messages sent by the implementation.

RFC text: `{{AdvCurHopLimit`
`The default value to be placed in the Cur Hop Limit field in the`
`Router Advertisement messages sent by the router.}}` The value should be set to that current diameter of the Internet. The value zero means unspecified (by this router).
Default: The value specified in the "Assigned Numbers" RFC [ASSIGNED] that was in effect at the time of implementation.

RQ_COR_8280 Router Advertisement Config: AdvCurHopLimit

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* SHOULD *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the value to be placed in the Cur Hop Limit field of Router Advertisement messages sent by the implementation. System management sets the value for the Cur Hop Limit field.

Requirement: The value set by system management is the current diameter of the Internet.

RFC text: AdvCurHopLimit
The default value to be placed in the Cur Hop Limit field in the Router Advertisement messages sent by the router. `{{The value should be set to that current diameter of the Internet}}`. The value zero means unspecified (by this router).
Default: The value specified in the "Assigned Numbers" RFC [ASSIGNED] that was in effect at the time of implementation.

RQ_COR_8281 Router Advertisement Config: AdvCurHopLimit

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the value to be placed in the Cur Hop Limit field of Router Advertisement messages sent by the implementation. This value is set during configuration by system management to 0.

Requirement: The implementation sets the Cur Hop Limit field to 0 in generated Router Advertisement messages indicating that the hop limit is unspecified.

RFC text: AdvCurHopLimit
The default value to be placed in the Cur Hop Limit field in the Router Advertisement messages sent by the router. The value should be set to that current diameter of the Internet. `{{The value zero means unspecified (by this router).}}`
Default: The value specified in the "Assigned Numbers" RFC [ASSIGNED] that was in effect at the time of implementation.

RQ_COR_8282 Router Advertisement Config: AdvCurHopLimit

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the value to be placed in the Cur Hop Limit field of Router Advertisement messages sent by the implementation. This value is not set during configuration by system management thereby remaining at its default value as specified in the "Assigned Numbers" RFC [ASSIGNED] that was in effect at the time of implementation.

Requirement: The implementation sets the Cur Hop Limit field to the default value in generated Router Advertisement messages.

RFC text: AdvCurHopLimit
The default value to be placed in the Cur Hop Limit field in the Router Advertisement messages sent by the router. The value should be set to that current diameter of the Internet. The value zero means unspecified (by this router).
`{{Default: The value specified in the "Assigned Numbers" RFC [ASSIGNED] that was in effect at the time of implementation.}}`

RQ_COR_8283 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface the time in seconds to be placed in the Router Lifetime field of Router Advertisement messages sent by the implementation.

RFC text: AdvDefaultLifetime
`{{The value to be placed in the Router Lifetime field of Router Advertisements sent from the interface, in seconds}}`. MUST be either zero or between MaxRtrAdvInterval and 9000 seconds. A value of zero indicates that the router is not to be used as a default router.
Default: 3 * MaxRtrAdvInterval

RQ_COR_8284 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the time in seconds to be placed in the Router Lifetime field of Router Advertisement messages sent by the implementation. This value is not set during configuration by system management thereby remaining at its default value.

Requirement: The implementation sets the Router Lifetime field in generated Router Advertisement messages to the default value of 3 times the maximum router advertisement interval.

RFC text: AdvDefaultLifetime
`{{The value to be placed in the Router Lifetime field of Router Advertisements sent from the interface, in seconds}}`. MUST be either zero or between MaxRtrAdvInterval and 9000 seconds. A value of zero indicates that the router is not to be used as a default router.
`{{Default: 3 * MaxRtrAdvInterval}}`

RQ_COR_8285 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the time in seconds to be placed in the Router Lifetime field of Router Advertisement messages sent by the implementation. This value is not set during configuration by system management thereby remaining at its default value.

Requirement: The implementation sets the Router Lifetime field in generated Router Advertisement messages to the default value of 3 times the maximum router advertisement interval or 9000 seconds, whatever is smaller.

RFC text: AdvDefaultLifetime
`{{The value to be placed in the Router Lifetime field of Router Advertisements sent from the interface, in seconds}}`. `{{MUST be either zero or between MaxRtrAdvInterval and 9000 seconds. A value of zero indicates that the router is not to be used as a default router. Default: 3 * MaxRtrAdvInterval}}`

RQ_COR_8286 Router Advertisement Config:

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface the time in seconds to be placed in the Router Lifetime field of Router Advertisement messages sent by the implementation. This value is set during configuration by system management.

Requirement: The implementation sets the Router Lifetime field in generated Router Advertisement messages to a value between the maximum router advertisement interval and 9000 seconds.

RFC text: AdvDefaultLifetime
`{{The value to be placed in the Router Lifetime field of Router Advertisements sent from the interface, in seconds. MUST be either zero or between MaxRtrAdvInterval and 9000 seconds}}`. A value of zero indicates that the router is not to be used as a default router.
Default: 3 * MaxRtrAdvInterval

RQ_COR_8287 Router Advertisement: Lifetime field

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement field with the Router Lifetime field set to 0.

Requirement: The implementation does not use the router transmitting the advertisement as a default router.

RFC text: AdvDefaultLifetime
The value to be placed in the Router Lifetime field of Router Advertisements sent from the interface, in seconds. MUST be either zero or between MaxRtrAdvInterval and 9000 seconds. `{{A value of zero indicates that the router is not to be used as a default router}}`.
Default: 3 * MaxRtrAdvInterval

RQ_COR_8288 Router Advertisement: Prefix Option

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:*

Context: The implementation is being configured for operation.

Requirement: System management provides for each implementation's multicast interface a list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface. For each prefix in the list, system management can configure the value in seconds to be placed in the Valid Lifetime field and the value to be placed in the on-link flag ("L-bit") field in the Prefix Information option. For Address Autoconfiguration of each prefix in the list, system management can configure the value in seconds to be placed in the Preferred Lifetime and the value to be placed in the Autonomous Flag field in the Prefix Information option.

RFC text: AdvPrefixList
`{{A list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface}}`.
Default: all prefixes that the router advertises via routing protocols as being on-link for the interface from which the advertisement is sent. The link-local prefix SHOULD NOT be included in the list of advertised prefixes.
`{{Each prefix has an associated:`
`AdvValidLifetime`
`The value to be placed in the Valid Lifetime in the Prefix Information option, in seconds}}`. The designated value of all 1's (0xffffffff) represents infinity. Implementations MUST allow AdvValidLifetime to be specified in two ways:
- a time that decrements in real time, that is, one that will result in a Lifetime of zero at the specified time in the future, or
- a fixed time that stays the same in consecutive advertisements.

Default: 2592000 seconds (30 days), fixed (i.e., stays the same in consecutive advertisements).
`{{AdvOnLinkFlag`
`The value to be placed in the on-link flag ("L-bit") field in the Prefix Information option}}`.
Default: TRUE
`{{Automatic address configuration [ADDRCONF] defines additional information associated with each the prefixes:`
`AdvPreferredLifetime`
`The value to be placed in the Preferred Lifetime in the Prefix Information option, in seconds}}`. The designated value of all 1's (0xffffffff) represents infinity. See [ADDRCONF] for details on how this value is used. Implementations MUST allow AdvPreferredLifetime to be specified in two ways:
- a time that decrements in real time, that is, one that will result in a Lifetime of zero at a specified time in the future, or
- a fixed time that stays the same in consecutive advertisements.

Default: 604800 seconds (7 days), fixed (i.e., stays the same in consecutive advertisements).
`{{AdvAutonomousFlag`
`The value to be placed in the Autonomous Flag field in the Prefix Information option}}`. See [ADDRCONF].
Default: TRUE

RQ_COR_8289 Router Advertisement: Prefix Option

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:*

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface. For each prefix in the list, system management can configure the value in seconds to be placed in the Valid Lifetime field and the value to be placed in the on-link flag ("L-bit") field in the Prefix Information option. For Address Autoconfiguration of each prefix in the list, system management can configure the value in seconds to be placed in the Preferred Lifetime and the value to be placed in the Autonomous Flag field in the Prefix Information option. System management does provide input to change the default values for the prefixes and their information to be placed in Prefix Information options of the advertisements.

Requirement: The implementation places all prefixes that it advertises via routing protocos as on-link for the interface in the Prefix Information options. For each prefix, the implementation sets the Valid Lifetime field to the default value of 2,592,000 seconds (30 days)-fixed, the L-bit field to TRUE, Preferred Lifetime field to 604,800 seconds (7 days)-fixed, and the Autonomous Flag field to TRUE.

RFC text: AdvPrefixList
A list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface.
`{{Default: all prefixes that the router advertises via routing protocols as being on-link for the interface from which the advertisement is sent.}}` The link-local prefix SHOULD NOT be included in the list of advertised prefixes.
Each prefix has an associated:
AdvValidLifetime
The value to be placed in the Valid Lifetime in the Prefix Information option, in seconds. The designated value of all 1's (0xffffffff) represents infinity. Implementations MUST allow AdvValidLifetime to be specified in two ways:
- a time that decrements in real time, that is, one that will result in a Lifetime of zero at the specified time in the future, or
- a fixed time that stays the same in consecutive advertisements.
`{{Default: 2592000 seconds (30 days), fixed (i.e., stays the same in consecutive advertisements)}}`.
AdvOnLinkFlag
The value to be placed in the on-link flag ("L-bit") field in the Prefix Information option.
Default: TRUE
Automatic address configuration [ADDRCONF] defines additional information associated with each the prefixes:
AdvPreferredLifetime
The value to be placed in the Preferred Lifetime in the Prefix Information option, in seconds. The designated value of all 1's (0xffffffff) represents infinity. See [ADDRCONF] for details on how this value is used. Implementations MUST allow AdvPreferredLifetime to be specified in two ways:
- a time that decrements in real time, that is, one that will result in a Lifetime of zero at a specified time in the future, or
- a fixed time that stays the same in consecutive advertisements.
`{{Default: 604800 seconds (7 days), fixed (i.e., stays the same in consecutive advertisements)}}`.
AdvAutonomousFlag
The value to be placed in the Autonomous Flag field in the Prefix Information option. See [ADDRCONF].
`{{Default: TRUE}}`

RQ_COR_8290 Router Advertisement: Prefix Option

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* SHOULD *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface.

Requirement: The implementation does not place a link-local prefix in the list of advertised prefixes.

RFC text: AdvPrefixList
A list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface.
Default: all prefixes that the router advertises via routing protocols as being on-link for the interface from which the advertisement is sent. `{{The link-local prefix SHOULD NOT be included in the list of advertised prefixes}}`.
Each prefix has an associated:
AdvValidLifetime
The value to be placed in the Valid Lifetime in the Prefix Information option, in seconds. The designated value of all 1's (0xffffffff) represents infinity. Implementations MUST allow AdvValidLifetime to be specified in two ways:
- a time that decrements in real time, that is, one that will result in a Lifetime of zero at the specified time in the future, or
- a fixed time that stays the same in consecutive advertisements.
Default: 2592000 seconds (30 days), fixed (i.e., stays the same in consecutive advertisements).
AdvOnLinkFlag
The value to be placed in the on-link flag ("L-bit") field in the Prefix Information option.
Default: TRUE
Automatic address configuration [ADDRCONF] defines additional information associated with each the prefixes:
AdvPreferredLifetime
The value to be placed in the Preferred Lifetime in the Prefix Information option, in seconds. The designated value of all 1's (0xffffffff) represents infinity. See [ADDRCONF] for details on how this value is used. Implementations MUST allow AdvPreferredLifetime to be specified in two ways:
- a time that decrements in real time, that is, one that will result in a Lifetime of zero at a specified time in the future, or
- a fixed time that stays the same in consecutive advertisements.
Default: 604800 seconds (7 days), fixed (i.e., stays the same in consecutive advertisements).
AdvAutonomousFlag
The value to be placed in the Autonomous Flag field in the Prefix Information option. See [ADDRCONF].
Default: TRUE

RQ_COR_8291 Router Advertisement: Prefix Option

RFC 2461 *Clause:* 6.2.1 *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface.

Requirement: The implementation configures the value to be placed in the Valid Lifetime in the Prefix Information option as a time that decrements in real time.

RFC text: AdvPreferredLifetime
The value to be placed in the Preferred Lifetime in the Prefix Information option, in seconds. The designated value of all 1's (0xffffffff) represents infinity. See [ADDRCONF] for details on how this value is used. `{{Implementations MUST allow AdvPreferredLifetime to be specified in two ways:`
`- a time that decrements in real time, that is, one that will result in a Lifetime of zero at a specified time in the future,}}` or
- a fixed time that stays the same in consecutive advertisements.
Default: 604800 seconds (7 days), fixed (i.e., stays the same in consecutive advertisements).

RQ_COR_8292 Router Advertisement: Prefix Option

RFC 2461 *Clause:* 6.2.1 *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a list of prefixes to be placed in Prefix Information options in Router Advertisement messages sent from the interface.

Requirement: The implementation configures the value to be placed in the Preferred Lifetime in the Prefix Information option as a fixed time that stays the same in consecutive advertisements.

RFC text: AdvValidLifetime
The value to be placed in the Valid Lifetime in the Prefix Information option, in seconds. The designated value of all 1's (0xffffffff) represents infinity. {{Implementations MUST allow AdvValidLifetime to be specified in two ways:}}
- a time that decrements in real time, that is, one that will result in a Lifetime of zero at the specified time in the future, or
{{ - a fixed time that stays the same in consecutive advertisements}}.}}
Default: 2592000 seconds (30 days), fixed (i.e., stays the same in consecutive advertisements).

RQ_COR_8293 Router Advertisement Behavior

RFC 2461 *Clause:* 6.2.1 ¶4 *Type:* MUST *applies to:* Router

Context: Hosts correctly implement CurHopLimit, RetransTimer, and ReachableTime.

Requirement: The implementation uses CurHopLimit, RetransTimer, and ReachableTime is the same manner as the hosts.

RFC text: The above variables contain information that is placed in outgoing Router Advertisement messages. Hosts use the received information to initialize a set of analogous variables that control their external behavior (see section 6.3.2). {{Some of these host variables (e.g., CurHopLimit, RetransTimer, and ReachableTime) apply to all nodes including routers. In practice, these variables may not actually be present on routers, since their contents can be derived from the variables described above. However, external router behavior MUST be the same as host behavior with respect to these variables. In particular, this includes the occasional randomization of the ReachableTime value as described in section 6.3.2.}}

RQ_COR_8294 Router Advertisement Behavior at Startup

RFC 2461 *Clause:* 6.2.2 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation has a functioning and enabled multicast interface having at least one assigned unicast IP address. The system has not been configured by system management to send periodic Router Advertisements and to respond to Router Solicitations.

Requirement: The implementation does not send Router advertisements out the functioning and enabled multicast interface.

RFC text: {{The term "advertising interface" refers to any functioning and enabled multicast interface that has at least one unicast IP address assigned to it and whose corresponding AdvSendAdvertisements flag is TRUE. {{A router MUST NOT send Router Advertisements out any interface that is not an advertising interface.}}}}

RQ_COR_8295 Router Advertisement [Generate]

RFC 2461 *Clause:* 6.2.2 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation has a functioning and enabled multicast interface. The system has been configured by system management to send periodic Router Advertisements and to respond to Router Solicitations. The multicast interface does not have at least one assigned unicast IP address.

Requirement: The implementation does not send Router advertisements out the functioning and enabled multicast interface.

RFC text: `{{The term "advertising interface" refers to any functioning and enabled multicast interface that has at least one unicast IP address assigned to it and whose corresponding AdvSendAdvertisements flag is TRUE. {{A router MUST NOT send Router Advertisements out any interface that is not an advertising interface.}}}}`

RQ_COR_8296 Router Advertisement Behavior on

RFC 2461 *Clause:* 6.2.2 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation is configured to not be an advertising interface and is operating correctly.

Requirement: On being configured by system management to send Router Advertisements and respond to Router Solicitations, the system becomes an advertising interface.

RFC text: An interface may become an advertising interface at times other than system startup. For example:
`{{- changing the AdvSendAdvertisements flag on an enabled interface from FALSE to TRUE}}`, or
- administratively enabling the interface, if it had been administratively disabled, and its AdvSendAdvertisements flag is TRUE, or
- enabling IP forwarding capability (i.e., changing the system from being a host to being a router), when the interface's AdvSendAdvertisements flag is TRUE.

RQ_COR_8297 Router Advertisement Behavior at Startup

RFC 2461 *Clause:* 6.2.2 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation is configured to be an advertising interface but that interface has been administrative disabled.

Requirement: On re-enabling of the interface and still being configured by system management to send Router Advertisements and respond to Router Solicitations, the system returns to being an advertising interface.

RFC text: An interface may become an advertising interface at times other than system startup. For example:
- changing the AdvSendAdvertisements flag on an enabled interface from FALSE to TRUE, or
`{{- administratively enabling the interface, if it had been administratively disabled, and its AdvSendAdvertisements flag is TRUE}}`, or
- enabling IP forwarding capability (i.e., changing the system from being a host to being a router), when the interface's AdvSendAdvertisements flag is TRUE.

RQ_COR_8298

RFC 2461 *Clause:* 6.2.2 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation is operating correctly as a host. System management changes the implementation from being a host to router and the configures the implementation to send Router Advertisements and to respond to Router Solicitations (cf AdvSendAdvertisements).

Requirement: The implementation becomes an advertising interface.

RFC text: An interface may become an advertising interface at times other than system startup. For example:
- changing the AdvSendAdvertisements flag on an enabled interface from FALSE to TRUE, or
- administratively enabling the interface, if it had been administratively disabled, and its AdvSendAdvertisements flag is TRUE, or
`{{- enabling IP forwarding capability (i.e., changing the system from being a host to being a router), when the interface's AdvSendAdvertisements flag is TRUE.}}`

RQ_COR_8299 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.2 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation receives a valid Router Solicitation on an advertising interface with the all-routers multicast address in IP Destination Address field.

Requirement: The implementation responds to the Router Solicitation sent on the all-routers multicast address.

RFC text: `{{A router MUST join the all-routers multicast address on an advertising interface. Routers respond to Router Solicitations sent to the all-routers address}}` and verify the consistency of Router Advertisements sent by neighboring routers.

RQ_COR_8300 Router Advertisement [Process]

RFC 2461 *Clause:* 6.2.2 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation receives a valid Router Advertisement on an advertising interface with the all-routers multicast address in IP Destination Address field sent by neighboring routers.

Requirement: The implementation verifies the Router Advertisement sent on the all-routers multicast address.

RFC text: `{{A router MUST join the all-routers multicast address on an advertising interface. Routers respond to Router Solicitations sent to the all-routers address}}` and verify the consistency of Router Advertisements sent by neighboring routers.

RQ_COR_8301 Router Advertisement Options [Form]

RFC 2461 *Clause:* 6.2.3 ¶1 "In the *Type:* MUST *applies to:* Router

Context: The implementation is generating a Router Advertisement and wants to facilitate in-bound load balancing over replicated interfaces.

Requirement: The implementation omits the Source Link-layer Address option in the advertisement.

RFC text: - In the options:
o Source Link-Layer Address option: link-layer address of the sending interface. `{{This option MAY be omitted to facilitate in-bound load balancing over replicated interfaces.}}`

RQ_COR_8302 Router Advertisement Prefix Option

RFC 2461 *Clause:* 6.2.3 ¶1 "In the *Type:* MUST *applies to:* Router

Context: The implementation is generating a Router Advertisement.

Requirement: The implementation generates one Prefix Information for each prefix provided by system management during configuration (cf AdvPrefixList).

RFC text: Prefix Information options: `{{one Prefix Information option for each prefix listed in AdvPrefixList}}` with the option fields set from the information in the AdvPrefixList entry as follows:

RQ_COR_8303 Router Advertisement Behavior at Startup

RFC 2461 *Clause:* 6.2.3 ¶2 *Type:* MUST *applies to:* Router

Context: System configures the implementation to send Router Advertisements out an interface without advertising itself as a default router.

Requirement: The implementation generates the advertisement with the Router Lifetime field set to zero.

RFC text: A router might want to send Router Advertisements without advertising itself as a default router. For instance, a router might advertise prefixes for address autoconfiguration while not wishing to forward packets. `{{Such a router sets the Router Lifetime field in outgoing advertisements to zero.}}`

RQ_COR_8304 Router Advertisement Options [Form]

RFC 2461 *Clause:* 6.2.3 ¶3 *Type:* MAY *applies to:* Router

Context: The implementation is generating unsolicited Router Advertisements.

Requirement: The implementation does not include some or all options when sending the unsolicited advertisements.

RFC text: `{{A router MAY choose not to include some or all options when sending unsolicited Router Advertisements}}`. For example, if prefix lifetimes are much longer than AdvDefaultLifetime, including them every few advertisements may be sufficient. However, when responding to a Router Solicitation or while sending the first few initial unsolicited advertisements, a router SHOULD include all options so that all information (e.g., prefixes) is propagated quickly during system initialization.

RQ_COR_8305 Router Advertisement Options [Form]

RFC 2461 *Clause:* 6.2.3 ¶3 *Type:* MAY *applies to:* Router

Context: The implementation is generating the first few unsolicited Router Advertisements during system initialization.

Requirement: The implementation includes all options when sending the unsolicited advertisements.

RFC text: A router MAY choose not to include some or all options when sending unsolicited Router Advertisements. For example, if prefix lifetimes are much longer than AdvDefaultLifetime, including them every few advertisements may be sufficient. However, when responding to a Router Solicitation or `{{while sending the first few initial unsolicited advertisements, a router SHOULD include all options so that all information (e.g., prefixes) is propagated quickly during system initialization}}`.

RQ_COR_8306 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.3 ¶3 *Type:* SHOULD *applies to:* Router

Context: The implementation has received a valid Router Solicitation.

Requirement: The implementation transmits a Router Advertisement that includes all options.

RFC text: A router MAY choose not to include some or all options when sending unsolicited Router Advertisements. For example, if prefix lifetimes are much longer than AdvDefaultLifetime, including them every few advertisements may be sufficient. However, `{{when responding to a Router Solicitation}}` or while sending the first few initial unsolicited advertisements, `{{a router SHOULD include all options so that all information (e.g., prefixes) is propagated quickly during system initialization}}`.

RQ_COR_8307 Router Advertisement Options [Form]

RFC 2461 *Clause:* 6.2.3 ¶4 *Type:* SHOULD *applies to:* Router

Context: The implementation is generating a Router Advertisement that includes all options. The size of the advertisement exceeds the link MTU.

Requirement: The implementation transmits multiple advertisements each containing a subset of the options needed for the advertisement that was too large.

RFC text: `{{If including all options causes the size of an advertisement to exceed the link MTU, multiple advertisements can be sent, each containing a subset of the options}}`.

RQ_COR_8308 Router Advertisement [Generate]

RFC 2461 *Clause:* 6.2.4 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is operating.

Requirement: The implementation does not send Router Advertisement packets.

RFC text: `{{A host MUST NOT send Router Advertisement messages at any time}}`.

RQ_COR_8309 Router Advertisement Behavior at Startup

RFC 2461 *Clause:* 6.2.4 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation is generating unsolicited Router Advertisement packets on an advertising interface. The implementation has already sent more than the protocol constant MAX_INITIAL_RTR_ADVERTISEMENTS of unsolicited advertisements.

Requirement: The intervals between successive transmissions of unsolicited advertisements are random values uniformly-distributed between the minimum and maximum router advertisement interval.

RFC text: Unsolicited Router Advertisements are not strictly periodic: the interval between subsequent transmissions is randomized to reduce the probability of synchronization with the advertisements from other routers on the same link [SYNC]. Each advertising interface has its own timer. `{{Whenever a multicast advertisement is sent from an interface, the timer is reset to a uniformly-distributed random value between the interface's configured MinRtrAdvInterval and MaxRtrAdvInterval; expiration of the timer causes the next advertisement to be sent and a new random value to be chosen}}`.

RQ_COR_8310 Router Advertisement Behavior at Startup

RFC 2461 *Clause:* 6.2.4 ¶3 *Type:* SHOULD *applies to:* Router

Context: The implementation has just become an advertising interface.

Requirement: For the first advertisements (the protocol constant MAX_INITIAL_RTR_ADVERTISEMENTS), the random interval of unsolicited Router Advertisements is less then or equal to the protocol constant MAX_INITIAL_RTR_ADVERT_INTERVAL.

RFC text: {{For the first few advertisements (up to MAX_INITIAL_RTR_ADVERTISEMENTS) sent from an interface when it becomes an advertising interface, if the randomly chosen interval is greater than MAX_INITIAL_RTR_ADVERT_INTERVAL, the timer SHOULD be set to MAX_INITIAL_RTR_ADVERT_INTERVAL instead}}. Using a smaller interval for the initial advertisements increases the likelihood of a router being discovered quickly when it first becomes available, in the presence of possible packet loss.

RQ_COR_8311 Router Advertisement Behavior on

RFC 2461 *Clause:* 6.2.4 ¶4 *Type:* MAY *applies to:* Router

Context: The implementation is functioning normally and sending Router Advertisements. Then system management changes information contained in the advertisements.

Requirement: The implementation sends up to the protocol constant MAX_INITIAL_RTR_ADVERTISEMENTS unsolicited advertisements using the same rules when the interface became an advertising interface.

RFC text: {{The information contained in Router Advertisements may change through actions of system management}}. For instance, the lifetime of advertised prefixes may change, new prefixes could be added, a router could cease to be a router (i.e., switch from being a router to being a host), etc. {{In such cases, the router MAY transmit up to MAX_INITIAL_RTR_ADVERTISEMENTS unsolicited advertisements, using the same rules as when an interface becomes an advertising interface}}.

RQ_COR_8312 Router Advertisement Behavior on

RFC 2461 *Clause:* 6.2.5 ¶2 *Type:* SHOULD *applies to:* Router

Context: System management causes an interface of the implementation to cease being an advertising interface.

Requirement: The implementation transmits not more than the protocol constant MAX_FINAL_RTR_ADVERTISEMENTS multicast Router Advertisements on the interface with a Router Lifetime field of zero.

RFC text: {{An interface may cease to be an advertising interface, through actions of system management such as:
- changing the AdvSendAdvertisements flag of an enabled interface from TRUE to FALSE, or
- administratively disabling the interface, or
- shutting down the system.
In such cases the router SHOULD transmit one or more (but not more than MAX_FINAL_RTR_ADVERTISEMENTS) final multicast Router Advertisements on the interface with a Router Lifetime field of zero}}. In the case of a router becoming a host, the system SHOULD also depart from the all-routers IP multicast group on all interfaces on which the router supports IP multicast (whether or not they had been advertising interfaces). In addition, the host MUST insure that subsequent Neighbor Advertisement messages sent from the interface have the Router flag set to zero.

RQ_COR_8313 address: [Configure]

RFC 2461 *Clause:* 6.2.5 ¶2 *Type:* SHOULD *applies to:* Router

Context: System management has just caused an interface of the implementation to cease being an advertising interface and the implementation to become a host. The implementation has transmitted not more than the protocol constant MAX_FINAL_RTR_ADVERTISEMENTS multicast Router Advertisements on the interface with a Router Lifetime field of zero.

Requirement: The implementation departs from the all-routers IP multicast group on all interfaces on which the router supports IP multicast (whether or not they had been advertising interfaces).

RFC text: <An interface may cease to be an advertising interface, through actions of system management such as:
- changing the AdvSendAdvertisements flag of an enabled interface from TRUE to FALSE, or
- administratively disabling the interface, or
- shutting down the system.

In such cases the router SHOULD transmit one or more (but not more than MAX_FINAL_RTR_ADVERTISEMENTS) final multicast Router Advertisements on the interface with a Router Lifetime field of zero. `{{In the case of a router becoming a host, the system SHOULD also depart from the all-routers IP multicast group on all interfaces on which the router supports IP multicast (whether or not they had been advertising interfaces)}}`. In addition, the host MUST insure that subsequent Neighbor Advertisement messages sent from the interface have the Router flag set to zero.

RQ_COR_8314 Neighbor Advertisement: Unsolicited NA

RFC 2461 *Clause:* 6.2.5 ¶2 *Type:* MUST *applies to:* Router

Context: System management has just caused an interface of the implementation to cease being an advertising interface and the implementation to become a host. The implementation has transmitted not more than the protocol constant MAX_FINAL_RTR_ADVERTISEMENTS multicast Router Advertisements on the interface with a Router Lifetime field of zero. The implementation departs from the all-routers IP multicast group on all interfaces on which the router supports IP multicast (whether or not they had been advertising interfaces).

Requirement: The implementation, now as host, transmits Neighbor Advertisements from the multicast interface with the Router flag set to 0.

RFC text: <An interface may cease to be an advertising interface, through actions of system management such as:
- changing the AdvSendAdvertisements flag of an enabled interface from TRUE to FALSE, or
- administratively disabling the interface, or
- shutting down the system.

In such cases the router SHOULD transmit one or more (but not more than MAX_FINAL_RTR_ADVERTISEMENTS) final multicast Router Advertisements on the interface with a Router Lifetime field of zero. `{{In the case of a router becoming a host, the system SHOULD also depart from the all-routers IP multicast group on all interfaces on which the router supports IP multicast (whether or not they had been advertising interfaces)}}`. In addition, the host MUST insure that subsequent Neighbor Advertisement messages sent from the interface have the Router flag set to zero.

RQ_COR_8315 Router Advertisement Behavior on

RFC 2461 *Clause:* 6.2.5 ¶3 *Type:* MUST *applies to:* Router

Context: System management has disabled the implementation's IP forwarding capability but continues operation of the advertising interfaces.

Requirement: The implementation, now as a host, sends subsequent Router Advertisements with the Router Lifetime field set to zero.

RFC text: `{{Note that system management may disable a router's IP forwarding capability (i.e., changing the system from being a router to being a host), a step that does not necessarily imply that the router's interfaces stop being advertising interfaces. In such cases, subsequent Router Advertisements MUST set the Router Lifetime field to zero}}`.

RQ_COR_8316 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Router Solicitation.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A host MUST silently discard any received Router Solicitation messages.}}`

RQ_COR_8317 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶2 *Type:* MAY *applies to:* Router

Context: The implementation has received a valid Router Solicitation on an advertising interface. The solicitation's source address is not the Unspecified Address.

Requirement: The implementation sends a Router Advertisement setting the IP Header Destination Address to the soliciting host's address.

RFC text: In addition to sending periodic, unsolicited advertisements, `{{a router sends advertisements in response to valid solicitations received on an advertising interface. A router MAY choose to unicast the response directly to the soliciting host's address (if the solicitation's source address is not the unspecified address)}}`, but the usual case is to multicast the response to the all-nodes group. In the latter case, the interface's interval timer is reset to a new random value, as if an unsolicited advertisement had just been sent (see section 6.2.4).

RQ_COR_8318 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶2 *Type:* SHOULD *applies to:* Router

Context: The implementation has received a valid Router Solicitation on an advertising interface.

Requirement: The implementation sends a Router Advertisement setting the IP Header Destination Address to the all-nodes multicast address and resets the advertisement interval timer to a new random value as if an unsolicited advertisement had just been sent.

RFC text: In addition to sending periodic, unsolicited advertisements, `{{a router sends advertisements in response to valid solicitations received on an advertising interface}}`. A router MAY choose to unicast the response directly to the soliciting host's address (if the solicitation's source address is not the unspecified address), but `{{the usual case is to multicast the response to the all-nodes group. In the latter case, the interface's interval timer is reset to a new random value, as if an unsolicited advertisement had just been sent}}` (see section 6.2.4).

RQ_COR_8319 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation has received a valid Router Solicitation.

Requirement: The implementation delays sending the Router Advertisement responding to the solicitation by a random time between 0 and the protocol constant MAX_RA_DELAY_TIME seconds.

RFC text: `{{In all cases, Router Advertisements sent in response to a Router Solicitation MUST be delayed by a random time between 0 and MAX_RA_DELAY_TIME seconds}}`. (If a single advertisement is sent in response to multiple solicitations, the delay is relative to the first solicitation.) In addition, consecutive Router Advertisements sent to the all-nodes multicast address MUST be rate limited to no more than one advertisement every MIN_DELAY_BETWEEN_RAS seconds.

RQ_COR_8320 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation has received multiple valid Router Solicitations to which it has yet to respond with a Router Advertisement.

Requirement: The implementation delays sending the Router Advertisement responding to the solicitations by a random time between 0 and the protocol constant MAX_RA_DELAY_TIME seconds relative to the first solicitation.

RFC text: `{{In all cases, Router Advertisements sent in response to a Router Solicitation MUST be delayed by a random time between 0 and MAX_RA_DELAY_TIME seconds. (If a single advertisement is sent in response to multiple solicitations, the delay is relative to the first solicitation.}})` In addition, consecutive Router Advertisements sent to the all-nodes multicast address MUST be rate limited to no more than one advertisement every MIN_DELAY_BETWEEN_RAS seconds.

RQ_COR_8321 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation has received multiple valid Router Solicitations.

Requirement: In response to the solicitations, the implementation limits consecutive Router Advertisements sent to the all-nodes multicast address to no more than one advertisement every protocol constant MIN_DELAY_BETWEEN_RAS seconds.

RFC text: In all cases, Router Advertisements sent in response to a Router Solicitation MUST be delayed by a random time between 0 and MAX_RA_DELAY_TIME seconds. (If a single advertisement is sent in response to multiple solicitations, the delay is relative to the first solicitation.) `{{In addition, consecutive Router Advertisements sent to the all-nodes multicast address MUST be rate limited to no more than one advertisement every MIN_DELAY_BETWEEN_RAS seconds}}`.

RQ_COR_8322 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶4 *Type:* MAY *applies to:* Router

Context: The implementation has received a Router Solicitation and computed a random value between 0 and MAX_RA_DELAY_TIME. The computed value is a time later than the next multicast Router Advertisement is scheduled for sending.

Requirement: The implementation ignores the computed random delay and sends the advertisement at the already-scheduled time.

RFC text: `{{A router might process Router Solicitations as follows:`
`- Upon receipt of a Router Solicitation, compute a random delay within the range 0 through MAX_RA_DELAY_TIME. If the computed value corresponds to a time later than the time the next multicast Router Advertisement is scheduled to be sent, ignore the random delay and send the advertisement at the already-scheduled time.}}`
- If the router sent a multicast Router Advertisement (solicited or unsolicited) within the last MIN_DELAY_BETWEEN_RAS seconds, schedule the advertisement to be sent at a time corresponding to MIN_DELAY_BETWEEN_RAS plus the random value after the previous advertisement was sent. This ensures that the multicast Router Advertisements are rate limited.
- Otherwise, schedule the sending of a Router Advertisement at the time given by the random value.

RQ_COR_8323 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶4 *Type:* MAY *applies to:* Router

Context: The implementation has received a valid Router Solicitation. The implementation has already sent a solicited or unsolicited Router Advertisement with the last MIN_DELAY_BETWEEN_RAS seconds.

Requirement: The implementation sends the Router Advertisement in response to the solicitation at a time corresponding to MIN_DELAY_BETWEEN_RAS plus a random value between 0 and MAX_RA_DELAY_TIME after the last solicitation was sent.

RFC text: `{{A router might process Router Solicitations as follows:}}`
- Upon receipt of a Router Solicitation, compute a random delay within the range 0 through MAX_RA_DELAY_TIME. If the computed value corresponds to a time later than the time the next multicast Router Advertisement is scheduled to be sent, ignore the random delay and send the advertisement at the already-scheduled time.
`{{- If the router sent a multicast Router Advertisement (solicited or unsolicited) within the last MIN_DELAY_BETWEEN_RAS seconds, schedule the advertisement to be sent at a time corresponding to MIN_DELAY_BETWEEN_RAS plus the random value after the previous advertisement was sent.}}` This ensures that the multicast Router Advertisements are rate limited.
- Otherwise, schedule the sending of a Router Advertisement at the time given by the random value.

RQ_COR_8324 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶4 *Type:* MAY *applies to:* Router

Context: The implementation has received a Router Solicitation and computed a random value between 0 and MAX_RA_DELAY_TIME.

Requirement: The implementation sends a Router Advertisement in response to the solicitation at the computed random value time in seconds.

RFC text: `{{A router might process Router Solicitations as follows: }}`
- Upon receipt of a Router Solicitation, compute a random delay within the range 0 through MAX_RA_DELAY_TIME. If the computed value corresponds to a time later than the time the next multicast Router Advertisement is scheduled to be sent, ignore the random delay and send the advertisement at the already-scheduled time.
- If the router sent a multicast Router Advertisement (solicited or unsolicited) within the last MIN_DELAY_BETWEEN_RAS seconds, schedule the advertisement to be sent at a time corresponding to MIN_DELAY_BETWEEN_RAS plus the random value after the previous advertisement was sent. This ensures that the multicast Router Advertisements are rate limited.
- Otherwise, `{{schedule the sending of a Router Advertisement at the time given by the random value.}}`

RQ_COR_8325 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶5 *Type:* MAY *applies to:* Router

Context: The implementation has received Router Solicitations during several contiguous minimim router advertisement intervals (cf MinRtrAdvInterval).

Requirement: The implementation transmits Router Advertisments more frequently than the number of minimum router advertisement intervals.

RFC text: `{{Note that a router is permitted to send multicast Router Advertisements more frequently than indicated by the MinRtrAdvInterval configuration variable so long as the more frequent advertisements are responses to Router Solicitations}}`. In all cases, however, unsolicited multicast advertisements MUST NOT be sent more frequently than indicated by MinRtrAdvInterval.

RQ_COR_8326 Router Advertisement Behavior at Startup

RFC 2461 *Clause:* 6.2.6 ¶5 *Type:* MUST *applies to:* Router

Context: The implementation is transmitting unsolicited multicast Router Advertisments over 'n' continuous router advertisement intervals.

Requirement: The implementation SENDS no more than 'n' unsolicited advertisements over the 'n' intervals at one advertisement per interval.

RFC text: Note that a router is permitted to send multicast Router Advertisements more frequently than indicated by the MinRtrAdvInterval configuration variable so long as the more frequent advertisements are responses to Router Solicitations. `{{In all cases, however, unsolicited multicast advertisements MUST NOT be sent more frequently than indicated by MinRtrAdvInterval}}`.

RQ_COR_8327 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶6 *Type:* MUST *applies to:* Router

Context: The implementation receives a Router Solicitation from a new neighbor. The IP Source Address of the solicitation is the Unspecified Address.

Requirement: The implementation does not recognize the neighbor as being on the link.

RFC text: `{{Router Solicitations in which the Source Address is the unspecified address MUST NOT update the router's Neighbor Cache}}`; solicitations with a proper source address update the Neighbor Cache as follows. If the router already has a Neighbor Cache entry for the solicitation's sender, the solicitation contains a Source Link-Layer Address option, and the received link-layer address differs from that already in the cache, the link-layer address SHOULD be updated in the appropriate Neighbor Cache entry, and its reachability state MUST also be set to STALE. If there is no existing Neighbor Cache entry for the solicitation's sender, the router creates one, installs the link- layer address and sets its reachability state to STALE as specified in section 7.3.3. Whether or not a Source Link-Layer Address option is provided, if a Neighbor Cache entry for the solicitation's sender exists (or is created) the entry's IsRouter flag MUST be set to FALSE.

RQ_COR_8328 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶6 *Type:* SHOULD *applies to:* Router

Context: The implementation receives a valid Router Solicitation from a known neighbor having an IP unicast Source Address and with a Source Link-layer Address option indicating a link-layer address different from the current link-layer address.

Requirement: The implementation uses the new link-layer address for subsequent IP communication. The implementation makes no attempt to verify the reachability of the known neighbor until traffic is sent to the neighbor. The implementation also treats that the known neighbor as a host and not as a router.

RFC text: Router Solicitations in which the Source Address is the unspecified address MUST NOT update the router's Neighbor Cache; solicitations with a proper source address update the Neighbor Cache as follows. `{{If the router already has a Neighbor Cache entry for the solicitation's sender, the solicitation contains a Source Link-Layer Address option, and the received link-layer address differs from that already in the cache, the link-layer address SHOULD be updated in the appropriate Neighbor Cache entry, and its reachability state MUST also be set to STALE}}`. If there is no existing Neighbor Cache entry for the solicitation's sender, the router creates one, installs the link- layer address and sets its reachability state to STALE as specified in section 7.3.3. `{{Whether or not a Source Link-Layer Address option is provided, if a Neighbor Cache entry for the solicitation's sender exists (or is created) the entry's IsRouter flag MUST be set to FALSE}}`.

RQ_COR_8329 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶6 *Type:* MUST *applies to:* Router

Context: The implementation receives a valid Router Solicitation from a new neighbor having an IP unicast Source Address and containing a Source Link-layer Address option with the neighbor's link-layer address.

Requirement: The implementation uses the new link-layer address for subsequent IP communication. The implementation makes no attempt to verify the reachability of the new neighbor until traffic is sent to the neighbor. The implementation also treats that the known neighbor as a host and not as a router.

RFC text: Router Solicitations in which the Source Address is the unspecified address MUST NOT update the router's Neighbor Cache; solicitations with a proper source address update the Neighbor Cache as follows. If the router already has a Neighbor Cache entry for the solicitation's sender, the solicitation contains a Source Link-Layer Address option, and the received link-layer address differs from that already in the cache, the link-layer address SHOULD be updated in the appropriate Neighbor Cache entry, and its reachability state MUST also be set to STALE. `{{If there is no existing Neighbor Cache entry for the solicitation's sender, the router creates one, installs the link- layer address and sets its reachability state to STALE as specified in section 7.3.3.}}` Whether or not a Source Link-Layer Address option is provided, if a Neighbor Cache entry for the solicitation's sender exists (or is created) the entry's IsRouter flag MUST be set to FALSE.

RQ_COR_8330 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.6 ¶6 *Type:* MUST *applies to:* Router

Context: The implementation receives a valid Router Solicitation from a known router-neighbor having an IP unicast Source Address. There is no Source Link-layer Address option in the solicitation.

Requirement: The implementation now treats the neighbor as a host and no longer as a router.

RFC text: Router Solicitations in which the Source Address is the unspecified address MUST NOT update the router's Neighbor Cache; solicitations with a proper source address update the Neighbor Cache as follows. If the router already has a Neighbor Cache entry for the solicitation's sender, the solicitation contains a Source Link-Layer Address option, and the received link-layer address differs from that already in the cache, the link-layer address SHOULD be updated in the appropriate Neighbor Cache entry, and its reachability state MUST also be set to STALE. If there is no existing Neighbor Cache entry for the solicitation's sender, the router creates one, installs the link- layer address and sets its reachability state to STALE as specified in section 7.3.3. `{{Whether or not a Source Link-Layer Address option is provided, if a Neighbor Cache entry for the solicitation's sender exists (or is created) the entry's IsRouter flag MUST be set to FALSE}}`.

RQ_COR_8331 Router Advertisement [Process]

RFC 2461 *Clause:* 6.2.7 ¶1 *Type:* SHOULD *applies to:* Router

Context: The implementation receives valid Router Advertisements from other routers.

Requirement: The implementaton inspects the advertisements and verifies that router-neighbors are advertising consistent information on the link. The implementations inspects and verifies the following minimum set of information: Cur Hop Limit values (except for the unspecified value of zero); the values of the M and O flags; the Reachable Time values (except for the unspecified value of zero); the Retrans Timer values (except for the unspecified value of zero); the values in the MTU options; and the Preferred and Valid Lifetimes for the same prefix.

RFC text: `{{Routers SHOULD inspect valid Router Advertisements sent by other routers and verify that the routers are advertising consistent information on a link}}`. Detected inconsistencies indicate that one or more routers might be misconfigured and SHOULD be logged to system or network management. `{{The minimum set of information to check includes:`
`- Cur Hop Limit values (except for the unspecified value of zero).`
`- Values of the M or O flags.`
`- Reachable Time values (except for the unspecified value of zero).`
`- Retrans Timer values (except for the unspecified value of zero).`
`- Values in the MTU options.`
`- Preferred and Valid Lifetimes for the same prefix}}`. If AdvPreferredLifetime and/or AdvValidLifetime decrement in real time as specified in section 6.2.7 then the comparison of the lifetimes can not compare the content of the fields in the Router Advertisement but must instead compare the time at which the prefix will become deprecated and invalidated, respectively. Due to link propagation delays and potentially poorly synchronized clocks between the routers such comparison SHOULD allow some time skew.

RQ_COR_8332 Router Advertisement [Process]

RFC 2461 *Clause:* 6.2.7 ¶1, ¶2 *Type:* SHOULD *applies to:* Router

Context: The implementation receives valid Router Advertisements from other routers. It inspects the advertisements and determines that a router-neighbor is advertising inconsistent information on the link that causes hosts to switch from one value to another with each received advertisement.

Requirement: The implementation logs the found inconsistency(ies) to system or network management.

RFC text: `{{Routers SHOULD inspect valid Router Advertisements sent by other routers and verify that the routers are advertising consistent information on a link. Detected inconsistencies indicate that one or more routers might be misconfigured and SHOULD be logged to system or network management}}`. The minimum set of information to check includes:
- Cur Hop Limit values (except for the unspecified value of zero).
- Values of the M or O flags.
- Reachable Time values (except for the unspecified value of zero).
- Retrans Timer values (except for the unspecified value of zero).
- Values in the MTU options.
- Preferred and Valid Lifetimes for the same prefix. If AdvPreferredLifetime and/or AdvValidLifetime decrement in real time as specified in section 6.2.7 then the comparison of the lifetimes can not compare the content of the fields in the Router Advertisement but must instead compare the time at which the prefix will become deprecated and invalidated, respectively. Due to link propagation delays and potentially poorly synchronized clocks between the routers such comparison SHOULD allow some time skew.

...

Note that it is not an error for different routers to advertise different sets of prefixes. Also, some routers might leave some fields as unspecified, i.e., with the value zero, while other routers specify values. The logging of errors SHOULD be restricted to conflicting information that causes hosts to switch from one value to another with each received advertisement.

RQ_COR_8333 Router Advertisement [Process]

RFC 2461 *Clause:* 6.2.7 ¶1 *Type:* SHOULD *applies to:* Router

Context: The implementation receives valid Router Advertisements from other routers. It inspects the advertisements and verifies that router-neighbors are advertising consistent information on the link. The inspected Preferred and Valid Lifetimes for a prefix decrement in real time rather than be fixed.

Requirement: The implementation compares the times at which the prefix will be respectively deprecated and invalidated allowing some time skew for link propagation delays and poorly synchronized clocks.

RFC text: Routers SHOULD inspect valid Router Advertisements sent by other routers and verify that the routers are advertising consistent information on a link. Detected inconsistencies indicate that one or more routers might be misconfigured and SHOULD be logged to system or network management. The minimum set of information to check includes:
- Cur Hop Limit values (except for the unspecified value of zero).
- Values of the M or O flags.
- Reachable Time values (except for the unspecified value of zero).
- Retrans Timer values (except for the unspecified value of zero).
- Values in the MTU options.
`{{- Preferred and Valid Lifetimes for the same prefix. If AdvPreferredLifetime and/or AdvValidLifetime decrement in real time as specified in section 6.2.7 then the comparison of the lifetimes can not compare the content of the fields in the Router Advertisement but must instead compare the time at which the prefix will become deprecated and invalidated, respectively. Due to link propagation delays and potentially poorly synchronized clocks between the routers such comparison SHOULD allow some time skew.}}`

RQ_COR_8334 address: Link-local [Form]

RFC 2461 *Clause:* 6.2.8 ¶1 *Type:* SHOULD *applies to:* Router

Context: The implementation is functioning with a given link-local address.

Requirement: The implementation does not change its link-local address.

RFC text: `{{The link-local address on a router SHOULD change rarely, if ever}}`. Nodes receiving Neighbor Discovery messages use the source address to identify the sender. If multiple packets from the same router contain different source addresses, nodes will assume they come from different routers, leading to undesirable behavior. For example, a node will ignore Redirect messages that are believed to have been sent by a router other than the current first-hop router. Thus the source address used in Router Advertisements sent by a particular router must be identical to the target address in a Redirect message when redirecting to that router.

RQ_COR_8335

RFC 2461 *Clause:* 6.2.8 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Redirect message sent by a router other than the current first-hop router for its IP traffic.

Requirement: The implementation ignores the Redirect message [and continues using the current first-hop router for its IP traffic].

RFC text: The link-local address on a router SHOULD change rarely, if ever. Nodes receiving Neighbor Discovery messages use the source address to identify the sender. If multiple packets from the same router contain different source addresses, nodes will assume they come from different routers, leading to undesirable behavior. `{{For example, a node will ignore Redirect messages that are believed to have been sent by a router other than the current first-hop router}}`. Thus the source address used in Router Advertisements sent by a particular router must be identical to the target address in a Redirect message when redirecting to that router.

RQ_COR_8336 Redirect Target Address Field Determination

RFC 2461 *Clause:* 6.2.8 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation generates a Redirect message to a given host.

Requirement: The implementation uses the IP Source Address used in its Router Advertisements to the host as the value in the Target Address field of the Redirect message.

RFC text: The link-local address on a router SHOULD change rarely, if ever. Nodes receiving Neighbor Discovery messages use the source address to identify the sender. If multiple packets from the same router contain different source addresses, nodes will assume they come from different routers, leading to undesirable behavior. For example, a node will ignore Redirect messages that are believed to have been sent by a router other than the current first-hop router. `{{Thus the source address used in Router Advertisements sent by a particular router must be identical to the target address in a Redirect message when redirecting to that router}}`.

RQ_COR_8337 Router Advertisement Behavior on

RFC 2461 *Clause:* 6.2.8 ¶3 *Type:* SHOULD *applies to:* Router

Context: The implementation changes the link-local address for one of its interfaces.

Requirement: The implementation multicasts from the old link-local address a few Router Advertisements with the Router Lifetime field set to zero and multicasts from the new link-local address a few Router Advertisements.

RFC text: If a router changes the link-local address for one of its interfaces, it SHOULD inform hosts of this change. `{{The router SHOULD multicast a few Router Advertisements from the old link-local address with the Router Lifetime field set to zero and also multicast a few Router Advertisements from the new link-local address.}}` The overall effect should be the same as if one interface ceases being an advertising interface, and a different one starts being an advertising interface.

RQ_COR_8338 Initialize

RFC 2461 *Clause:* 6.3.2 ¶2 and ¶4 *Type:* MUST *applies to:* Host

Context: There is no router on the implementation's link or the implementaton has not received any Router Advertisements overriding the default values.

Requirement: For each of its interfaces, the implementation assigns the following default values: Link MTU - the value defined in the specific document that describes how IPv6 operates over the particular link layer (e.g., [IPv6-ETHER]); Current Hop Limit - For sending unicast packets, the value specified in the "Assigned Numbers" RFC [ASSIGNED] that is in effect at the time of implementation; Reachable Time - a uniformly distributed random value between the protocol constants MIN_RANDOM_FACTOR and MAX_RANDOM_FACTOR times the Base Reachable Time in ms; and Retrans Timer - the protocol constant RETRANS_TIMER in milliseconds.

RFC text: {{These variables have default values that are overridden by information received in Router Advertisement messages. The default values are used when there is no router on the link or when all received Router Advertisements have left a particular value unspecified.
...
For each interface:
LinkMTU
The MTU of the link. Default: The value defined in the specific document that describes how IPv6 operates over the particular link layer (e.g., [IPv6-ETHER]).
CurHopLimit
The default hop limit to be used when sending (unicast) IP packets. Default: The value specified in the "Assigned Numbers" RFC [ASSIGNED] that was in effect at the time of implementation.
BaseReachableTime
A base value used for computing the random ReachableTime value. Default: REACHABLE_TIME milliseconds.
ReachableTime
The time a neighbor is considered reachable after receiving a reachability confirmation.
This value should be a uniformly-distributed random value between MIN_RANDOM_FACTOR and MAX_RANDOM_FACTOR times BaseReachableTime milliseconds.}} A new random value should be calculated when BaseReachableTime changes (due to Router Advertisements) or at least every few hours even if no Router Advertisements are received.
{{RetransTimer
The time between retransmissions of Neighbor Solicitation messages to a neighbor when resolving the address or when probing the reachability of a neighbor.
Default: RETRANS_TIMER milliseconds}}

RQ_COR_8339 Initialize

RFC 2461 *Clause:* 6.3.2 ¶2 and ¶4 *Type:* SHOULD *applies to:* Host

Context: There is no router on the implementation's link or the implementaton has not received any Router Advertisements overriding the default values.

Requirement: For each of its interfaces, the implementation assigns the following default values: Base Reachable Time - the protocol constant REACHABLE_TIME in milliseconds used as the base for computing the random Reachable Time value. A new random value is calculated at least every few hours.

RFC text: These variables have default values that are overridden by information received in Router Advertisement messages. The default values are used when there is no router on the link or when all received Router Advertisements have left a particular value unspecified.

...

For each interface:

LinkMTU

The MTU of the link. Default: The value defined in the specific document that describes how IPv6 operates over the particular link layer (e.g., [IPv6-ETHER]).

CurHopLimit

The default hop limit to be used when sending (unicast) IP packets.

Default: The value specified in the "Assigned Numbers" RFC [ASSIGNED] that was in effect at the time of implementation.

BaseReachableTime

A base value used for computing the random ReachableTime value.

Default: REACHABLE_TIME milliseconds.

{{ReachableTime

The time a neighbor is considered reachable after receiving a reachability confirmation.

This value should be a uniformly-distributed random value between MIN_RANDOM_FACTOR and MAX_RANDOM_FACTOR times BaseReachableTime milliseconds. A new random value should be calculated when BaseReachableTime changes (due to Router Advertisements) or at least every few hours even if no Router Advertisements are received}}.

RetransTimer

The time between retransmissions of Neighbor Solicitation messages to a neighbor when resolving the address or when probing the reachability of a neighbor.

Default: RETRANS_TIMER milliseconds

RQ_COR_8340 Initialize

RFC 2461 *Clause:* 6.3.3 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is intializing.

Requirement: For each of its multicast-capable interfaces, the implementation joins the all-nodes multicast address.

RFC text: {{The host joins the all-nodes multicast address on all multicast-capable interfaces}}.

RQ_COR_8341 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is on the same link as several advertising routers.

Requirement: The implementation accepts the union of all received information received through Router Advertisements and all other dynamic means.

RFC text: `{{When multiple routers are present, the information advertised collectively by all routers may be a superset of the information contained in a single Router Advertisement. Moreover, information may also be obtained through other dynamic means, such as stateful autoconfiguration. Hosts accept the union of all received information}}`; the receipt of a Router Advertisement MUST NOT invalidate all information received in a previous advertisement or from another source. However, when received information for a specific parameter (e.g., Link MTU) or option (e.g., Lifetime on a specific Prefix) differs from information received earlier, and the parameter/option can only have one value, the most recently-received information is considered authoritative.

RQ_COR_8342 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is on the same link as several advertising routers and receives a valid Router Advertisement on one of its interfaces.

Requirement: The implementation does not invalidate all information received from previous advertisements and other sources.

RFC text: When multiple routers are present, the information advertised collectively by all routers may be a superset of the information contained in a single Router Advertisement. Moreover, information may also be obtained through other dynamic means, such as stateful autoconfiguration. Hosts accept the union of all received information; `{{the receipt of a Router Advertisement MUST NOT invalidate all information received in a previous advertisement or from another source}}`. However, when received information for a specific parameter (e.g., Link MTU) or option (e.g., Lifetime on a specific Prefix) differs from information received earlier, and the parameter/option can only have one value, the most recently-received information is considered authoritative.

RQ_COR_8343 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is on the same link as several advertising routers and receives a valid Router Advertisement on one of its interfaces. The advertisement contains information for a specific parameter or option that differs from earlier received information. This parameter/option can have only one value.

Requirement: The implementation updates the corresponding value of the parameter/option with the most recently-received information

RFC text: When multiple routers are present, the information advertised collectively by all routers may be a superset of the information contained in a single Router Advertisement. Moreover, information may also be obtained through other dynamic means, such as stateful autoconfiguration. Hosts accept the union of all received information; the receipt of a Router Advertisement MUST NOT invalidate all information received in a previous advertisement or from another source. However, `{{when received information for a specific parameter (e.g., Link MTU) or option (e.g., Lifetime on a specific Prefix) differs from information received earlier, and the parameter/option can only have one value, the most recently-received information is considered authoritative}}`.

RQ_COR_8344 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Router Advertisement with a field containing the Unspecified value.

Requirement: The implementation ignores the field and value. It continues to use the value in use for the corresponding field.

RFC text: {{Some Router Advertisement fields (e.g., Cur Hop Limit, Reachable Time and Retrans Timer) may contain a value denoting unspecified. In such cases, the parameter should be ignored and the host should continue using whatever value it is already using}}. In particular, a host MUST NOT interpret the unspecified value as meaning change back to the default value that was in use before the first Router Advertisement was received. This rule prevents hosts from continually changing an internal variable when one router advertises a specific value, but other routers advertise the unspecified value.

RQ_COR_8345 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement with a field containing the Unspecified value.

Requirement: The implementation does not change the field's value back to its default.

RFC text: {{Some Router Advertisement fields (e.g., Cur Hop Limit, Reachable Time and Retrans Timer) may contain a value denoting unspecified}}. In such cases, the parameter should be ignored and the host should continue using whatever value it is already using. {{In particular, a host MUST NOT interpret the unspecified value as meaning change back to the default value that was in use before the first Router Advertisement was received}}. This rule prevents hosts from continually changing an internal variable when one router advertises a specific value, but other routers advertise the unspecified value.

RQ_COR_8346 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶2+ *Type:* MUST *applies to:* Host

Context: The implementation does not yet have at least two default routers. It receives a valid Router Advertisement on an interface from a new router containing a non-zero Router Lifetime.

Requirement: The implementation adds the advertising router to its default router list and starts a timer set to the advertisement's Router Lifetime value.

RFC text: {{On receipt of a valid Router Advertisement, a host extracts the source address of the packet and does the following:
- If the address is not already present in the host's Default Router List, and the advertisement's Router Lifetime is non-zero, create a new entry in the list, and initialize its invalidation timer value from the advertisement's Router Lifetime field.}}
- If the address is already present in the host's Default Router List as a result of a previously-received advertisement, reset its invalidation timer to the Router Lifetime value in the newly-received advertisement.
- If the address is already present in the host's Default Router List and the received Router Lifetime value is zero, immediately time-out the entry as specified in section 6.3.5.
To limit the storage needed for the Default Router List, a host MAY choose not to store all of the router addresses discovered via advertisements. However, a host MUST retain at least two router addresses and SHOULD retain more. Default router selections are made whenever communication to a destination appears to be failing. Thus, the more routers on the list, the more likely an alternative working router can be found quickly (e.g., without having to wait for the next advertisement to arrive).

RQ_COR_8347 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶2+ *Type:* SHOULD *applies to:* Host

Context: The implementation has at least two default routers. It receives a valid Router Advertisement on an interface from a new router containing a non-zero Router Lifetime.

Requirement: The implementation adds the advertising router to its default router list and starts a timer set to the advertisement's Router Lifetime value.

RFC text: `{{On receipt of a valid Router Advertisement, a host extracts the source address of the packet and does the following:`
`- If the address is not already present in the host's Default Router List, and the advertisement's Router Lifetime is non- zero, create a new entry in the list, and initialize its invalidation timer value from the advertisement's Router Lifetime field.}}`
- If the address is already present in the host's Default Router List as a result of a previously-received advertisement, reset its invalidation timer to the Router Lifetime value in the newly-received advertisement.
- If the address is already present in the host's Default Router List and the received Router Lifetime value is zero, immediately time-out the entry as specified in section 6.3.5.
To limit the storage needed for the Default Router List, a host MAY choose not to store all of the router addresses discovered via advertisements. However, `{{a host MUST retain at least two router addresses and SHOULD retain more}}`. Default router selections are made whenever communication to a destination appears to be failing. Thus, the more routers on the list, the more likely an alternative working router can be found quickly (e.g., without having to wait for the next advertisement to arrive).

RQ_COR_8348 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶2+ *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement on an interface from one of its existing default routers. The advertisement's Router Lifetime field is set to a value other than 0.

Requirement: The implementation resets its invalidation timer for this router to the value in the advertisement's Router Lifetime field.

RFC text: On receipt of a valid Router Advertisement, a host extracts the source address of the packet and does the following:
- If the address is not already present in the host's Default Router List, and the advertisement's Router Lifetime is non- zero, create a new entry in the list, and initialize its invalidation timer value from the advertisement's Router Lifetime field.
`{{- If the address is already present in the host's Default Router List as a result of a previously-received advertisement, reset its invalidation timer to the Router Lifetime value in the newly-received advertisement}}`.
- If the address is already present in the host's Default Router List and the received Router Lifetime value is zero, immediately time-out the entry as specified in section 6.3.5.
To limit the storage needed for the Default Router List, a host MAY choose not to store all of the router addresses discovered via advertisements. However, a host MUST retain at least two router addresses and SHOULD retain more. Default router selections are made whenever communication to a destination appears to be failing. Thus, the more routers on the list, the more likely an alternative working router can be found quickly (e.g., without having to wait for the next advertisement to arrive).

RQ_COR_8349 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶2+, *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement on an interface from one of its existing default routers. The advertisement's Router Lifetime field is set to 0.

Requirement: The implementation immediately times out the router's invalidation timer and no longer uses the default router.

RFC text: On receipt of a valid Router Advertisement, a host extracts the source address of the packet and does the following:
- If the address is not already present in the host's Default Router List, and the advertisement's Router Lifetime is non- zero, create a new entry in the list, and initialize its invalidation timer value from the advertisement's Router Lifetime field.
- If the address is already present in the host's Default Router List as a result of a previously-received advertisement, reset its invalidation timer to the Router Lifetime value in the newly-received advertisement.
{{- If the address is already present in the host's Default Router List and the received Router Lifetime value is zero, immediately time-out the entry as specified in section 6.3.5.}}
To limit the storage needed for the Default Router List, a host MAY choose not to store all of the router addresses discovered via advertisements. However, a host MUST retain at least two router addresses and SHOULD retain more. Default router selections are made whenever communication to a destination appears to be failing. Thus, the more routers on the list, the more likely an alternative working router can be found quickly (e.g., without having to wait for the next advertisement to arrive).
...
{{Whenever the Lifetime of an entry in the Default Router List expires, that entry is discarded}}.

RQ_COR_8350 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶4 *Type:* MAY *applies to:* Host

Context: The implementation has at least two default routers. It receives a valid Router Advertisement on an interface from a new router containing a non-zero Router Lifetime. It limits the storage for its default router information.

Requirement: The host does not use the new advertising router as a default router.

RFC text: {{To limit the storage needed for the Default Router List, a host MAY choose not to store all of the router addresses discovered via advertisements}}. However, a host MUST retain at least two router addresses and SHOULD retain more. Default router selections are made whenever communication to a destination appears to be failing. Thus, the more routers on the list, the more likely an alternative working router can be found quickly (e.g., without having to wait for the next advertisement to arrive).

RQ_COR_8351 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶5 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router on one of its interfaces with the Cur Hop Limit field set to a different value other than 0.

Requirement: The implementation uses the new Cur Hop Limit value for IP communication through the default router.

RFC text: {{If the received Cur Hop Limit value is non-zero the host SHOULD set its CurHopLimit variable to the received value.}}

RQ_COR_8352 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶6 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a non-zero Reachable Time value different from the Reachable Time value received in the previous advertisement from the same router.

Requirement: The implementation recomputes the router's Reachable Time [and behaves accordingly].

RFC text: If the received Reachable Time value is non-zero the host SHOULD set its BaseReachableTime variable to the received value. `{{If the new value differs from the previous value, the host SHOULD recompute a new random ReachableTime value.}}` ReachableTime is computed as a uniformly-distributed random value between MIN_RANDOM_FACTOR and MAX_RANDOM_FACTOR times the BaseReachableTime. Using a random component eliminates the possibility Neighbor Unreachability Detection messages synchronize with each other.

RQ_COR_8353 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶7 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a non-zero Reachable Time value the same as from the Reachable Time value received in the previous advertisements from the same router.

Requirement: The implementation recomputes the router's Reachable Time at least once every few hours [and behaves accordingly].

RFC text: `{{In most cases, the advertised Reachable Time value will be the same in consecutive Router Advertisements and a host's BaseReachableTime rarely changes. In such cases, an implementation SHOULD insure that a new random value gets recomputed at least once every few hours.}}`

RQ_COR_8354 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶8 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a non-zero Retrans Timer field value.

Requirement: The implementation resets the Retrans Timer to the new field value after the currently running Retrans Timer expires.

RFC text: `{{The RetransTimer variable SHOULD be copied from the Retrans Timer field, if the received value is non-zero.}}`

RQ_COR_8355 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶12 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a L-Flag of a Prefix Information option set to 0 (off-link). The Lifetime field in the option is set to a value other than 0.

Requirement: The implementation treats the prefix in the advertisement's option as on-link.

RFC text: Prefix Information options that have the "on-link" (L) flag set indicate a prefix identifying a range of addresses that should be considered on-link. {{Note, however, that a Prefix Information option with the on-link flag set to zero conveys no information concerning on-link determination and MUST NOT be interpreted to mean that addresses covered by the prefix are off-link.}} The only way to cancel a previous on-link indication is to advertise that prefix with the L-bit set and the Lifetime set to zero. The default behavior (see section 5.2) when sending a packet to an address for which no information is known about the on-link status of the address is to forward the packet to a default router; the reception of a Prefix Information option with the "on-link " (L) flag set to zero does not change this behavior. The reasons for an address being treated as on-link is specified in the definition of "on-link" in section 2.1. Prefixes with the on-link flag set to zero would normally have the autonomous flag set and be used by [ADDRCONF].

RQ_COR_8356 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶12 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a L-Flag of a Prefix Information option set to 0 (off-link). The Lifetime field in the option is set to 0.

Requirement: The implementation treats the prefix in the advertisement's option as off-link.

RFC text: Prefix Information options that have the "on-link" (L) flag set indicate a prefix identifying a range of addresses that should be considered on-link. {{Note, however, that a Prefix Information option with the on-link flag set to zero conveys no information concerning on-link determination and MUST NOT be interpreted to mean that addresses covered by the prefix are off-link. The only way to cancel a previous on-link indication is to advertise that prefix with the L-bit set and the Lifetime set to zero. }} The default behavior (see section 5.2) when sending a packet to an address for which no information is known about the on-link status of the address is to forward the packet to a default router; the reception of a Prefix Information option with the "on-link " (L) flag set to zero does not change this behavior. The reasons for an address being treated as on-link is specified in the definition of "on-link" in section 2.1. Prefixes with the on-link flag set to zero would normally have the autonomous flag set and be used by [ADDRCONF].

RQ_COR_8357 address: Destination Address [Generate]

RFC 2461 *Clause:* 6.3.4 ¶12 *Type:* MUST *applies to:* Host

Context: The implementation is generating a packet to an IP address that is not known to be on-link.

Requirement: The implementation forwards the packet to one of its default routers.

RFC text: Prefix Information options that have the "on-link" (L) flag set indicate a prefix identifying a range of addresses that should be considered on-link. Note, however, that a Prefix Information option with the on-link flag set to zero conveys no information concerning on-link determination and MUST NOT be interpreted to mean that addresses covered by the prefix are off-link. The only way to cancel a previous on-link indication is to advertise that prefix with the L-bit set and the Lifetime set to zero. {{The default behavior (see section 5.2) when sending a packet to an address for which no information is known about the on-link status of the address is to forward the packet to a default router}}; the reception of a Prefix Information option with the "on-link " (L) flag set to zero does not change this behavior. The reasons for an address being treated as on-link is specified in the definition of "on-link" in section 2.1. Prefixes with the on-link flag set to zero would normally have the autonomous flag set and be used by [ADDRCONF].

RQ_COR_8358 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶13 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a L-Flag of a Prefix Information option set to 1 (on-link). The prefix in the option is the link-local prefix.

Requirement: The implementation silently ignores the option [and continues processing the remainder of the Prefix Information option].

RFC text: `{{For each Prefix Information option with the on-link flag set, a host does the following:`
`- If the prefix is the link-local prefix, silently ignore the Prefix Information option.}}`
- If the prefix is not already present in the Prefix List, and the Prefix Information option's Valid Lifetime field is non-zero, create a new entry for the prefix and initialize its invalidation timer to the Valid Lifetime value in the Prefix Information option.
- If the prefix is already present in the host's Prefix List as the result of a previously-received advertisement, reset its invalidation timer to the Valid Lifetime value in the Prefix Information option. If the new Lifetime value is zero, time-out the prefix immediately (see section 6.3.5).
- If the Prefix Information option's Valid Lifetime field is zero, and the prefix is not present in the host's Prefix List, silently ignore the option.

RQ_COR_8359 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶13 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a L-Flag of a Prefix Information option set to 1 (on-link). The prefix in the option is new to the implementation and the option's Valid Lifetime is set to a non-zero value.

Requirement: The implementation treats the new prefix as on-link and sets the prefix's invalidation timer to the option's Valid Lifetime field's value.

RFC text: `{{For each Prefix Information option with the on-link flag set, a host does the following:}}`
- If the prefix is the link-local prefix, silently ignore the Prefix Information option.
`{{- If the prefix is not already present in the Prefix List, and the Prefix Information option's Valid Lifetime field is non-zero, create a new entry for the prefix and initialize its invalidation timer to the Valid Lifetime value in the Prefix Information option.}}`
- If the prefix is already present in the host's Prefix List as the result of a previously-received advertisement, reset its invalidation timer to the Valid Lifetime value in the Prefix Information option. If the new Lifetime value is zero, time-out the prefix immediately (see section 6.3.5).
- If the Prefix Information option's Valid Lifetime field is zero, and the prefix is not present in the host's Prefix List, silently ignore the option.

RQ_COR_8360 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶13 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a L-Flag of a Prefix Information option set to 1 (on-link). The prefix in the option is already known to the implementation and the option's Valid Lifetime is set to a non-zero value.

Requirement: The implementation sets the prefix's invalidation timer to the option's Valid Lifetime field's value.

RFC text: `{{For each Prefix Information option with the on-link flag set, a host does the following:}}`
- If the prefix is the link-local prefix, silently ignore the Prefix Information option.
- If the prefix is not already present in the Prefix List, and the Prefix Information option's Valid Lifetime field is non-zero, create a new entry for the prefix and initialize its invalidation timer to the Valid Lifetime value in the Prefix Information option.
`{{- If the prefix is already present in the host's Prefix List as the result of a previously-received advertisement, reset its invalidation timer to the Valid Lifetime value in the Prefix Information option.}}` If the new Lifetime value is zero, time-out the prefix immediately (see section 6.3.5).
- If the Prefix Information option's Valid Lifetime field is zero, and the prefix is not present in the host's Prefix List, silently ignore the option.

RQ_COR_8361 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶13, *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a L-Flag of a Prefix Information option set to 1 (on-link). The prefix in the option is already known to the implementation and the option's Valid Lifetime is set to zero.

Requirement: The implementation immediately times out the prefix's invalidation timer and treats the prefix as off-link.

RFC text: `{{For each Prefix Information option with the on-link flag set, a host does the following:}}`
- If the prefix is the link-local prefix, silently ignore the Prefix Information option.
- If the prefix is not already present in the Prefix List, and the Prefix Information option's Valid Lifetime field is non-zero, create a new entry for the prefix and initialize its invalidation timer to the Valid Lifetime value in the Prefix Information option.
`{{- If the prefix is already present in the host's Prefix List as the result of a previously-received advertisement, reset its invalidation timer to the Valid Lifetime value in the Prefix Information option. If the new Lifetime value is zero, time-out the prefix immediately (see section 6.3.5).}}`
- If the Prefix Information option's Valid Lifetime field is zero, and the prefix is not present in the host's Prefix List, silently ignore the option.

...

`{{Whenever the invalidation timer expires for a Prefix List entry, that entry is discarded.}}` No existing Destination Cache entries need be updated, however. Should a reachability problem arise with an existing Neighbor Cache entry, Neighbor Unreachability Detection will perform any needed recovery.

RQ_COR_8362 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.4 ¶13 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Router Advertisement from a default router with a L-Flag of a Prefix Information option set to 1 (on-link). The prefix in the option is unknown to the implementation and the option's Valid Lifetime is set to zero.

Requirement: The implementation silently ignores the option [and continues processing the remainder of the Prefix Information option].

RFC text: `{{For each Prefix Information option with the on-link flag set, a host does the following:}}`
- If the prefix is the link-local prefix, silently ignore the Prefix Information option.
- If the prefix is not already present in the Prefix List, and the Prefix Information option's Valid Lifetime field is non-zero, create a new entry for the prefix and initialize its invalidation timer to the Valid Lifetime value in the Prefix Information option.
- If the prefix is already present in the host's Prefix List as the result of a previously-received advertisement, reset its invalidation timer to the Valid Lifetime value in the Prefix Information option. If the new Lifetime value is zero, time-out the prefix immediately (see section 6.3.5).
`{{- If the Prefix Information option's Valid Lifetime field is zero, and the prefix is not present in the host's Prefix List, silently ignore the option.}}`

RQ_COR_8363 Neighbor Unreachability Detection

RFC 2461 *Clause:* 6.3.5 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation has started an invalidation timer for a known prefix that is on-link. That timer for the prefix then expires.

Requirement: The implementation treats the prefix as off-link.

RFC text: `{{Whenever the invalidation timer expires for a Prefix List entry, that entry is discarded.}}` No existing Destination Cache entries need be updated, however. Should a reachability problem arise with an existing Neighbor Cache entry, Neighbor Unreachability Detection will perform any needed recovery.

RQ_COR_8364 Next Hop Determination

RFC 2461 *Clause:* 6.3.5 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation has started the Lifetime timer for one of its default routers. The Lifetime timer expires.

Requirement: The implementation stops using the default router and performs next-hop determination for all addresses that were using the now-deleted router.

RFC text: `{{Whenever the Lifetime of an entry in the Default Router List expires, that entry is discarded.}}` When removing a router from the Default Router list, the node MUST update the Destination Cache in such a way that `{{all entries using the router perform next-hop determination again rather than continue sending traffic to the (deleted) router.}}`

RQ_COR_8365 Next Hop Determination

RFC 2461 *Clause:* 6.3.6 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is performing next-hop determination for a packet to be sent off-link. No router is identified to forward the packet.

Requirement: The implementation selects a default router to forward the packet.

RFC text: The algorithm for selecting a router depends in part on whether or not a router is known to be reachable. The exact details of how a node keeps track of a neighbor's reachability state are covered in section 7.3. `{{The algorithm for selecting a default router is invoked during next-hop determination when no Destination Cache entry exists for an off-link destination}}` or when communication through an existing router appears to be failing. Under normal conditions, a router would be selected the first time traffic is sent to a destination, with subsequent traffic for that destination using the same router as indicated in the Destination Cache modulo any changes to the Destination Cache caused by Redirect messages.

RQ_COR_8366 Next Hop Determination

RFC 2461 *Clause:* 6.3.6 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is performing next-hop determination for a packet to be sent off-link. Communication with the default router that forwards the packet fails.

Requirement: The implementation selects a default router to forward the packet.

RFC text: The algorithm for selecting a router depends in part on whether or not a router is known to be reachable. The exact details of how a node keeps track of a neighbor's reachability state are covered in section 7.3. `{{The algorithm for selecting a default router is invoked}}` during next-hop determination when no Destination Cache entry exists for an off-link destination or `{{when communication through an existing router appears to be failing.}}` Under normal conditions, a router would be selected the first time traffic is sent to a destination, with subsequent traffic for that destination using the same router as indicated in the Destination Cache modulo any changes to the Destination Cache caused by Redirect messages.

RQ_COR_8367 Next Hop Determination

RFC 2461 *Clause:* 6.3.6 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation has performed next-hop determination for a packet to be sent off-link. It has sent the off-link packet via the default router. The implementation has not received any Redirect messages pertaining to the off-link destination.

Requirement: The implementation sends subsequent packets for the same off-link destination to the same default router.

RFC text: The algorithm for selecting a router depends in part on whether or not a router is known to be reachable. The exact details of how a node keeps track of a neighbor's reachability state are covered in section 7.3. The algorithm for selecting a default router is invoked during next-hop determination when no Destination Cache entry exists for an off-link destination or when communication through an existing router appears to be failing. `{{Under normal conditions, a router would be selected the first time traffic is sent to a destination, with subsequent traffic for that destination using the same router as indicated in the Destination Cache modulo any changes to the Destination Cache caused by Redirect messages.}}`

RQ_COR_8368 Next Hop Determination

RFC 2461 *Clause:* 6.3.6 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation has a packet to send to a new off-link address. The implementation is selecting a default router to forward the packet.

Requirement: The implementation selects a router that is reachable or probably reachable over a router that is unknown or suspect.

RFC text: `{{The policy for selecting routers from the Default Router List is as follows:`
`1) Routers that are reachable or probably reachable (i.e., in any state other than INCOMPLETE) SHOULD be preferred over routers whose reachability is unknown or suspect (i.e., in the INCOMPLETE state, or for which no Neighbor Cache entry exists).}}` An implementation may choose to always return the same router or cycle through the router list in a round-robin fashion as long as it always returns a reachable or a probably reachable router when one is available.
2) When no routers on the list are known to be reachable or probably reachable, routers SHOULD be selected in a round-robin fashion, so that subsequent requests for a default router do not return the same router until all other routers have been selected. Cycling through the router list in this case ensures that all available routers are actively probed by the Neighbor Unreachability Detection algorithm. A request for a default router is made in conjunction with the sending of a packet to a router, and the selected router will be probed for reachability as a side effect.
3) If the Default Router List is empty, assume that all destinations are on-link as specified in section 5.2.

RQ_COR_8369 Next Hop Determination

RFC 2461 *Clause:* 6.3.6 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation has a packet to send to a new off-link address. The implementation is selecting a default router to forward the packet. There are no known routers that are reachable or probably reachable.

Requirement: The implementation selects in a round-robin fashion a router from the remaining available routers.

RFC text: The policy for selecting routers from the Default Router List is as follows:
1) Routers that are reachable or probably reachable (i.e., in any state other than INCOMPLETE) SHOULD be preferred over routers whose reachability is unknown or suspect (i.e., in the INCOMPLETE state, or for which no Neighbor Cache entry exists). An implementation may choose to always return the same router or cycle through the router list in a round-robin fashion as long as it always returns a reachable or a probably reachable router when one is available.
`{{2) When no routers on the list are known to be reachable or probably reachable, routers SHOULD be selected in a round-robin fashion, so that subsequent requests for a default router do not return the same router until all other routers have been selected.}}`
Cycling through the router list in this case ensures that all available routers are actively probed by the Neighbor Unreachability Detection algorithm. A request for a default router is made in conjunction with the sending of a packet to a router, and the selected router will be probed for reachability as a side effect.
3) If the Default Router List is empty, assume that all destinations are on-link as specified in section 5.2.

RQ_COR_8370 Next Hop Determination

RFC 2461 *Clause:* 6.3.6 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation has a packet to send to a new off-site address. The implementation is selecting a default router to forward the packet. There are no default routers.

Requirement: The implementation assumes the new destinations is on-link [and sends the packet].

RFC text: `{{The policy for selecting routers from the Default Router List is as follows:}}`
1) Routers that are reachable or probably reachable (i.e., in any state other than INCOMPLETE) SHOULD be preferred over routers whose reachability is unknown or suspect (i.e., in the INCOMPLETE state, or for which no Neighbor Cache entry exists). An implementation may choose to always return the same router or cycle through the router list in a round-robin fashion as long as it always returns a reachable or a probably reachable router when one is available.
2) When no routers on the list are known to be reachable or probably reachable, routers SHOULD be selected in a round-robin fashion, so that subsequent requests for a default router do not return the same router until all other routers have been selected. Cycling through the router list in this case ensures that all available routers are actively probed by the Neighbor Unreachability Detection algorithm. A request for a default router is made in conjunction with the sending of a packet to a router, and the selected router will be probed for reachability as a side effect.
`{{3) If the Default Router List is empty, assume that all destinations are on-link as specified in section 5.2.}}`

RQ_COR_8371 Router Solicitation [Generate]

RFC 2461 *Clause:* 6.3.7 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation has just enabled an interface and is not willing to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes.

Requirement: The implementation transmits up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds. MAX_RTR_SOLICITATIONS and RTR_SOLICITATION_INTERVAL are protocol constants

RFC text: `{{When an interface becomes enabled, a host may be unwilling to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. To obtain Router Advertisements quickly, a host SHOULD transmit up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds.}}` Router Solicitations may be sent after any of the following events:

RQ_COR_8372 Initialize

RFC 2461 *Clause:* 6.3.7 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation's interface has just initialized at system startup time.

Requirement: The implementation sends a Router Solicitation to quickly obtain Router Advertisements.

RFC text: When an interface becomes enabled, a host may be unwilling to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. To obtain Router Advertisements quickly, a host SHOULD transmit up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds. `{{Router Solicitations may be sent after any of the following events:`
`- The interface is initialized at system startup time.}}`
- The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management.
- The system changes from being a router to being a host, by having its IP forwarding capability turned off by system management.
- The host attaches to a link for the first time.
- The host re-attaches to a link after being detached for some time.

RQ_COR_8373 Initialize

RFC 2461 *Clause:* 6.3.7 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation's interface has just reinitialized after a temporary interface failure.

Requirement: The implementation sends a Router Solicitation to quickly obtain Router Advertisements.

RFC text: When an interface becomes enabled, a host may be unwilling to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. To obtain Router Advertisements quickly, a host SHOULD transmit up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds. `{{Router Solicitations may be sent after any of the following events:}}`
- The interface is initialized at system startup time.
`{{- The interface is reinitialized after a temporary interface failure}}` or after being temporarily disabled by system management.
- The system changes from being a router to being a host, by having its IP forwarding capability turned off by system management.
- The host attaches to a link for the first time.
- The host re-attaches to a link after being detached for some time.

RQ_COR_8374 Initialize

RFC 2461 *Clause:* 6.3.7 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation's interface has just reinitialized after being temporarily disabled by system management.

Requirement: The implementation sends a Router Solicitation to quickly obtain Router Advertisements.

RFC text: When an interface becomes enabled, a host may be unwilling to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. To obtain Router Advertisements quickly, a host SHOULD transmit up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds. `{{Router Solicitations may be sent after any of the following events:}}`
- The interface is initialized at system startup time.
`{{- The interface is reinitialized}}` after a temporary interface failure or `{{after being temporarily disabled by system management.}}`
- The system changes from being a router to being a host, by having its IP forwarding capability turned off by system management.
- The host attaches to a link for the first time.
- The host re-attaches to a link after being detached for some time.

RQ_COR_8375 Router Solicitation [Generate]

RFC 2461 *Clause:* 6.3.7 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation's interface has just changed from being a router to being a host.

Requirement: The implementation sends a Router Solicitation to quickly obtain Router Advertisements.

RFC text: When an interface becomes enabled, a host may be unwilling to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. To obtain Router Advertisements quickly, a host SHOULD transmit up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds. `{{Router Solicitations may be sent after any of the following events:}}`
- The interface is initialized at system startup time.
- The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management.
`{{- The system changes from being a router to being a host}}`, by having its IP forwarding capability turned off by system management.
- The host attaches to a link for the first time.
- The host re-attaches to a link after being detached for some time.

RQ_COR_8376 Router Solicitation [Generate]

RFC 2461 *Clause:* 6.3.7 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation attaches to a link for the first time.

Requirement: The implementation sends a Router Solicitation to quickly obtain Router Advertisements.

RFC text: When an interface becomes enabled, a host may be unwilling to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. To obtain Router Advertisements quickly, a host SHOULD transmit up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds. `{{Router Solicitations may be sent after any of the following events:}}`
- The interface is initialized at system startup time.
- The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management.
- The system changes from being a router to being a host}}, by having its IP forwarding capability turned off by system management.
`{{- The host attaches to a link for the first time.}}`
- The host re-attaches to a link after being detached for some time.

RQ_COR_8377 Router Solicitation [Generate]

RFC 2461 *Clause:* 6.3.7 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation re-attaches to a link after being detached for some time.

Requirement: The implementation sends a Router Solicitation to quickly obtain Router Advertisements.

RFC text: When an interface becomes enabled, a host may be unwilling to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. To obtain Router Advertisements quickly, a host SHOULD transmit up to MAX_RTR_SOLICITATIONS Router Solicitation messages each separated by at least RTR_SOLICITATION_INTERVAL seconds. `{{Router Solicitations may be sent after any of the following events:}}`
- The interface is initialized at system startup time.
- The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management.
- The system changes from being a router to being a host}}, by having its IP forwarding capability turned off by system management.
- The host attaches to a link for the first time.
`{{- The host re-attaches to a link after being detached for some time.}}`

RQ_COR_8378

RFC 2461 *Clause:* 6.3.7 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation is generating a Router Solicitation to send.

Requirement: The implementation sets the solicitation's Destination Address to the All-Routers multicast address. The Source Address is set to either one of the interface's unicast addresses or the unspecified address.

RFC text: `{{A host sends Router Solicitations to the All-Routers multicast address. The IP source address is set to either one of the interface's unicast addresses or the unspecified address.}}` The Source Link-Layer Address option SHOULD be set to the host's link-layer address, if the IP source address is not the unspecified address.
Before a host sends an initial solicitation, it SHOULD delay the transmission for a random amount of time between 0 and MAX_RTR_SOLICITATION_DELAY. This serves to alleviate congestion when many hosts start up on a link at the same time, such as might happen after recovery from a power failure. If a host has already performed a random delay since the interface became (re)enabled (e.g., as part of Duplicate Address Detection [ADDRCONF]) there is no need to delay again before sending the first Router Solicitation message.

RQ_COR_8379 Router Solicitation: Source Link-Layer Address

RFC 2461 *Clause:* 6.3.7 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation is generating a Router Solicitation to send. The Source Address is set to one of the interface's unicast addresses.

Requirement: The implementation sets the solicitation's Destination Address to the All-Routers multicast address. The Source Link-Layer Address option is set to the host's link-layer address.

RFC text: {{A host sends Router Solicitations to the All-Routers multicast address. The IP source address is set to either one of the interface's unicast addresses}} or the unspecified address. {{The Source Link-Layer Address option SHOULD be set to the host's link-layer address, if the IP source address is not the unspecified address.}}
Before a host sends an initial solicitation, it SHOULD delay the transmission for a random amount of time between 0 and MAX_RTR_SOLICITATION_DELAY. This serves to alleviate congestion when many hosts start up on a link at the same time, such as might happen after recovery from a power failure. If a host has already performed a random delay since the interface became (re)enabled (e.g., as part of Duplicate Address Detection [ADDRCONF]) there is no need to delay again before sending the first Router Solicitation message.

RQ_COR_8380 Router Solicitation Behavior

RFC 2461 *Clause:* 6.3.7 ¶3 *Type:* SHOULD *applies to:* Host

Context: The implementation is generating the initial Router Solicitation to send from an interface.

Requirement: The implementation delays the sending of the solicitation for a random amount of time between 0 and MAX_RTR_SOLICITATION_DELAY (a protocol constant).

RFC text: A host sends Router Solicitations to the All-Routers multicast address. The IP source address is set to either one of the interface's unicast addresses or the unspecified address. The Source Link-Layer Address option SHOULD be set to the host's link-layer address, if the IP source address is not the unspecified address.
{{Before a host sends an initial solicitation, it SHOULD delay the transmission for a random amount of time between 0 and MAX_RTR_SOLICITATION_DELAY.}} This serves to alleviate congestion when many hosts start up on a link at the same time, such as might happen after recovery from a power failure. If a host has already performed a random delay since the interface became (re)enabled (e.g., as part of Duplicate Address Detection [ADDRCONF]) there is no need to delay again before sending the first Router Solicitation message.

RQ_COR_8381 Router Solicitation Behavior

RFC 2461 *Clause:* 6.3.7 ¶3 *Type:* MAY *applies to:* Host

Context: The implementation is generating the initial Router Solicitation to send from an interface. It has already performed a random delay since the interface became enabled.

Requirement: The implementation does not delay for an additional random amount of time before sending the first Router Solicitation message.

RFC text: A host sends Router Solicitations to the All-Routers multicast address. The IP source address is set to either one of the interface's unicast addresses or the unspecified address. The Source Link-Layer Address option SHOULD be set to the host's link-layer address, if the IP source address is not the unspecified address.Before a host sends an initial solicitation, it SHOULD delay the transmission for a random amount of time between 0 and MAX_RTR_SOLICITATION_DELAY.}} This serves to alleviate congestion when many hosts start up on a link at the same time, such as might happen after recovery from a power failure.
{{If a host has already performed a random delay since the interface became (re)enabled (e.g., as part of Duplicate Address Detection [ADDRCONF]) there is no need to delay again before sending the first Router Solicitation message.}}

RQ_COR_8382 Router Solicitation Behavior

RFC 2461 *Clause:* 6.3.7 ¶4 *Type:* MUST *applies to:* Host

Context: The implementation has sent a Router Solicitation and received a valid Router Advertisement with a non-zero Router Lifetime.

Requirement: The host stops sending Router Solicitations until one of the following events occurs: The interface is initialized at system startup time; The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management; The system changes from being a router to being a host; The host attaches to a link for the first time; or The host re-attaches to a link after being detached for some time.

RFC text: `{{Once the host sends a Router Solicitation, and receives a valid Router Advertisement with a non-zero Router Lifetime, the host MUST desist from sending additional solicitations on that interface, until the next time one of the above events occurs.}}` Moreover, a host SHOULD send at least one solicitation in the case where an advertisement is received prior to having sent a solicitation. Unsolicited Router Advertisements may be incomplete (see section 6.2.3); solicited advertisements are expected to contain complete information.

RQ_COR_8383 Router Solicitation Behavior

RFC 2461 *Clause:* 6.3.7 ¶4 *Type:* SHOULD *applies to:* Host

Context: While generating the initial Router Solicitation to send from an interface or waiting the random delay time for sending this solicitation, the implementation receives a valid Router Advertisement with a non-zero Router Lifetime.

Requirement: The implementation sends at least one Router Solicitation on the interface.

RFC text: Once the host sends a Router Solicitation, and receives a valid Router Advertisement with a non-zero Router Lifetime, the host MUST desist from sending additional solicitations on that interface, until the next time one of the above events occurs. `{{Moreover, a host SHOULD send at least one solicitation in the case where an advertisement is received prior to having sent a solicitation. Unsolicited Router Advertisements may be incomplete (see section 6.2.3); solicited advertisements are expected to contain complete information.}}`

RQ_COR_8384 Router Solicitation Behavior

RFC 2461 *Clause:* 6.3.7 ¶5 *Type:* MUST *applies to:* Host

Context: The implementation has just enabled an interface and is not willing to wait for the next unsolicited Router Advertisement to locate default routers or learn prefixes. The implementation has sent MAX_RTR_SOLICITATIONS solicitations and received no Router Advertisements after having waited MAX_RTR_SOLICITATION_DELAY seconds after sending the last solicitation.

Requirement: The implementation concludes that there are no routers on the link for the purpose of [ADDRCONF].

RFC text: `{{If a host sends MAX_RTR_SOLICITATIONS solicitations, and receives no Router Advertisements after having waited MAX_RTR_SOLICITATION_DELAY seconds after sending the last solicitation, the host concludes that there are no routers on the link for the purpose of [ADDRCONF].}}` However, the host continues to receive and process Router Advertisements messages in the event that routers appear on the link.

RQ_COR_8385 Router Advertisement [Process]

RFC 2461 *Clause:* 6.3.7 ¶5 *Type:* MUST *applies to:* Host

Context: The implementation has enabled an interface and determined that the link has no routers. It then receives a valid Router Advertisement message from a router.

Requirement: The implementation processes the advertisement and adds the router to its Default Router List.

RFC text: If a host sends MAX_RTR_SOLICITATIONS solicitations, and receives no Router Advertisements after having waited MAX_RTR_SOLICITATION_DELAY seconds after sending the last solicitation, the host concludes that there are no routers on the link for the purpose of [ADDRCONF]. {{However, the host continues to receive and process Router Advertisements messages in the event that routers appear on the link.}}

RQ_COR_8386 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message with the IP Header Hop Limit field set to a value other than 255.

Requirement: The implementation silently discards the solicitation.

RFC text: {{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:
- The IP Hop Limit field has a value of 255,}} i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8387 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message including an IP Authentication Header. The solicitation does not correctly authenticate.

Requirement: The implementation silently discards the solicitation.

RFC text: {{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
{{- If the message includes an IP Authentication Header, the message authenticates correctly.}}
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8388 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The calculated checksum does not match the value in the solicitation's Checksum field.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
`{{- ICMP Checksum is valid. }}`
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8389 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The ICMP's Code field is set to a value other than 0.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
`{{- ICMP Code is 0.}}`
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8390 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The ICMP length derived from the IP Header's Length field is less than 24 octets.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
`{{- ICMP length (derived from the IP length) is 24 or more octets. }}`
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8391 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The IP Source Address is the Unspecified Address (0::0). The IP Destination Address is not the solicited-node multicast address.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.
`{{- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.}}`
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8392 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The IP Source Address is the Unspecified Address (0::0). The solicitation contains a Source Link-layer Address option.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.
`{{- If the IP source address is the unspecified address, there is no source link-layer address option in the message.}}`

RQ_COR_8393 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The Target Address field is set to a multicast address.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
`{{- Target Address is not a multicast address.}}`
- All included options have a length that is greater than zero.
- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.
- If the IP source address is the unspecified address, there is no source link-layer address option in the message.

RQ_COR_8394 Neighbor Solicitation - Option Anomalies

RFC 2461 *Clause:* 7.1.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The solicitation includes an option whose Length field is set to 0.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A node MUST silently discard any received Neighbor Solicitation messages that do not satisfy all of the following validity checks:}}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- All included options have a length that is greater than zero.}}
`{{- If the IP source address is the unspecified address, the IP destination address is a solicited-node multicast address.`
`- If the IP source address is the unspecified address, there is no source link-layer address option in the message.}}`

RQ_COR_8395 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.1.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message. The solicitation's Reserved field is set to a value other than 0.

Requirement: The implementation ignores the Reserved field's contents [and processes the remainder of the solicitation].

RFC text: `{{The contents of the Reserved field}}`, and of any unrecognized options, `{{MUST be ignored.}}` Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8396 Neighbor Solicitation - Option Anomalies

RFC 2461 *Clause:* 7.1.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message and does not recognize an option in the solicitation.

Requirement: The implementation ignores the unrecognizable option [and processes the remainder of the solicitation].

RFC text: `{{The contents}}` of the Reserved field, and `{{of any unrecognized options, MUST be ignored.}}` Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8397 Neighbor Solicitation - Option Anomalies

RFC 2461 *Clause:* 7.1.1 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message that includes a Target Link-layer Address option.

Requirement: The implementation ignores the Target Link-layer Address option and processes the remainder of the solicitation.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Solicitation messages MUST be ignored and the packet processed as normal.}}` The only defined option that may appear is the Source Link-Layer Address option.

RQ_COR_8398 Neighbor Solicitation - Option Anomalies

RFC 2461 *Clause:* 7.1.1 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message that includes a Prefix Information option.

Requirement: The implementation ignores the Prefix Information option and processes the remainder of the solicitation.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Solicitation messages MUST be ignored and the packet processed as normal.}}` The only defined option that may appear is the Source Link-Layer Address option.

RQ_COR_8399 Neighbor Solicitation - Option Anomalies

RFC 2461 *Clause:* 7.1.1 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message that includes a Redirected Header option.

Requirement: The implementation ignores the Redirected Header option and processes the remainder of the solicitation.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Solicitation messages MUST be ignored and the packet processed as normal.}}` The only defined option that may appear is the Source Link-Layer Address option.

RQ_COR_8400 Neighbor Solicitation - Option Anomalies

RFC 2461 *Clause:* 7.1.1 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation message that includes an MTU option.

Requirement: The implementation ignores the MTU option and processes the remainder of the solicitation.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Solicitation messages MUST be ignored and the packet processed as normal.}}` The only defined option that may appear is the Source Link-Layer Address option.

RQ_COR_8401 Neighbor Advertisement - Field Anomalies

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message containing an IP Header Hop Limit field set to a value other than 255.

Requirement: The implementation silently discards the advertisement.

RFC text: `{{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks:}}`
`{{- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.}}`
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- If the IP Destination Address is a multicast address the Solicited flag is zero.
- All included options have a length that is greater than zero.

RQ_COR_8402 Neighbor Advertisement - Field Anomalies

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message that includes an IP Authentication Header. The solicitation does not correctly authenticate.

Requirement: The implementation silently discards the advertisement.

RFC text: {{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks:}}
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
{{- If the message includes an IP Authentication Header, the message authenticates correctly.}}
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- If the IP Destination Address is a multicast address the Solicited flag is zero.
- All included options have a length that is greater than zero.

RQ_COR_8403 Neighbor Advertisement - Field Anomalies

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message. The advertisement's ICMP Checksum field's value does not match the calculated checksum.

Requirement: The implementation silently discards the advertisement.

RFC text: {{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks:}}
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
{{- ICMP Checksum is valid.}}
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- If the IP Destination Address is a multicast address the Solicited flag is zero.
- All included options have a length that is greater than zero.

RQ_COR_8404 Neighbor Advertisement - Field Anomalies

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message. The advertisement's ICMP Code field's value is not 0.

Requirement: The implementation silently discards the advertisement.

RFC text: {{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks:}}
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
{{- ICMP Code is 0.}}
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- If the IP Destination Address is a multicast address the Solicited flag is zero.
- All included options have a length that is greater than zero.

RQ_COR_8405 Neighbor Advertisement - Field Anomalies

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message. The advertisement's ICMP length derived from the IP Header's Length field's value is less than 24 octets.

Requirement: The implementation silently discards the advertisement.

RFC text: `{{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks: }}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
`{{- ICMP length (derived from the IP length) is 24 or more octets.}}`
- Target Address is not a multicast address.
- If the IP Destination Address is a multicast address the Solicited flag is zero.
- All included options have a length that is greater than zero.

RQ_COR_8406 Neighbor Advertisement - Field Anomalies

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message. The advertisement's Target Address field is set to a multicast address.

Requirement: The implementation silently discards the advertisement.

RFC text: `{{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks: }}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
`{{- Target Address is not a multicast address.}}`
- If the IP Destination Address is a multicast address the Solicited flag is zero.
- All included options have a length that is greater than zero.

RQ_COR_8407 Neighbor Advertisement: Solicited NA [Process]

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message. The advertisement's IP Header's Destination Address is a multicast address and the Solicited flag is set to 1.

Requirement: The implementation silently discards the advertisement.

RFC text: `{{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks: }}`
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
`{{- If the IP Destination Address is a multicast address the Solicited flag is zero.}}`
- All included options have a length that is greater than zero.

RQ_COR_8408 Neighbor Advertisement - Option Anomalies

RFC 2461 *Clause:* 7.1.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message containing on option whose Length field is set to 0.

Requirement: The implementation silently discards the advertisement.

RFC text: `{{A node MUST silently discard any received Neighbor Advertisement messages that do not satisfy all of the following validity checks: }}`

- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 24 or more octets.
- Target Address is not a multicast address.
- If the IP Destination Address is a multicast address the Solicited flag is zero.

`{{- All included options have a length that is greater than zero.}}`

RQ_COR_8409

RFC 2461 *Clause:* 7.1.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message. The solicitation's Reserved field is set to a value other than 0.

Requirement: The implementation ignores the Reserved field's contents [and processes the remainder of the advertisement].

RFC text: `{{The contents of the Reserved field}}`, and of any unrecognized options, `{{MUST be ignored.}}` Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8410 Neighbor Advertisement - Option Anomalies

RFC 2461 *Clause:* 7.1.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message and does not recognize an option in the advertisement.

Requirement: The implementation ignores the unrecognizable option [and processes the remainder of the advertisement].

RFC text: `{{The contents}}` of the Reserved field, and `{{of any unrecognized options, MUST be ignored.}}` Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8411 Neighbor Advertisement - Option Anomalies

RFC 2461 *Clause:* 7.1.2 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message that includes a Source Link-layer option.

Requirement: The implementation ignores the Source Link-layer option and processes the remainder of the advertisement.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Advertisement messages MUST be ignored and the packet processed as normal. The only defined option that may appear is the Target Link-Layer Address option.}}`

RQ_COR_8412 Neighbor Advertisement - Option Anomalies

RFC 2461 *Clause:* 7.1.2 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message that includes a Prefix Information option.

Requirement: The implementation ignores the Prefix Information option and processes the remainder of the advertisement.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Advertisement messages MUST be ignored and the packet processed as normal. The only defined option that may appear is the Target Link-Layer Address option.}}`

RQ_COR_8413 Neighbor Advertisement - Option Anomalies

RFC 2461 *Clause:* 7.1.2 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message that includes a Redirected Header option.

Requirement: The implementation ignores the Redirected Header option and processes the remainder of the advertisement.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Advertisement messages MUST be ignored and the packet processed as normal. The only defined option that may appear is the Target Link-Layer Address option.}}`

RQ_COR_8414 Neighbor Advertisement - Option Anomalies

RFC 2461 *Clause:* 7.1.2 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Advertisement message that includes an MTU option.

Requirement: The implementation ignores the MTU option and processes the remainder of the advertisement.

RFC text: `{{The contents of any defined options that are not specified to be used with Neighbor Advertisement messages MUST be ignored and the packet processed as normal. The only defined option that may appear is the Target Link-Layer Address option.}}`

RQ_COR_8415 Address Resolution

RFC 2461 *Clause:* 7.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation determines that an address is on-link. It is sending a packet to that address. The implementation does not know the link-layer address associated with the packet's Destination Address.

Requirement: The implementation performs address resolution on the Destination Address.

RFC text: Address resolution is the process through which a node determines the link-layer address of a neighbor given only its IP address. `{{Address resolution is performed only on addresses that are determined to be on-link and for which the sender does not know the corresponding link-layer address.}}` Address resolution is never performed on multicast addresses.

RQ_COR_8416 Address Resolution

RFC 2461 *Clause:* 7.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation determines that an address is off-link. It is sending a packet to that address. The implementation does not know the link-layer address associated with the packet's Destination Address.

Requirement: The implementation does not perform address resolution on the Destination Address.

RFC text: Address resolution is the process through which a node determines the link-layer address of a neighbor given only its IP address. `{{Address resolution is performed only on addresses that are determined to be on-link and for which the sender does not know the corresponding link-layer address.}}` Address resolution is never performed on multicast addresses.

RQ_COR_8417 Address Resolution

RFC 2461 *Clause:* 7.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation determines that an address is on-link. It is sending a packet to that address. The implementation knows the link-layer address associated with the packet's Destination Address.

Requirement: The implementation does not perform address resolution on the Destination Address.

RFC text: Address resolution is the process through which a node determines the link-layer address of a neighbor given only its IP address. `{{Address resolution is performed only on addresses that are determined to be on-link and for which the sender does not know the corresponding link-layer address.}}` Address resolution is never performed on multicast addresses.

RQ_COR_8418 Address Resolution

RFC 2461 *Clause:* 7.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is sending a packet to a multicast address.

Requirement: The implementation does not perform address resolution on the Destination Address.

RFC text: Address resolution is the process through which a node determines the link-layer address of a neighbor given only its IP address. Address resolution is performed only on addresses that are determined to be on-link and for which the sender does not know the corresponding link-layer address. `{{Address resolution is never performed on multicast addresses.}}`

RQ_COR_8419 Initialize

RFC 2461 *Clause:* 7.2.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation has just enabled a multicast-capable interface.

Requirement: The implementation joins the all-nodes multicast address on that interface and the solicited-node multicast address corresponding to each of the IP addresses assigned to the interface.

RFC text: `{{When a multicast-capable interface becomes enabled the node MUST join the all-nodes multicast address on that interface, as well as the solicited-node multicast address corresponding to each of the IP addresses assigned to the interface.}}`

RQ_COR_8420 Address Use

RFC 2461 *Clause:* 7.2.1 ¶2 *Type:* MUST *applies to:* Node

Context: A new address is added to an implementation's multicast-capable interface that is already operating.

Requirement: The implementation joins the solicited-node multicast address corresponding to the new address.

RFC text: `{{The set of addresses assigned to an interface may change over time. New addresses might be added and old addresses might be removed [ADDRCONF]. In such cases the node MUST join and leave the solicited-node multicast address corresponding to the new and old addresses, respectively.}}` Note that multiple unicast addresses may map into the same solicited-node multicast address; a node MUST NOT leave the solicited-node multicast group until all assigned addresses corresponding to that multicast address have been removed.

RQ_COR_8421 Address Use

RFC 2461 *Clause:* 7.2.1 ¶2 *Type:* MUST *applies to:* Node

Context: An address is removed from an implementation's multicast-capable interface that is already operating. This is the last unicast address assigned to the solicited-node multicast address.

Requirement: The implementation leaves the solicited-node multicast address and solicited-node multicast group corresponding to the old address.

RFC text: `{{The set of addresses assigned to an interface may change over time. New addresses might be added and old addresses might be removed [ADDRCONF]. In such cases the node MUST join and leave the solicited-node multicast address corresponding to the new and old addresses, respectively. Note that multiple unicast addresses may map into the same solicited-node multicast address; a node MUST NOT leave the solicited-node multicast group until all assigned addresses corresponding to that multicast address have been removed.}}`

RQ_COR_8422 Address Use

RFC 2461 *Clause:* 7.2.1 ¶2 *Type:* MUST *applies to:* Node

Context: An address is removed from an implementation's multicast-capable interface that is already operating. This is not the last unicast address assigned to the solicited-node multicast address.

Requirement: The implementation continues to use the solicited-node multicast address and remains in the solicited-node multicast group corresponding to the old address.

RFC text: `{{The set of addresses assigned to an interface may change over time. New addresses might be added and old addresses might be removed [ADDRCONF]. In such cases the node MUST join and leave the solicited-node multicast address corresponding to the new and old addresses, respectively. Note that multiple unicast addresses may map into the same solicited-node multicast address; a node MUST NOT leave the solicited-node multicast group until all assigned addresses corresponding to that multicast address have been removed.}}`

RQ_COR_8423 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation has a unicast packet to send to a neighbor but does not know the neighbor's link-layer address. The implementation is on a multicast-capable interface.

Requirement: To determine the neighbor's link-layer address, the implementation transmits a Neighbor Solicitation message for address resolution. The solicitation's IP Header Destination Address is the neighbor's the solicited-node multicast address.

RFC text: `{{When a node has a unicast packet to send to a neighbor, but does not know the neighbor's link-layer address, it performs address resolution. For multicast-capable interfaces this entails creating a Neighbor Cache entry in the INCOMPLETE state and transmitting a Neighbor Solicitation message targeted at the neighbor. The solicitation is sent to the solicited-node multicast address corresponding to the target address.}}`

RQ_COR_8424 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.2 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation for address resolution. The source address of the packet provoking the solicitation is the same as one of the addresses assigned to the sending interface.

Requirement: The implementation sets the solicitation's IP Header's Source Address to the provoking packet's source address.

RFC text: `{{If the source address of the packet prompting the solicitation is the same as one of the addresses assigned to the outgoing interface, that address SHOULD be placed in the IP Source Address of the outgoing solicitation.}}` Otherwise, any one of the addresses assigned to the interface should be used. Using the prompting packet's source address when possible insures that the recipient of the Neighbor Solicitation installs in its Neighbor Cache the IP address that is highly likely to be used in subsequent return traffic belonging to the prompting packet's "connection".

RQ_COR_8425 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.2 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation for address resolution. The source address of the packet provoking the solicitation is not the same as one of the addresses assigned to the sending interface.

Requirement: The implementation sets the solicitation's IP Header's Source Address to any of the addresses assigned to the sending interface.

RFC text: If the source address of the packet prompting the solicitation is the same as one of the addresses assigned to the outgoing interface, that address SHOULD be placed in the IP Source Address of the outgoing solicitation. `{{Otherwise, any one of the addresses assigned to the interface should be used.}}` Using the prompting packet's source address when possible insures that the recipient of the Neighbor Solicitation installs in its Neighbor Cache the IP address that is highly likely to be used in subsequent return traffic belonging to the prompting packet's "connection".

RQ_COR_8426 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.2 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation for address resolution. The solicitation's destination is a solicited-node multicast address. The implementation has a link-layer address.

Requirement: The implementation includes a Source Link-layer Address option with its link-layer address in the Address field of the option.

RFC text: `{{If the solicitation is being sent to a solicited-node multicast address, the sender MUST include its link-layer address (if it has one) as a Source Link-Layer Address option.}}` Otherwise, the sender SHOULD include its link-layer address (if it has one) as a Source Link-Layer Address option. Including the source link-layer address in a multicast solicitation is required to give the target an address to which it can send the Neighbor Advertisement. On unicast solicitations, an implementation MAY omit the Source Link-Layer Address option. The assumption here is that if the sender has a peer's link-layer address in its cache, there is a high probability that the peer will also have an entry in its cache for the sender. Consequently, it need not be sent.

RQ_COR_8427 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.2 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation for address resolution. The solicitation's destination is neither a solicited-node multicast address nor a unicast address. The implementation has a link-layer address.

Requirement: The implementation includes a Source Link-layer Address option with its link-layer address in the Address field of the option.

RFC text: If the solicitation is being sent to a solicited-node multicast address, the sender MUST include its link-layer address (if it has one) as a Source Link-Layer Address option. `{{Otherwise, the sender SHOULD include its link-layer address (if it has one) as a Source Link-Layer Address option. Including the source link-layer address in a multicast solicitation is required to give the target an address to which it can send the Neighbor Advertisement. On unicast solicitations, an implementation MAY omit the Source Link-Layer Address option. The assumption here is that if the sender has a peer's link-layer address in its cache, there is a high probability that the peer will also have an entry in its cache for the sender. Consequently, it need not be sent.}}`

RQ_COR_8428 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.2 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation is generating a Neighbor Solicitation for address resolution. The solicitation's destination is a unicast address. The implementation has a link-layer address.

Requirement: The implementation includes a Source Link-layer Address option with its link-layer address in the Address field of the option.

RFC text: If the solicitation is being sent to a solicited-node multicast address, the sender MUST include its link-layer address (if it has one) as a Source Link-Layer Address option. Otherwise, the sender SHOULD include its link-layer address (if it has one) as a Source Link-Layer Address option. Including the source link-layer address in a multicast solicitation is required to give the target an address to which it can send the Neighbor Advertisement. `{{On unicast solicitations, an implementation MAY omit the Source Link-Layer Address option. The assumption here is that if the sender has a peer's link-layer address in its cache, there is a high probability that the peer will also have an entry in its cache for the sender. Consequently, it need not be sent.}}`

RQ_COR_8429 Address Resolution Data Queue Handling

RFC 2461 *Clause:* 7.2.2 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation is in the process of address resolution for a neighbor.

Requirement: The implementation retains a small queue of packets whose IP Destination Address is the neighbor while address resolution is in process. The queue holds at least one packet.

RFC text: `{{While waiting for address resolution to complete, the sender MUST, for each neighbor, retain a small queue of packets waiting for address resolution to complete. The queue MUST hold at least one packet,}}` and MAY contain more. However, the number of queued packets per neighbor SHOULD be limited to some small value. When a queue overflows, the new arrival SHOULD replace the oldest entry. Once address resolution completes, the node transmits any queued packets.

RQ_COR_8430 Address Resolution Data Queue Handling

RFC 2461 *Clause:* 7.2.2 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation is in the process of address resolution for a neighbor. The implementation is retaining a queue of packets to be sent to this neighbor.

Requirement: The size of the queue is limited to a small value.

RFC text: While waiting for address resolution to complete, the sender MUST, for each neighbor, retain a small queue of packets waiting for address resolution to complete. The queue MUST hold at least one packet, and MAY contain more. `{{However, the number of queued packets per neighbor SHOULD be limited to some small value.}}` When a queue overflows, the new arrival SHOULD replace the oldest entry. Once address resolution completes, the node transmits any queued packets.

RQ_COR_8431 Address Resolution Data Queue Handling

RFC 2461 *Clause:* 7.2.2 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation is in the process of address resolution for a neighbor. The implementation is retaining a full queue of packets to be sent to this neighbor. Another packet for the queue arrives during address resolution.

Requirement: The implementation replaces the oldest queue entry with the new packet.

RFC text: While waiting for address resolution to complete, the sender MUST, for each neighbor, retain a small queue of packets waiting for address resolution to complete. The queue MUST hold at least one packet, and MAY contain more. However, the number of queued packets per neighbor SHOULD be limited to some small value. `{{When a queue overflows, the new arrival SHOULD replace the oldest entry.}}` Once address resolution completes, the node transmits any queued packets.

RQ_COR_8432 Address Resolution Data Queue Handling

RFC 2461 *Clause:* 7.2.2 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation is queueing packets for a neighbor whose address is being resolved. The address resolution has just been successfully completed.

Requirement: The implementation transmits all the queued packets.

RFC text: While waiting for address resolution to complete, the sender MUST,for each neighbor, retain a small queue of packets waiting for address resolution to complete. The queue MUST hold at least one packet, and MAY contain more. However, the number of queued packets per neighbor SHOULD be limited to some small value. When a queue overflows, the new arrival SHOULD replace the oldest entry. `{{Once address resolution completes, the node transmits any queued packets.}}`

RQ_COR_8433 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.2 ¶5 *Type:* SHOULD *applies to:* Node

Context: The implementation has transmitted a Neighbor Solicitation for Address Resolution to a neighbor and has not received a valid Neighbor Advertisement in response.

Requirement: The implementation retransmits the Neighbor Solicitation to its neighbor approximately RetransTimer milliseconds after the sending of the unanswered solicitation.

RFC text: `{{While awaiting a response, the sender SHOULD retransmit Neighbor Solicitation messages approximately every RetransTimer milliseconds, even in the absence of additional traffic to the neighbor.}}` Retransmissions MUST be rate-limited to at most one solicitation per neighbor every RetransTimer milliseconds.

RQ_COR_8434 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.2 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation has transmitted multiple Neighbor Solicitations for Address Resolution to a neighbor and has not received a valid Neighbor Advertisement in response to any of the solicitations.

Requirement: The implementation retransmits the Neighbor Solicitations to its neighbor at a maximum rate of one solicition every RetransTimer milliseconds.

RFC text: While awaiting a response, the sender SHOULD retransmit Neighbor Solicitation messages approximately every RetransTimer milliseconds, even in the absence of additional traffic to the neighbor. `{{Retransmissions MUST be rate-limited to at most one solicitation per neighbor every RetransTimer milliseconds.}}`

RQ_COR_8435 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.2 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation has transmitted MAX_MULTICAST_SOLICIT Neighbor Solicitations to a neighbor for Address Resolution. The RetransTimer has expired after the sending of the last solicitation.

Requirement: The implementation determines that address resolution has failed. For each packet in the queue awaiting address resolution, the implementation transmits an ICMP Destination Unreachable message. The IP Address of the ICMP message is the source address of the packet. The implementations sets the Code field of the ICMP message to 3 (Address Unreachable).

RFC text: `{{If no Neighbor Advertisement is received after MAX_MULTICAST_SOLICIT solicitations, address resolution has failed. The sender MUST return ICMP destination unreachable indications with code 3 (Address Unreachable) for each packet queued awaiting address resolution.}}`

RQ_COR_8436 Neighbor Solicitation - Field Anomalies [Process]

RFC 2461 *Clause:* 7.2.3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation whose Target Address field is set to a unicast or anycast address that is not assigned to the implementation's receiving address; nor is the Target Address field set to a unicast address for which the node is offering proxy service, nor is it set to a "tentative" address on which Duplicate Address Detection is being performed.

Requirement: The implementation silently discards the solicitation.

RFC text: `{{A valid Neighbor Solicitation that does not meet any the following requirements MUST be silently discarded:}}`
`{{- The Target Address is a "valid" unicast or anycast address assigned to the receiving interface [ADDRCONF],}}`
- The Target Address is a unicast address for which the node is offering proxy service, or
- The Target Address is a "tentative" address on which Duplicate Address Detection is being performed [ADDRCONF].

RQ_COR_8437 address: Duplicate Address Detection (DAD)

RFC 2461 *Clause:* 7.2.3 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation whose Target Address field is set to a "tentative" address.

Requirement: The implementation processes the solicitation for Duplicate Address Detection.

RFC text: `{{If the Target Address is tentative, the Neighbor Solicitation should be processed as described in [ADDRCONF]}}`. Otherwise, the following description applies. If the Source Address is not the unspecified address and, on link layers that have addresses, the solicitation includes a Source Link-Layer Address option, then the recipient SHOULD create or update the Neighbor Cache entry for the IP Source Address of the solicitation. If an entry does not already exist, the node SHOULD create a new one and set its reachability state to STALE as specified in section 7.3.3. If an entry already exists, and the cached link-layer address differs from the one in the received Source Link-Layer option, the cached address should be replaced by the received address and the entry's reachability state MUST be set to STALE.

RQ_COR_8438 Address Resolution

RFC 2461 *Clause:* 7.2.3 ¶2-3, 5 *Type:* SHOULD *applies to:* Node

Context: The implementation is on a link layer that has addresses and receives a valid Neighbor Solicitation. The solicitation's IP Header Source Address is not the Unspecified Address (0::0) and the solicitation includes a Source Link-Layer Address option. The link-layer address is a known neighbor's address.

Requirement: The implementation performs Address Resolution, updates its information on the known neighbor, and associates the IP Source Address with the neighbor's link-layer address. The neighbor's status as a router or host remains unchanged. The implementation then sends a Neighbor Advertisement message in response to the solicitation.

RFC text: If the Target Address is tentative, the Neighbor Solicitation should be processed as described in [ADDRCONF]. Otherwise, the following description applies. `{{If the Source Address is not the unspecified address and, on link layers that have addresses, the solicitation includes a Source Link-Layer Address option, then the recipient SHOULD}}` create or `{{update the Neighbor Cache entry for the IP Source Address of the solicitation}}`. If an entry does not already exist, the node SHOULD create a new one and set its reachability state to STALE as specified in section 7.3.3. If an entry already exists, and the cached link-layer address differs from the one in the received Source Link-Layer option, the cached address should be replaced by the received address and the entry's reachability state MUST be set to STALE.

If a Neighbor Cache entry is created the IsRouter flag SHOULD be set to FALSE. This will be the case even if the Neighbor Solicitation is sent by a router since the Neighbor Solicitation messages do not contain an indication of whether or not the sender is a router. In the event that the sender is a router, subsequent Neighbor Advertisement or Router Advertisement messages will set the correct IsRouter value. `{{If a Neighbor Cache entry already exists its IsRouter flag MUST NOT be modified.}}`

...

`{{After any updates to the Neighbor Cache, the node sends a Neighbor Advertisement response as described in the next section.}}`

RQ_COR_8439 Address Resolution

RFC 2461 *Clause:* 7.2.3 ¶2-3, 5 *Type:* SHOULD *applies to:* Node

Context: The implementation is on a link layer that has addresses and receives a valid Neighbor Solicitation. The solicitation's IP Header Source Address is not the Unspecified Address (0::0) and the solicitation includes a Source Link-Layer Address option. The link-layer address and Source Address are new to the implementation.

Requirement: The implementation performs Address Resolution, adds the new neighbor to its list of neighbors, associates the IP Source Address with the neighbor's link-layer address, and treats the neighbor as unreachable and as a host. The implementation does not attempt to verify the neighbor's reachability. The implementation then sends a Neighbor Advertisement message in response to the solicitation.

RFC text: If the Target Address is tentative, the Neighbor Solicitation should be processed as described in [ADDRCONF]. Otherwise, the following description applies. If the Source Address is not the unspecified address and, on link layers that have addresses, the solicitation includes a Source Link-Layer Address option, then the recipient SHOULD create or update the Neighbor Cache entry for the IP Source Address of the solicitation. `{{If an entry does not already exist, the node SHOULD create a new one and set its reachability state to STALE as specified in section 7.3.3.}}` If an entry already exists, and the cached link-layer address differs from the one in the received Source Link-Layer option, the cached address should be replaced by the received address and the entry's reachability state MUST be set to STALE.
`{{If a Neighbor Cache entry is created the IsRouter flag SHOULD be set to FALSE.}}` This will be the case even if the Neighbor Solicitation is sent by a router since the Neighbor Solicitation messages do not contain an indication of whether or not the sender is a router. In the event that the sender is a router, subsequent Neighbor Advertisement or Router Advertisement messages will set the correct IsRouter value. If a Neighbor Cache entry already exists its IsRouter flag MUST NOT be modified.

...

`{{After any updates to the Neighbor Cache, the node sends a Neighbor Advertisement response as described in the next section.}}`

RQ_COR_8440 Address Resolution

RFC 2461 *Clause:* 7.2.3 ¶2-3, 5 *Type:* SHOULD *applies to:* Node

Context: The implementation is on a link layer that has addresses and receives a valid Neighbor Solicitation. The solicitation's IP Header Source Address is not the Unspecified Address (0::0) and the solicitation includes a Source Link-Layer Address option. The Source Address is known to the implementation. The link-layer address is new to it.

Requirement: The implementation performs Address Resolution, updates the neighbor information, associates the IP Source Address with the neighbor's new link-layer address, and treats the neighbor as unreachable. The implementation does not attempt to verify the neighbor's reachability. The neighbor's status as a router or host remains unchanged. The implementation then sends a Neighbor Advertisement message in response to the solicitation.

RFC text: If the Target Address is tentative, the Neighbor Solicitation should be processed as described in [ADDRCONF]. Otherwise, the following description applies. If the Source Address is not the unspecified address and, on link layers that have addresses, the solicitation includes a Source Link-Layer Address option, then the recipient SHOULD create or update the Neighbor Cache entry for the IP Source Address of the solicitation. If an entry does not already exist, the node SHOULD create a new one and set its reachability state to STALE as specified in section 7.3.3. `{{If an entry already exists, and the cached link-layer address differs from the one in the received Source Link-Layer option, the cached address should be replaced by the received address and the entry's reachability state MUST be set to STALE.}}`
If a Neighbor Cache entry is created the IsRouter flag SHOULD be set to FALSE. This will be the case even if the Neighbor Solicitation is sent by a router since the Neighbor Solicitation messages do not contain an indication of whether or not the sender is a router. In the event that the sender is a router, subsequent Neighbor Advertisement or Router Advertisement messages will set the correct IsRouter value. `{{If a Neighbor Cache entry already exists its IsRouter flag MUST NOT be modified.}}`
...
`{{After any updates to the Neighbor Cache, the node sends a Neighbor Advertisement response as described in the next section.}}`

RQ_COR_8441 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.3 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation is on a link layer that has addresses and receives a valid Neighbor Solicitation. The solicitation's IP Header Source Address is the Unspecified Address (0::0) and includes a Source Link-layer Address option with a known link-layer address.

Requirement: The implementation does not update any information associated with the known link-layer address.

RFC text: `{{If the Source Address is the unspecified address the node MUST NOT}}` create or `{{update the Neighbor Cache entry.}}`

RQ_COR_8442 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.3 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation is on a link layer that has addresses and receives a valid Neighbor Solicitation. The solicitation's IP Header Source Address is the Unspecified Address (0::0) and includes a Source Link-layer Address option with an unknown link-layer address.

Requirement: The implementation does not add a new neighbor to its neighbor list for the link-layer address.

RFC text: `{{If the Source Address is the unspecified address the node MUST NOT create}}` or update `{{the Neighbor Cache entry.}}`

RQ_COR_8443 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation with its IP Destination Address not being a multicast address. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement.

Requirement: The implementation does not include the Target Link-Layer Address option in the Neighbor Advertisement response.

RFC text: A node sends a Neighbor Advertisement in response to a valid Neighbor Solicitation targeting one of the node's assigned addresses. The Target Address of the advertisement is copied from the Target Address of the solicitation. `{{If the solicitation's IP Destination Address is not a multicast address, the Target Link-Layer Address option MAY be omitted;}}` the neighboring node's cached value must already be current in order for the solicitation to have been received. If the solicitation's IP Destination Address is a multicast address, the Target Link-Layer option MUST be included in the advertisement. Furthermore, if the node is a router, it MUST set the Router flag to one; otherwise it MUST set the flag to zero.

RQ_COR_8444 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation with its IP Destination Address being a multicast address. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement.

Requirement: The implementation includes the Target Link-Layer Address option in the Neighbor Advertisement response.

RFC text: A node sends a Neighbor Advertisement in response to a valid Neighbor Solicitation targeting one of the node's assigned addresses. The Target Address of the advertisement is copied from the Target Address of the solicitation. If the solicitation's IP Destination Address is not a multicast address, the Target Link-Layer Address option MAY be omitted; the neighboring node's cached value must already be current in order for the solicitation to have been received. `{{If the solicitation's IP Destination Address is a multicast address, the Target Link-Layer option MUST be included in the advertisement.}}` Furthermore, if the node is a router, it MUST set the Router flag to one; otherwise it MUST set the flag to zero.

RQ_COR_8445 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation receives a valid Neighbor Solicitation. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement.

Requirement: The implementation sets the advertisement's Router flag to one.

RFC text: A node sends a Neighbor Advertisement in response to a valid Neighbor Solicitation targeting one of the node's assigned addresses. The Target Address of the advertisement is copied from the Target Address of the solicitation. If the solicitation's IP Destination Address is not a multicast address, the Target Link-Layer Address option MAY be omitted; the neighboring node's cached value must already be current in order for the solicitation to have been received. If the solicitation's IP Destination Address is a multicast address, the Target Link-Layer option MUST be included in the advertisement. `{{Furthermore, if the node is a router,}}` it MUST set the Router flag to one; otherwise it MUST set the flag to zero.

RQ_COR_8446 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Neighbor Solicitation. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement.

Requirement: The implementation sets the advertisement's Router flag to zero.

RFC text: A node sends a Neighbor Advertisement in response to a valid Neighbor Solicitation targeting one of the node's assigned addresses. The Target Address of the advertisement is copied from the Target Address of the solicitation. If the solicitation's IP Destination Address is not a multicast address, the Target Link-Layer Address option MAY be omitted; the neighboring node's cached value must already be current in order for the solicitation to have been received. If the solicitation's IP Destination Address is a multicast address, the Target Link-Layer option MUST be included in the advertisement. Furthermore, if the node is a router, it MUST set the Router flag to one; `{{otherwise it MUST set the flag to zero.}}`

RQ_COR_8447 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement. The target's IP Address (i.e. the advertisement's IP Destination Address) is an anycast address.

Requirement: The implementation sets the advertisement's Override flag to zero.

RFC text: `{{If the Target Address is either an anycast address}}` or a unicast address for which the node is providing proxy service, or the Target Link-Layer Address option is not included, `{{the Override flag SHOULD be set to zero.}}` Otherwise, the Override flag SHOULD be set to one. Proper setting of the Override flag ensures that nodes give preference to non-proxy advertisements, even when received after proxy advertisements, and also ensures that the first advertisement for an anycast address "wins".

RQ_COR_8448 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement. The target's IP Address (i.e. the advertisement's IP Destination Address) is a unicast address for which the node is providing proxy service.

Requirement: The implementation sets the advertisement's Override flag to zero.

RFC text: `{{If the Target Address is }}`either an anycast address or `{{a unicast address for which the node is providing proxy service,}}` or the Target Link-Layer Address option is not included, `{{the Override flag SHOULD be set to zero.}}` Otherwise, the Override flag SHOULD be set to one. Proper setting of the Override flag ensures that nodes give preference to non-proxy advertisements, even when received after proxy advertisements, and also ensures that the first advertisement for an anycast address "wins".

RQ_COR_8449 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement. The Target Link-layer Address option is not included in the solicitation.

Requirement: The implementation sets the advertisement's Override flag to zero.

RFC text: `{{If the Target Address is }}`either an anycast address or a unicast address for which the node is providing proxy service, or `{{the Target Link-Layer Address option is not included,}}` `{{the Override flag SHOULD be set to zero.}}` Otherwise, the Override flag SHOULD be set to one. Proper setting of the Override flag ensures that nodes give preference to non-proxy advertisements, even when received after proxy advertisements, and also ensures that the first advertisement for an anycast address "wins".

RQ_COR_8450 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement. The target's IP Address (i.e. the advertisement's IP Destination Address) is not an anycast address. Neither is the target's IP Address (i.e. the advertisement's IP Destination Address) is a unicast address for which the node is providing proxy service. The Target Link-layer Address option is included in the solicitation.

Requirement: The implementation sets the advertisement's Override flag to one.

RFC text: If the Target Address is either an anycast address or a unicast address for which the node is providing proxy service, or the Target Link-Layer Address option is not included, the Override flag SHOULD be set to zero. `{{Otherwise, the Override flag SHOULD be set to one.}}` Proper setting of the Override flag ensures that nodes give preference to non-proxy advertisements, even when received after proxy advertisements, and also ensures that the first advertisement for an anycast address "wins".

RQ_COR_8451 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. The solicitation's IP Source Address is the Unspecified Address (0::0). All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement.

Requirement: The implementation sets the advertisement's Solicited flag to zero and the IP Destination Address to the all-nodes multicast address.

RFC text: `{{If the source of the solicitation is the unspecified address, the node MUST set the Solicited flag to zero and multicast the advertisement to the all-nodes address.}}` Otherwise, the node MUST set the Solicited flag to one and unicast the advertisement to the Source Address of the solicitation.

RQ_COR_8452 Neighbor Solicitation for Address Resolution

RFC 2461 *Clause:* 7.2.4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. The solicitation's IP Source Address is not the Unspecified Address (0::0). All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement.

Requirement: The implementation sets the advertisement's Solicited flag to one and the IP Destination Address to the solicitation's IP Source Address.

RFC text: If the source of the solicitation is the unspecified address, the node MUST set the Solicited flag to zero and multicast the advertisement to the all-nodes address. `{{Otherwise, the node MUST set the Solicited flag to one and unicast the advertisement to the Source Address of the solicitation.}}`

RQ_COR_8453 Neighbor Advertisement: Solicited NA

RFC 2461 *Clause:* 7.2.4 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation is generating the advertisement. The advertisement's Target Address is an anycast.

Requirement: The implementation delays send the advertisement for a random time between 0 and MAX_ANYCAST_DELAY_TIME seconds.

RFC text: `{{If the Target Address is an anycast address the sender SHOULD delay sending a response for a random time between 0 and MAX_ANYCAST_DELAY_TIME seconds.}}`

RQ_COR_8454 Neighbor Solicitation [Generate]

RFC 2461 *Clause:* 7.2.4 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation that does not include a Source Link-Layer Address. All conditions are met for the implementation to respond with a Neighbor Advertisement message. The implementation does not have a link-layer address for the node that sent the solicitation.

Requirement: The implementation initiates Neighbor Discovery to determine the link-layer address of its neighbor by sending a multicast Neighbor Solicitation.

RFC text: Because unicast Neighbor Solicitations are not required to include a Source Link-Layer Address, it is possible that `{{a node sending a solicited Neighbor Advertisement does not have a corresponding link-layer address for its neighbor in its Neighbor Cache. In such situations, a node will first have to use Neighbor Discovery to determine the link-layer address of its neighbor (i.e, send out a multicast Neighbor Solicitation).}}`

RQ_COR_8455 Neighbor Advertisement [Process]

RFC 2461 *Clause:* 7.2.5 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a valid Neighbor Advertisement. The advertisement's target is unknown.

Requirement: The implementation silently discards the advertisement.

RFC text: `{{When a valid Neighbor Advertisement is received (either solicited or unsolicited), the Neighbor Cache is searched for the target's entry. If no entry exists, the advertisement SHOULD be silently discarded.}}` There is no need to create an entry if none exists, since the recipient has apparently not initiated any communication with the target.

RQ_COR_8456 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation is on a link having addresses and performing Address Resolution. The neighbor's link-layer address is unknown. It receives a valid Neighbor Advertisement for Address Resolution. The advertisement omits the Target Link-Layer address option.

Requirement: The implementation silently discards the received advertisement.

RFC text: `{{If the target's Neighbor Cache entry is in the INCOMPLETE state when the advertisement is received, one of two things happens. If the link layer has addresses and no Target Link-Layer address option is included, the receiving node SHOULD silently discard the received advertisement.}}` Otherwise, the receiving node performs the following steps:

RQ_COR_8457 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶3-4 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and performing Address Resolution. The neighbor's link-layer address is unknown. It receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option, its Solicited flag is set to one, its Router flag is set to one.

Requirement: The implementation associates the link-layer address with the advertisement's IP Source Address, considers the neighbor to be a router and reachable, and sends queued packets to the neighbor that were waiting address resolution. The implementation ignores the Override flag.

RFC text: If the target's Neighbor Cache entry is in the INCOMPLETE state when the advertisement is received, one of two things happens. If the link layer has addresses and no Target Link-Layer address option is included, the receiving node SHOULD silently discard the received advertisement. `{{Otherwise, the receiving node performs the following steps:`
`- It records the link-layer address in the Neighbor Cache entry.`
`- If the advertisement's Solicited flag is set, the state of the entry is set to REACHABLE,}}` otherwise it is set to STALE.
\- `{{It sets the IsRouter flag in the cache entry based on the Router flag in the received advertisement.}}`
`{{- It sends any packets queued for the neighbor awaiting address resolution.}}`
`{{Note that the Override flag is ignored if the entry is in the INCOMPLETE state.}}`

RQ_COR_8458 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶3-4 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and performing Address Resolution. The neighbor's link-layer address is unknown. It receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option, its Solicited flag is set to zero, its Router flag is set to one.

Requirement: The implementation associates the link-layer address with the advertisement's IP Source Address, considers the neighbor to be a router and unreachable, and sends queued packets to the neighbor that were waiting address resolution. The implementation ignores the Override flag.

RFC text: If the target's Neighbor Cache entry is in the INCOMPLETE state when the advertisement is received, one of two things happens. If the link layer has addresses and no Target Link-Layer address option is included, the receiving node SHOULD silently discard the received advertisement. {{Otherwise, the receiving node performs the following steps:
- It records the link-layer address in the Neighbor Cache entry.
- If the advertisement's Solicited flag is set,}} the state of the entry is set to REACHABLE, {{otherwise it is set to STALE.}}
-{{It sets the IsRouter flag in the cache entry based on the Router flag in the received advertisement.}}
{{- It sends any packets queued for the neighbor awaiting address resolution.}}
{{Note that the Override flag is ignored if the entry is in the INCOMPLETE state.}}

RQ_COR_8459 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶3-4 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and performing Address Resolution. The neighbor's link-layer address is unknown. It receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option, its Solicited flag is set to one, its Router flag is set to zero.

Requirement: The implementation associates the link-layer address with the advertisement's IP Source Address, considers the neighbor to be a host and reachable, and sends queued packets to the neighbor that were waiting address resolution. The implementation ignores the Override flag.

RFC text: If the target's Neighbor Cache entry is in the INCOMPLETE state when the advertisement is received, one of two things happens. If the link layer has addresses and no Target Link-Layer address option is included, the receiving node SHOULD silently discard the received advertisement. {{Otherwise, the receiving node performs the following steps:
- It records the link-layer address in the Neighbor Cache entry.
- If the advertisement's Solicited flag is set, the state of the entry is set to REACHABLE,}} otherwise it is set to STALE.
-{{It sets the IsRouter flag in the cache entry based on the Router flag in the received advertisement.}}
{{- It sends any packets queued for the neighbor awaiting address resolution.}}
{{Note that the Override flag is ignored if the entry is in the INCOMPLETE state.}}

RQ_COR_8460 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶3-4 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and performing Address Resolution. The neighbor's link-layer address is unknown. It receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option, its Solicited flag is set to zero, its Router flag is set to zero.

Requirement: The implementation associates the link-layer address with the advertisement's IP Source Address, considers the neighbor to be a host and unreachable, and sends queued packets to the neighbor that were waiting address resolution. The implementation ignores the Override flag.

RFC text: If the target's Neighbor Cache entry is in the INCOMPLETE state when the advertisement is received, one of two things happens. If the link layer has addresses and no Target Link-Layer address option is included, the receiving node SHOULD silently discard the received advertisement. {{Otherwise, the receiving node performs the following steps:
- It records the link-layer address in the Neighbor Cache entry.
- If the advertisement's Solicited flag is set,}} the state of the entry is set to REACHABLE, {{otherwise it is set to STALE.}}
-{{It sets the IsRouter flag in the cache entry based on the Router flag in the received advertisement.}}
{{- It sends any packets queued for the neighbor awaiting address resolution.}}
{{Note that the Override flag is ignored if the entry is in the INCOMPLETE state.}}

RQ_COR_8461 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and the neighbor's link-layer address is known (i.e. in any other state than INCOMPLETE). The implementation considers the neighbor to be reachable. It then receives a valid Neighbor Advertisement for Address Resolution for that neighbor. The advertisement includes the Target Link-Layer address option with a link-layer address different than the one currently associated to the neighbor. The advertisement's Override flag is set to zero.

Requirement: The implementation no longer considers the neighbor to be reachable and does not attempt to verify reachability until there is traffic to be sent to the neighbor.

RFC text: If the target's Neighbor Cache entry is in any state other than INCOMPLETE when the advertisement is received, processing becomes quite a bit more complex. {{If the Override flag is clear and the supplied link-layer address differs from that in the cache, then one of two actions takes place: if the state of the entry is REACHABLE, set it to STALE, but do not update the entry in any other way;}} otherwise, the received advertisement should be ignored and MUST NOT update the cache. If the Override flag is set, both the Override flag is clear and the supplied link-layer address is the same as that in the cache, or no Target Link-layer address option was supplied, the received advertisement MUST update the Neighbor Cache entry as follows:

RQ_COR_8462 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and the neighbor's link-layer address is known (i.e. in any other state than INCOMPLETE). The implementation considers the neighbor not to be reachable. It then receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option with a link-layer address different than the one currently associated to the neighbor. The advertisement's Override flag is set to zero.

Requirement: The implementation ignores the advertisement and does change any information concerning the neighbor.

RFC text: If the target's Neighbor Cache entry is in any state other than INCOMPLETE when the advertisement is received, processing becomes quite a bit more complex. `{{If the Override flag is clear and the supplied link-layer address differs from that in the cache, then one of two actions takes place:}}` if the state of the entry is REACHABLE, set it to STALE, but do not update the entry in any other way; `{{otherwise, the received advertisement should be ignored and MUST NOT update the cache.}}` If the Override flag is set, both the Override flag is clear and the supplied link-layer address is the same as that in the cache, or no Target Link-layer address option was supplied, the received advertisement MUST update the Neighbor Cache entry as follows:

RQ_COR_8463 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and the neighbor's link-layer address is known (i.e. in any other state than INCOMPLETE). The implementation considers the neighbor not to be reachable. It then receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option with a link-layer address different than the one currently associated to the neighbor. The advertisement's Override flag is set to one.

Requirement: The implementation updates the association between the neighbor and different link-layer address.

RFC text: `{{If the target's Neighbor Cache entry is in any state other than INCOMPLETE when the advertisement is received, processing becomes quite a bit more complex.}}` If the Override flag is clear and the supplied link-layer address differs from that in the cache, then one of two actions takes place: if the state of the entry is REACHABLE, set it to STALE, but do not update the entry in any other way; otherwise, the received advertisement should be ignored and MUST NOT update the cache. `{{If the Override flag is set}}`, both the Override flag is clear and the supplied link-layer address is the same as that in the cache, or no Target Link-layer address option was supplied, the received advertisement MUST update the Neighbor Cache entry as follows:

- `{{The link-layer address in the Target Link-Layer Address option MUST be inserted in the cache (if one is supplied and is different than the already recorded address).}}`
- If the Solicited flag is set, the state of the entry MUST be set to REACHABLE. If the Solicited flag is zero and the link-layer address was updated with a different address the state MUST be set to STALE. Otherwise, the entry's state remains unchanged.

RQ_COR_8464 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and the neighbor's link-layer address is known (i.e. in any other state than INCOMPLETE). The implementation considers the neighbor not to be reachable. It then receives a valid Neighbor Advertisement for Address Resolution. The advertisement's Override flag is set to one and its Solicited flag is also set to one.

Requirement: The implementation considers the neighbor reachable.

RFC text: `{{If the target's Neighbor Cache entry is in any state other than INCOMPLETE when the advertisement is received, processing becomes quite a bit more complex.}}` If the Override flag is clear and the supplied link-layer address differs from that in the cache, then one of two actions takes place: if the state of the entry is REACHABLE, set it to STALE, but do not update the entry in any other way; otherwise, the received advertisement should be ignored and MUST NOT update the cache. `{{If the Override flag is set}}`, both the Override flag is clear and the supplied link-layer address is the same as that in the cache, or no Target Link-layer address option was supplied, the received advertisement MUST update the Neighbor Cache entry as follows:
- The link-layer address in the Target Link-Layer Address option MUST be inserted in the cache (if one is supplied and is different than the already recorded address).
`{{- If the Solicited flag is set, the state of the entry MUST be set to REACHABLE.}}` If the Solicited flag is zero and the link-layer address was updated with a different address the state MUST be set to STALE. Otherwise, the entry's state remains unchanged.

RQ_COR_8465 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and the neighbor's link-layer address is known (i.e. in any other state than INCOMPLETE). The implementation considers the neighbor not to be reachable. It then receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option with a link-layer address different than the one currently associated to the neighbor. The advertisement's Override flag is set to one and its Solicited flag is set to zero.

Requirement: The implementation updates the association between the neighbor and different link-layer address and considers the neighbor unreachable.

RFC text: `{{If the target's Neighbor Cache entry is in any state other than INCOMPLETE when the advertisement is received, processing becomes quite a bit more complex.}}` If the Override flag is clear and the supplied link-layer address differs from that in the cache, then one of two actions takes place: if the state of the entry is REACHABLE, set it to STALE, but do not update the entry in any other way; otherwise, the received advertisement should be ignored and MUST NOT update the cache. `{{If the Override flag is set}}`, both the Override flag is clear and the supplied link-layer address is the same as that in the cache, or no Target Link-layer address option was supplied, the received advertisement MUST update the Neighbor Cache entry as follows:
- The link-layer address in the Target Link-Layer Address option MUST be inserted in the cache (if one is supplied and is different than the already recorded address).
- If the Solicited flag is set, the state of the entry MUST be set to REACHABLE. `{{If the Solicited flag is zero and the link-layer address was updated with a different address the state MUST be set to STALE. Otherwise, the entry's state remains unchanged.}}`

RQ_COR_8466 Neighbor Solicitation [Process] Solicited

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Advertisement in response to a valid Neighbor Solicitation.

Requirement: The implementation sets the Solicited flag of the advertisement to one.

RFC text: `{{An advertisement's Solicited flag should only be set if the advertisement is a response to a Neighbor Solicitation.}}` Because Neighbor Unreachability Detection Solicitations are sent to the cached link-layer address, receipt of a solicited advertisement indicates that the forward path is working. Receipt of an unsolicited advertisement, however, suggests that a neighbor has urgent information to announce (e.g., a changed link-layer address). If the urgent information indicates a change from what a node is currently using, the node should verify the reachability of the (new) path when it sends the next packet. There is no need to update the state for unsolicited advertisements that do not change the contents of the cache.

RQ_COR_8467 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* SHOULD *applies to:* Node

Context: The implementation receives an unsolicited Neighbor Advertisement that indicates a change to what the implementation is currently using.

Requirement: The implementation verifies the reachability of the (new) when it sends the next packet.

RFC text: An advertisement's Solicited flag should only be set if the advertisement is a response to a Neighbor Solicitation. Because Neighbor Unreachability Detection Solicitations are sent to the cached link-layer address, receipt of a solicited advertisement indicates that the forward path is working. `{{Receipt of an unsolicited advertisement, however, suggests that a neighbor has urgent information to announce (e.g., a changed link-layer address). If the urgent information indicates a change from what a node is currently using, the node should verify the reachability of the (new) path when it sends the next packet.}}` There is no need to update the state for unsolicited advertisements that do not change the contents of the cache.

RQ_COR_8468

RFC 2461 *Clause:* 7.2.5 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Advertisement message.

Requirement: The implementation considers the neighbor as a host or router according to the advertisement's IsRouter flag.

RFC text: `{{- The IsRouter flag in the cache entry MUST be set based on the Router flag in the received advertisement.}}` In those cases where the IsRouter flag changes from TRUE to FALSE as a result of this update, the node MUST remove that router from the Default Router List and update the Destination Cache entries for all destinations using that neighbor as a router as specified in section 7.3.3. This is needed to detect when a node that is used as a router stops forwarding packets due to being configured as a host.

RQ_COR_8469 Neighbor Advertisement [Process]

RFC 2461 *Clause:* 7.2.5 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Advertisement message from a neighbor it considers a router. The advertisement's IsRouter flag is set to FALSE.

Requirement: The implementation no longer uses the neighbor as a default router and performs Neighbor Unreachability detection for all packets forwarded through the deleted router.

RFC text: - The IsRouter flag in the cache entry MUST be set based on the Router flag in the received advertisement. `{{In those cases where the IsRouter flag changes from TRUE to FALSE as a result of this update, the node MUST remove that router from the Default Router List and update the Destination Cache entries for all destinations using that neighbor as a router as specified in section 7.3.3.}}` This is needed to detect when a node that is used as a router stops forwarding packets due to being configured as a host.

RQ_COR_8470 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.2.5 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Advertisement message. The advertisement's Override flag is set. The advertisment also refers to a link-layer address that is unknown to the implementation. Neighbor Unreachability Detection is not in progress.

Requirement: The implementation begins Neighbor Unreachability Detection.

RFC text: The above rules ensure that the cache is updated either when the Neighbor Advertisement takes precedence (i.e., the Override flag is set) or when the Neighbor Advertisement refers to the same link-layer address that is currently recorded in the cache. `{{If none of the above apply, the advertisement prompts future Neighbor Unreachability Detection (if it is not already in progress) by changing the state in the cache entry.}}`

RQ_COR_8471 Neighbor Advertisement - Unsolicited NA

RFC 2461 *Clause:* 7.2.6 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation has determined that its link-layer address has changed.

Requirement: The implementation sends up to MAX_NEIGHBOR_ADVERTISEMENT unsolicited Neighbor Advertisement messages to quickly inform its neighbors of the new link-layer address. The advertisement's IP Header Destination Address is set to the all-nodes multicast address.

RFC text: `{{In some cases a node may be able to determine that its link-layer address has changed (e.g., hot-swap of an interface card) and may wish to inform its neighbors of the new link-layer address quickly. In such cases a node MAY send up to MAX_NEIGHBOR_ADVERTISEMENT unsolicited Neighbor Advertisement messages to the all-nodes multicast address.}}` These advertisements MUST be separated by at least RetransTimer seconds.

RQ_COR_8472 Neighbor Advertisement - Unsolicited NA

RFC 2461 *Clause:* 7.2.6 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation has determined that its link-layer address has changed. To quickly inform its neighbors of the new link-layer address, it sends up to MAX_NEIGHBOR_ADVERTISEMENT unsolicited Neighbor Advertisement messages with the advertisement's IP Header Destination Address is to the all-nodes multicast address.

Requirement: The implementation separate each successive advertisement by at least RetransTimer seconds.

RFC text: In some cases a node may be able to determine that its link-layer address has changed (e.g., hot-swap of an interface card) and may wish to inform its neighbors of the new link-layer address quickly. In such cases a node MAY send up to MAX_NEIGHBOR_ADVERTISEMENT unsolicited Neighbor Advertisement messages to the all-nodes multicast address. `{{These advertisements MUST be separated by at least RetransTimer seconds.}}`

RQ_COR_8473 Neighbor Advertisement - Unsolicited NA

RFC 2461 *Clause:* 7.2.6 ¶1-2 *Type:* MUST *applies to:* Router

Context: The implementation is generating an unsolicited Neighbor Advertisement message to quickly inform its neighbors of a new link-layer address.

Requirement: The implementation sets the advertisement's IP Destination Address to the all-nodes multicast address, the Target Address field to an IP address of the implementation's interface, the Target Link-Layer Address option to the new link-layer address, the Solicited flag to 0, and the Router flag to one.

RFC text: `{{The Target Address field in the unsolicited advertisement is set to an IP address of the interface, and the Target Link-Layer Address option is filled with the new link-layer address. The Solicited flag MUST be set to zero,}}` in order to avoid confusing the Neighbor Unreachability Detection algorithm. `{{If the node is a router, it MUST set the Router flag to one; }}`otherwise it MUST set it to zero. The Override flag MAY be set to either zero or one. In either case, neighboring nodes will immediately change the state of their Neighbor Cache entries for the Target Address to STALE, prompting them to verify the path for reachability. If the Override flag is set to one, neighboring nodes will install the new link-layer address in their caches. Otherwise, they will ignore the new link-layer address, choosing instead to probe the cached address.

RQ_COR_8474 Neighbor Advertisement - Unsolicited NA

RFC 2461 *Clause:* 7.2.6 ¶1-2 *Type:* MUST *applies to:* Host

Context: The implementation is generating an unsolicited Neighbor Advertisement message to quickly inform its neighbors of a new link-layer address.

Requirement: The implementation sets the advertisement's IP Destination Address to the all-nodes multicast address, the Target Address field to an IP address of the implementation's interface, the Target Link-Layer Address option to the new link-layer address, the Solicited flag to 0, and the Router flag to zero.

RFC text: `{{The Target Address field in the unsolicited advertisement is set to an IP address of the interface, and the Target Link-Layer Address option is filled with the new link-layer address. The Solicited flag MUST be set to zero,}}` in order to avoid confusing the Neighbor Unreachability Detection algorithm. `{{If the node is a router, it MUST set the Router flag to one; otherwise it MUST set it to zero.}}` The Override flag MAY be set to either zero or one. In either case, neighboring nodes will immediately change the state of their Neighbor Cache entries for the Target Address to STALE, prompting them to verify the path for reachability. If the Override flag is set to one, neighboring nodes will install the new link-layer address in their caches. Otherwise, they will ignore the new link-layer address, choosing instead to probe the cached address.

RQ_COR_8475 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.2.6 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives an unsolicited Neighbor Advertisement with a new link-layer address for one of its neighbors.

Requirement: The implementation verifies the reachability of the path associated with the new link-layer address.

RFC text: The Target Address field in the unsolicited advertisement is set to an IP address of the interface, and the Target Link-Layer Address option is filled with the new link-layer address. The Solicited flag MUST be set to zero, in order to avoid confusing the Neighbor Unreachability Detection algorithm. If the node is a router, it MUST set the Router flag to one; otherwise it MUST set it to zero. The Override flag MAY be set to either zero or one. `{{In either case, neighboring nodes will immediately change the state of their Neighbor Cache entries for the Target Address to STALE, prompting them to verify the path for reachability.}}` If the Override flag is set to one, neighboring nodes will install the new link-layer address in their caches. Otherwise, they will ignore the new link-layer address, choosing instead to probe the cached address.

RQ_COR_8476 Neighbor Advertisement - Unsolicited NA

RFC 2461 *Clause:* 7.2.6 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation is generating an unsolicited Neighbor Advertisement message to quickly inform its neighbors of a new link-layer address. It has multiple IP addresses assigned to the interface whose link-layer address has changed.

Requirement: The implementation sends a separate Neighbor Advertisement to each of the multiple IP addresses.

RFC text: `{{A node that has multiple IP addresses assigned to an interface MAY multicast a separate Neighbor Advertisement for each address.}}` In such a case the node SHOULD introduce a small delay between the sending of each advertisement to reduce the probability of the advertisements being lost due to congestion.

RQ_COR_8477 Neighbor Advertisement - Unsolicited NA

RFC 2461 *Clause:* 7.2.6 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation is generating an unsolicited Neighbor Advertisement message to quickly inform its neighbors of a new link-layer address. It has multiple IP addresses assigned to the interface whose link-layer address has changed. It is sending separate Neighbor Advertisement to each of the multiple IP addresses.

Requirement: The implementation introduces a small delay between the sending of each advertisement.

RFC text: A node that has multiple IP addresses assigned to an interface MAY multicast a separate Neighbor Advertisement for each address. `{{In such a case the node SHOULD introduce a small delay between the sending of each advertisement to reduce the probability of the advertisements being lost due to congestion.}}`

RQ_COR_8478 Neighbor Advertisement - Unsolicited Proxy NA

RFC 2461 *Clause:* 7.2.6 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation is acting as a proxy for an address. Its link-layer address changes.

Requirement: The implementation sends one or more Neighbor Advertisements with the IP Destination Address set to a multicast address.

RFC text: `{{A proxy MAY multicast Neighbor Advertisements when its link-layer address changes}}` or when it is configured (by system management or other mechanisms) to proxy for an address. If there are multiple nodes that are providing proxy services for the same set of addresses the proxies SHOULD provide a mechanism that prevents multiple proxies from multicasting advertisements for any one address, in order to reduce the risk of excessive multicast traffic.

RQ_COR_8479 Neighbor Advertisement - Unsolicited Proxy NA

RFC 2461 *Clause:* 7.2.6 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation is configured (by system management or other mechanisms) to proxy for an address.

Requirement: The implementation sends one or more Neighbor Advertisements with the IP Destination Address set to a multicast address.

RFC text: `{{A proxy MAY multicast Neighbor Advertisements}}` when its link-layer address changes or `{{when it is configured (by system management or other mechanisms) to proxy for an address.}}` If there are multiple nodes that are providing proxy services for the same set of addresses the proxies SHOULD provide a mechanism that prevents multiple proxies from multicasting advertisements for any one address, in order to reduce the risk of excessive multicast traffic.

RQ_COR_8480 Neighbor Solicitation - Proxy NS [Process]

RFC 2461 *Clause:* 7.2.6 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation is one of multiple nodes that are providing proxy services for the same set of addresses.

Requirement: The implementation provides a mechanism that prevents multiple proxies from multicasting Neighbor Advertisements for any one address.

RFC text: A proxy MAY multicast Neighbor Advertisements when its link-layer address changes or when it is configured (by system management or other mechanisms) to proxy for an address. `{{If there are multiple nodes that are providing proxy services for the same set of addresses the proxies SHOULD provide a mechanism that prevents multiple proxies from multicasting advertisements for any one address,}}` in order to reduce the risk of excessive multicast traffic.

RQ_COR_8481 Neighbor Advertisement: Unsolicited Anycast

RFC 2461 *Clause:* 7.2.6 ¶5 *Type:* MAY *applies to:* Node

Context: The implementation has an anycast address assigned to an interface. The implementation's link-layer address changes.

Requirement: The implementation sends unsolicited Neighbor Advertisements for the anycast address with advertisement's IP Destination Address set to a multicast address.

RFC text: Also, `{{a node belonging to an anycast address MAY multicast unsolicited Neighbor Advertisements for the anycast address when the node's link-layer address changes.}}`

RQ_COR_8482 Address Resolution

RFC 2461 *Clause:* 7.2.7 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is performing Address Resolution on an anycast address.

Requirement: The implementation treats the anycast address as if it were a unicast address during Address Resolution.

RFC text: From the perspective of Neighbor Discovery, anycast addresses are treated just like unicast addresses in most cases. Because an anycast address is syntactically the same as a unicast address, `{{nodes performing address resolution or Neighbor Unreachability Detection on an anycast address treat it as if it were a unicast address. No special processing takes place.}}`

RQ_COR_8483 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.2.7 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is performing Neighbor Unreachability Detection on an anycast address.

Requirement: The implementation treats the anycast address as if it were a unicast address during Neighbor Unreachability Detection.

RFC text: From the perspective of Neighbor Discovery, anycast addresses are treated just like unicast addresses in most cases. Because an anycast address is syntactically the same as a unicast address, `{{nodes performing address resolution or Neighbor Unreachability Detection on an anycast address treat it as if it were a unicast address. No special processing takes place.}}`

RQ_COR_8484 Neighbor Advertisement: Solicited NA

RFC 2461 *Clause:* 7.2.7 ¶2 *Type:* SHOULD *applies to:* Node

Context: The implementation has received a valid Neighbor Solicitation message for one of its anycast address. The implementation is generating the Neighbor Advertisement in response to this solicitation.

Requirement: The implementation sets the Override flag to 0 and delays sending the Neighbor Advertisement response by a random time between 0 and MAX_ANYCAST_DELAY_TIME.

RFC text: `{{Nodes that have an anycast address assigned to an interface treat them exactly the same as if they were unicast addresses with two exceptions. First, Neighbor Advertisements sent in response to a Neighbor Solicitation SHOULD be delayed by a random time between 0 and MAX_ANYCAST_DELAY_TIME to reduce the probability of network congestion. Second, the Override flag in Neighbor Advertisements SHOULD be set to 0, so that when multiple advertisements are received, the first received advertisement is used rather than the most recently received advertisement.}}`

RQ_COR_8485 Neighbor Advertisement - Unsolicited Proxy NA

RFC 2461 *Clause:* 7.2.8 ¶1 *Type:* MAY *applies to:* Router

Context: The implementation is functioning.

Requirement: The implementation proxies for one or more other nodes and sends Neighbor Advertisements to indicate that it is willing to accept packets not explicitly addressed to itself.

RFC text: Under limited circumstances, `{{a router MAY proxy for one or more other nodes, that is, through Neighbor Advertisements indicate that it is willing to accept packets not explicitly addressed to itself.}}` For example, a router might accept packets on behalf of a mobile node that has moved off-link. The mechanisms used by proxy are identical to the mechanisms used with anycast addresses.

RQ_COR_8486 Neighbor Advertisement - Unsolicited Proxy NA

RFC 2461 *Clause:* 7.2.8 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation is functioning, proxies for one or more other nodes, and sends Neighbor Advertisements to indicate that it is willing to accept packets not explicitly addressed to itself.

Requirement: The implementations uses the same mechanisms for sending proxy Neighbor Advertisements as those for sending anycast Neighbor Advertisements.

RFC text: Under limited circumstances, a router MAY proxy for one or more other nodes, that is, through Neighbor Advertisements indicate that it is willing to accept packets not explicitly addressed to itself. For example, a router might accept packets on behalf of a mobile node that has moved off-link. `{{The mechanisms used by proxy are identical to the mechanisms used with anycast addresses.}}`

RQ_COR_8487 Address Use

RFC 2461 *Clause:* 7.2.8 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is a proxy for a node.

Requirement: The implementation joins the solicited-node multicast address(es) that correspond to the IP address(es) assigned to the node for which it is proxying.

RFC text: `{{A proxy MUST join the solicited-node multicast address(es) that correspond to the IP address(es) assigned to the node for which it is proxying.}}`

RQ_COR_8488 Neighbor Solicitation - Proxy NS [Process]

RFC 2461 *Clause:* 7.2.8 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitations for one of the addresses that the implementation proxies.

Requirement: The implementation sends a Neighbor Advertisement response to the solicitation with solicitation's Override flag set to zero.

RFC text: `{{All solicited proxy Neighbor Advertisement messages MUST have the Override flag set to zero.}}` This ensures that if the node itself is present on the link its Neighbor Advertisement (with the Override flag set to one) will take precedence of any advertisement received from a proxy. A proxy MAY send unsolicited advertisements with the Override flag set to one as specified in section 7.2.6, but doing so may cause the proxy advertisement to override a valid entry created by the node itself.

RQ_COR_8489 Neighbor Advertisement - Unsolicited Proxy NA

RFC 2461 *Clause:* 7.2.8 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation is generating an unsolicited Neighbor Advertisement for one of the addresses that the implementation proxies.

Requirement: The implementation sends a Neighbor Advertisement with solicitation's Override flag set to one.

RFC text: All solicited proxy Neighbor Advertisement messages MUST have the Override flag set to zero. This ensures that if the node itself is present on the link its Neighbor Advertisement (with the Override flag set to one) will take precedence of any advertisement received from a proxy. `{{A proxy MAY send unsolicited advertisements with the Override flag set to one as specified in section 7.2.6, but doing so may cause the proxy advertisement to override a valid entry created by the node itself.}}`

RQ_COR_8490 Neighbor Advertisement - Solicited NA

RFC 2461 *Clause:* 7.2.8 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a Neighbor Solicitations for one of the addresses that the implementation proxies.

Requirement: The implementation delays the Neighbor Advertisement response to the solicitation by a random time between 0 and MAX_ANYCAST_DELAY_TIME seconds.

RFC text: Finally, `{{when sending a proxy advertisement in response to a Neighbor Solicitation, the sender should delay its response by a random time between 0 and MAX_ANYCAST_DELAY_TIME seconds.}}`

RQ_COR_8491 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation has determines that a neighbor is unreachable.

Requirement: The implementation performs next-hop determination.

RFC text: When a path to a neighbor appears to be failing, the specific recovery procedure depends on how the neighbor is being used. If the neighbor is the ultimate destination, for example, address resolution should be performed again. If the neighbor is a router, however, attempting to switch to another router would be appropriate. `{{The specific recovery that takes place is covered under next-hop determination; Neighbor Unreachability Detection signals the need for next-hop determination by deleting a Neighbor Cache entry.}}`

RQ_COR_8492 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation determines that the path between it and a unicast neighbor address appears to be failing.

Requirement: The implementation performs Neighbor Unreachability Detection.

RFC text: `{{Neighbor Unreachability Detection is performed only for neighbors to which unicast packets are sent;}}` it is not used when sending to multicast addresses.

RQ_COR_8493 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation determines that the path between it and a multicast neighbor address appears to be failing.

Requirement: The implementation does not perform Neighbor Unreachability Detection.

RFC text: Neighbor Unreachability Detection is performed only for neighbors to which unicast packets are sent; `{{it is not used when sending to multicast addresses.}}`

RQ_COR_8494 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation has recently received a Neighbor Advertisement message that is a response to a Neighbor Solicitation message.

Requirement: The implementation considers the neighbor reachable.

RFC text: `{{A neighbor is considered reachable if the node has recently received a confirmation that packets sent recently to the neighbor were received by its IP layer. Positive confirmation can be gathered in two ways:}}` hints from upper layer protocols that indicate a connection is making "forward progress", or `{{receipt of a Neighbor Advertisement message that is a response to a Neighbor Solicitation message.}}`

RQ_COR_8495 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.1 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation has received hints from upper layer protocols that indicate a connection is making "forward progress". "Forward progress" occurs if the packets received from a remote peer can only be arriving if recent packets sent to that peer are actually reaching it.

Requirement: The implementation considers the neighbor reachable.

RFC text: `{{A neighbor is considered reachable if the node has recently received a confirmation that packets sent recently to the neighbor were received by its IP layer. Positive confirmation can be gathered in two ways: hints from upper layer protocols that indicate a connection is making "forward progress"}}`, or receipt of a Neighbor Advertisement message that is a response to a Neighbor Solicitation message.
`{{A connection makes "forward progress" if the packets received from a remote peer can only be arriving if recent packets sent to that peer are actually reaching it.}}` In TCP, for example, receipt of a (new) acknowledgement indicates that previously sent data reached the peer. Likewise, the arrival of new (non-duplicate) data indicates that earlier acknowledgements are being delivered to the remote peer. If packets are reaching the peer, they must also be reaching the sender's next-hop neighbor; thus "forward progress" is a confirmation that the next-hop neighbor is reachable. For off-link destinations, forward progress implies that the first-hop router is reachable. When available, this upper-layer information SHOULD be used.

RQ_COR_8496 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation determines that "forward progress" occurs between itself and its remote peer.

Requirement: The implementation considers the remote peer and next-hop neighbor on the path to the remote peer as reachable.

RFC text: A connection makes "forward progress" if the packets received from a remote peer can only be arriving if recent packets sent to that peer are actually reaching it.> In TCP, for example, receipt of a (new) acknowledgement indicates that previously sent data reached the peer. Likewise, the arrival of new (non-duplicate) data indicates that earlier acknowledgements are being delivered to the remote peer. `{{If packets are reaching the peer, they must also be reaching the sender's next-hop neighbor; thus "forward progress" is a confirmation that the next-hop neighbor is reachable. }}`For off-link destinations, forward progress implies that the first-hop router is reachable. When available, this upper-layer information SHOULD be used.

RQ_COR_8497 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation determines that "forward progress" occurs between itself and an off-link destination.

Requirement: The implementation considers the off-link destination and the first-hop router on the path to the off-link destination as reachable.

RFC text: A connection makes "forward progress" if the packets received from a remote peer can only be arriving if recent packets sent to that peer are actually reaching it.> In TCP, for example, receipt of a (new) acknowledgement indicates that previously sent data reached the peer. Likewise, the arrival of new (non-duplicate) data indicates that earlier acknowledgements are being delivered to the remote peer. If packets are reaching the peer, they must also be reaching the sender's next-hop neighbor; thus "forward progress" is a confirmation that the next-hop neighbor is reachable. `{{For off-link destinations, forward progress implies that the first-hop router is reachable.}}` When available, this upper-layer information SHOULD be used.

RQ_COR_8498 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.1 ¶2 *Type:* SHOULD *applies to:*

Context: The implementation is performing Neighbor Unreachability Detection for a given path. Upper-layer information is available and indicates that "forward progress" is being made on that path.

Requirement: The implementation uses the upper-layer information in Neighbor Unreachability Detection.

RFC text: A connection makes "forward progress" if the packets received from a remote peer can only be arriving if recent packets sent to that peer are actually reaching it. In TCP, for example, receipt of a (new) acknowledgement indicates that previously sent data reached the peer. Likewise, the arrival of new (non-duplicate) data indicates that earlier acknowledgements are being delivered to the remote peer. If packets are reaching the peer, they must also be reaching the sender's next-hop neighbor; thus "forward progress" is a confirmation that the next-hop neighbor is reachable. For off-link destinations, forward progress implies that the first-hop router is reachable. `{{When available, this upper-layer information SHOULD be used.}}`

RQ_COR_8499 Neighbor Reachability Probing

RFC 2461 *Clause:* 7.3.1 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation is performing Neighbor Unreachability Detection for a given path. No upper-layer hints are available and the node is sending packets to a neighbor.

Requirement: The implementation actively probes the neighbor using unicast Neighbor Solicitation messages to verify that the forward path is still working.

RFC text: In some cases (e.g., UDP-based protocols and routers forwarding packets to hosts) such reachability information may not be readily available from upper-layer protocols. `{{When no hints are available and a node is sending packets to a neighbor, the node actively probes the neighbor using unicast Neighbor Solicitation messages to verify that the forward path is still working.}}`

RQ_COR_8500 Invalid Reachability Indications

RFC 2461 *Clause:* 7.3.1 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation is performing Neighbor Unreachability Detection for a given path. It receives a Router Advertisement for the path with the Solicited flag set to zero.

Requirement: The implementation does not use the advertisement to confirm reachability for the given path.

RFC text: The receipt of a solicited Neighbor Advertisement serves as reachability confirmation, since advertisements with the Solicited flag set to one are sent only in response to a Neighbor Solicitation. `{{Receipt of other Neighbor Discovery messages such as Router Advertisements}}` and Neighbor Advertisement `{{with the Solicited flag set to zero MUST NOT be treated as a reachability confirmation.}}` Receipt of unsolicited messages only confirm the one-way path from the sender to the recipient node. In contrast, Neighbor Unreachability Detection requires that a node keep track of the reachability of the forward path to a neighbor from the its perspective, not the neighbor's perspective. Note that receipt of a solicited advertisement indicates that a path is working in both directions. The solicitation must have reached the neighbor, prompting it to generate an advertisement. Likewise, receipt of an advertisement indicates that the path from the sender to the recipient is working. However, the latter fact is known only to the recipient; the advertisement's sender has no direct way of knowing that the advertisement it sent actually reached a neighbor. From the perspective of Neighbor Unreachability Detection, only the reachability of the forward path is of interest.

RQ_COR_8501 Invalid Reachability Indications

RFC 2461 *Clause:* 7.3.1 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation is performing Neighbor Unreachability Detection for a given path. It receives a Neighbor Advertisement for the path with the Solicited flag set to zero.

Requirement: The implementation does not use the advertisement to confirm reachability for the given path.

RFC text: The receipt of a solicited Neighbor Advertisement serves as reachability confirmation, since advertisements with the Solicited flag set to one are sent only in response to a Neighbor Solicitation. `{{Receipt of }}`other Neighbor Discovery messages such as Router Advertisements and `{{Neighbor Advertisement with the Solicited flag set to zero MUST NOT be treated as a reachability confirmation.}}` Receipt of unsolicited messages only confirm the one-way path from the sender to the recipient node. In contrast, Neighbor Unreachability Detection requires that a node keep track of the reachability of the forward path to a neighbor from the its perspective, not the neighbor's perspective. Note that receipt of a solicited advertisement indicates that a path is working in both directions. The solicitation must have reached the neighbor, prompting it to generate an advertisement. Likewise, receipt of an advertisement indicates that the path from the sender to the recipient is working. However, the latter fact is known only to the recipient; the advertisement's sender has no direct way of knowing that the advertisement it sent actually reached a neighbor. From the perspective of Neighbor Unreachability Detection, only the reachability of the forward path is of interest.

RQ_COR_8502 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation has received positive confirmation within the last ReachableTime milliseconds that the forward path to the neighbor was functioning properly.

Requirement: The implementation considers the forward path to the neighbor as reachable.

RFC text: `{{REACHABLE`
`Positive confirmation was received within the last ReachableTime milliseconds that the forward path to the neighbor was functioning properly.}}` While REACHABLE, no special action takes place as packets are sent.

RQ_COR_8503 Neighbor Reachability Probing

RFC 2461 *Clause:* 7.3.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation has sent a packet on the forward path within the last DELAY_FIRST_PROBE_TIME seconds. More than ReachableTime milliseconds have elapsed since the last positive confirmation was received that the implementations's forward path was functioning properly. The implementation has then started a timer with duration DELAY_FIRST_PROBE_TIME seconds after the ReachableTime has elapsed. This timer expires before reachability confirmation occurs.

Requirement: The implementation sends a Neighbor Solicitation to confirm the forward path's reachability.

RFC text: DELAY
`{{More than ReachableTime milliseconds have elapsed since the last positive confirmation was received that the forward path was functioning properly, and a packet was sent within the last DELAY_FIRST_PROBE_TIME seconds. If no reachability confirmation is received within DELAY_FIRST_PROBE_TIME seconds of entering the DELAY state, send a Neighbor Solicitation}}` and change the state to PROBE.

RQ_COR_8504 Neighbor Reachability Probing

RFC 2461 *Clause:* 7.3.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is performing Neighbor Unreachability Detection for a given path.

Requirement: The implementation retransmits Neighbor Solicitations every RetransTimer milliseconds until a reachability confirmation is received.

RFC text: PROBE
A reachability confirmation is actively sought by retransmitting Neighbor Solicitations every RetransTimer milliseconds until a reachability confirmation is received.

RQ_COR_8505 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is performing Neighbor Unreachability Detection for a given path to a neighbor.

Requirement: The continues continues sending packets using the known link-layer address in parallel with performing Neighbor Unreachability Detection to that neighbor.

RFC text: `{{Neighbor Unreachability Detection operates in parallel with the sending of packets to a neighbor. While reasserting a neighbor's reachability, a node continues sending packets to that neighbor using the cached link-layer address.}}` If no traffic is sent to a neighbor, no probes are sent.

RQ_COR_8506 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is not sending traffic to a neighbor.

Requirement: The implementation does not send Neighbor Unreachability Detection probes to the neighbor.

RFC text: Neighbor Unreachability Detection operates in parallel with the sending of packets to a neighbor. While reasserting a neighbor's reachability, a node continues sending packets to that neighbor using the cached link-layer address. `{{If no traffic is sent to a neighbor, no probes are sent.}}`

RQ_COR_8507 Next Hop Determination

RFC 2461 *Clause:* 7.3.3 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is performing address resolution on a neighboring address. The address resolution then fails.

Requirement: The implementation performs next-hop determination to try alternate default routers on the forward path.

RFC text: When a node needs to perform address resolution on a neighboring address, it creates an entry in the INCOMPLETE state and initiates address resolution as specified in section 7.2. `{{If address resolution fails, the entry SHOULD be deleted, so that subsequent traffic to that neighbor invokes the next-hop determination procedure again. Invoking next-hop determination at this point insures that alternate default routers are tried.}}`

RQ_COR_8508 Invalid Reachability Indications

RFC 2461 *Clause:* 7.3.3 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation has no link-layer address known for a forward path for which it is performing Neighbor Unreachability Detection. The implementation's upper layer provides reachability confirmation.

Requirement: The implementation does not consider the neighbor reachable.

RFC text: When a reachability confirmation is received (either through upper- layer advice or a solicited Neighbor Advertisement) an entry's state changes to REACHABLE. `{{The one exception is that upper-layer advice has no effect on entries in the INCOMPLETE state (e.g., for which no link-layer address is cached).}}`

RQ_COR_8509 Neighbor Reachability Probing

RFC 2461 *Clause:* 7.3.3 ¶4-6 *Type:* MUST *applies to:* Node

Context: The implementation has waited ReachableTime milliseconds since receipt of the last reachability confirmation for a neighbor. It then sends a packet to the neighbor and starts a timer of DELAY_FIRST_PROBE_TIME seconds duration. This timer expires without receipt of reachability confirmation.

Requirement: The implementation retransmits MAX_UNICAST_SOLICIT Neighbor Solicitation messages every RetransTimer milliseconds.

RFC text: `{{When ReachableTime milliseconds have passed since receipt of the last reachability confirmation for a neighbor,}}` the Neighbor Cache entry's state changes from REACHABLE to STALE.

Note: An implementation may actually defer changing the state from REACHABLE to STALE until a packet is sent to the neighbor, i.e., there need not be an explicit timeout event associated with the expiration of ReachableTime.

`{{The first time a node sends a packet to a neighbor whose entry is STALE, the sender changes the state to DELAY and a sets a timer to expire in DELAY_FIRST_PROBE_TIME seconds. If the entry is still in the DELAY state when the timer expires, the entry's state changes to PROBE.}}` If reachability confirmation is received, the entry's state changes to REACHABLE. `{{Upon entering the PROBE state, a node sends a unicast Neighbor Solicitation message to the neighbor using the cached link-layer address. While in the PROBE state, a node retransmits Neighbor Solicitation messages every RetransTimer milliseconds}}` until reachability confirmation is obtained. Probes are retransmitted even if no additional packets are sent to the neighbor. If no response is received after waiting RetransTimer milliseconds after sending the MAX_UNICAST_SOLICIT solicitations, retransmissions cease and the entry SHOULD be deleted. Subsequent traffic to that neighbor will recreate the entry and performs address resolution again.

RQ_COR_8510 Neighbor Reachability Probing

RFC 2461 *Clause:* 7.3.3 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation has waited ReachableTime milliseconds since receipt of the last reachability confirmation for a neighbor. It then sends a packet to the neighbor and starts a timer of DELAY_FIRST_PROBE_TIME seconds duration. This timer expires without receipt of reachability confirmation. The implementation retransmits MAX_UNICAST_SOLICIT Neighbor Solicitation messages every RetransTimer milliseconds. No response is received after waiting RetransTimer milliseconds after sending the MAX_UNICAST_SOLICIT solicitations.

Requirement: The implementation ceases retransmissions of the Neighbor Solicitation messages. The implementation deletes the neighbor from its known neighbor list. Subsequent traffic to this neighbor adds the neighbor to the known neighbor list.

RFC text: Upon entering the PROBE state, a node sends a unicast Neighbor Solicitation message to the neighbor using the cached link-layer address. While in the PROBE state, a node retransmits Neighbor Solicitation messages every RetransTimer milliseconds until reachability confirmation is obtained. Probes are retransmitted even if no additional packets are sent to the neighbor. `{{If no response is received after waiting RetransTimer milliseconds after sending the MAX_UNICAST_SOLICIT solicitations, retransmissions cease and the entry SHOULD be deleted. Subsequent traffic to that neighbor will recreate the entry and performs address resolution again.}}`

RQ_COR_8511 Neighbor Reachability Probing

RFC 2461 *Clause:* 7.3.3 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation is generating Neighbor Solicitations for each neighbor.

Requirement: The implementation does not send Neighbor Solicitations to the same neighbor more frequently than once every RetransTimer milliseconds.

RFC text: Note that all Neighbor Solicitations are rate-limited on a per-neighbor basis. `{{A node MUST NOT send Neighbor Solicitations to the same neighbor more frequently than once every RetransTimer milliseconds.}}`

RQ_COR_8512 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Solicitation that indicates there is a new neighbor on the link.

Requirement: The implementation recognizes the new neighbor and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: `{{A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements (i.e., Router Solicitations,}}` Router Advertisements, Redirects, and Neighbor Solicitations). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.

RQ_COR_8513 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation receives a Router Advertisement that indicates there is a new neighbor on the link.

Requirement: The implementation recognizes the new neighbor and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: `{{A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements }}`(i.e., Router Solicitations, `{{Router Advertisements,}}` Redirects, and Neighbor Solicitations). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.

RQ_COR_8514 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation receives a Redirect that indicates there is a new neighbor on the link.

Requirement: The implementation recognizes the new neighbor and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: `{{A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements }}`(i.e., Router Solicitations, Router Advertisements, `{{Redirects,}}` and Neighbor Solicitations). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.

RQ_COR_8515 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Solicitation that indicates there is a new neighbor on the link.

Requirement: The implementation recognizes the new neighbor and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: `{{A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements }}`(i.e., Router Solicitations, Router Advertisements, Redirects, and `{{Neighbor Solicitations}}`). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.

RQ_COR_8516 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a Router Solicitation that indicates a known neighbor's link-layer address is modified.

Requirement: The implementation updates the associations between the neighbor's IP addresses and the new link-layer address and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements (i.e., Router Solicitations, Router Advertisements, Redirects, and Neighbor Solicitations). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. `{{In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.}}`

RQ_COR_8517 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a Router Advertisement that indicates a known neighbor's link-layer address is modified.

Requirement: The implementation updates the associations between the neighbor's IP addresses and the new link-layer address and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements (i.e., Router Solicitations, Router Advertisements, Redirects, and Neighbor Solicitations). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. `{{In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.}}`

RQ_COR_8518 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a Redirect that indicates a known neighbor's link-layer address is modified.

Requirement: The implementation updates the associations between the neighbor's IP addresses and the new link-layer address and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements (i.e., Router Solicitations, Router Advertisements, Redirects, and Neighbor Solicitations). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. `{{In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.}}`

RQ_COR_8519 Neighbor Reachability Determination Startup

RFC 2461 *Clause:* 7.3.3 ¶8 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a Neighbor Solicitation that indicates a known neighbor's link-layer address is modified.

Requirement: The implementation updates the associations between the neighbor's IP addresses and the new link-layer address and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: A Neighbor Cache entry enters the STALE state when created as a result of receiving packets other than solicited Neighbor Advertisements (i.e., Router Solicitations, Router Advertisements, Redirects, and Neighbor Solicitations). These packets contain the link-layer address of either the sender or, in the case of Redirect, the redirection target. However, receipt of these link-layer addresses does not confirm reachability of the forward-direction path to that node. Placing a newly created Neighbor Cache entry for which the link-layer address is known in the STALE state provides assurance that path failures are detected quickly. `{{In addition, should a cached link-layer address be modified due to receiving one of the above messages the state SHOULD also be set to STALE to provide prompt verification that the path to the new link-layer address is working.}}`

RQ_COR_8520 Neighbor Advertisement [Process]

RFC 2461 *Clause:* 7.3.3 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Advertisment message.

Requirement: The implementation inspects the advertisement's Router flag to determines if the neighbor has changed from being a router to a host or vice versa.

RFC text: `{{To properly detect the case where a router switches from being a router to being a host (e.g., if its IP forwarding capability is turned off by system management), a node MUST compare the Router flag field in all received Neighbor Advertisement messages with the IsRouter flag recorded in the Neighbor Cache entry.}}` When a node detects that a neighbor has changed from being a router to being a host, the node MUST remove that router from the Default Router List and update the Destination Cache as described in section 6.3.5. Note that a router may not be listed in the Default Router List, even though a Destination Cache entry is using it (e.g., a host was redirected to it). In such cases, all Destination Cache entries that reference the (former) router must perform next-hop determination again before using the entry.

RQ_COR_8521

RFC 2461 *Clause:* 7.3.3 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Advertisment message. The implementation inspects the advertisement's Router flag and determines that the neighbor has changed from being a router to a host.

Requirement: The node implementation longer uses the neighbor as a default router.

RFC text: To properly detect the case where a router switches from being a router to being a host (e.g., if its IP forwarding capability is turned off by system management), a node MUST compare the Router flag field in all received Neighbor Advertisement messages with the IsRouter flag recorded in the Neighbor Cache entry. `{{When a node detects that a neighbor has changed from being a router to being a host, the node MUST remove that router from the Default Router List and update the Destination Cache as described in section 6.3.5.}}` Note that a router may not be listed in the Default Router List, even though a Destination Cache entry is using it (e.g., a host was redirected to it). In such cases, all Destination Cache entries that reference the (former) router must perform next-hop determination again before using the entry.

RQ_COR_8522 Neighbor Advertisement [Process]

RFC 2461 *Clause:* 7.3.3 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Advertisment message. The implementation inspects the advertisement's Router flag and determines that the neighbor has changed from being a router to a host. The router is not a default router for the implementation (e.g., a host was redirected to it).

Requirement: The implementation performs next-hop determination for all addresses forwarded through the former router.

RFC text: To properly detect the case where a router switches from being a router to being a host (e.g., if its IP forwarding capability is turned off by system management), a node MUST compare the Router flag field in all received Neighbor Advertisement messages with the IsRouter flag recorded in the Neighbor Cache entry. `{{When a node detects that a neighbor has changed from being a router to being a host, the node MUST remove that router from the Default Router List and update the Destination Cache as described in section 6.3.5.}}` Note that a router may not be listed in the Default Router List, even though a Destination Cache entry is using it (e.g., a host was redirected to it). In such cases, all Destination Cache entries that reference the (former) router must perform next-hop determination again before using the entry.

RQ_COR_8523 Neighbor Unreachability Detection

RFC 2461 *Clause:* 7.3.3 ¶10 *Type:* MAY *applies to:* Node

Context: The implementation has link-specific information that indicates that a path to a neighbor has failed.

Requirement: The implementation uses the link-specific information to purge Neighbor Cache entries before Neighbor Unreachability Detection does so.

RFC text: `{{In some cases, link-specific information may indicate that a path to a neighbor has failed (e.g., the resetting of a virtual circuit). In such cases, link-specific information may be used to purge Neighbor Cache entries before the Neighbor Unreachability Detection would do so.}}` However, link-specific information MUST NOT be used to confirm the reachability of a neighbor; such information does not provide end-to-end confirmation between neighboring IP layers.

RQ_COR_8524 Invalid Reachability Indications

RFC 2461 *Clause:* 7.3.3 ¶10 *Type:* MUST *applies to:* Node

Context: The implementation has link-specific information that indicates that a path to a neighbor has failed.

Requirement: The implementation does not use the link-specific information to confirm the neighbor's reachability.

RFC text: `{{In some cases, link-specific information may indicate that a path to a neighbor has failed (e.g., the resetting of a virtual circuit). In such cases, link-specific information may be used to purge Neighbor Cache entries before the Neighbor Unreachability Detection would do so.}}` However, link-specific information MUST NOT be used to confirm the reachability of a neighbor; such information does not provide end-to-end confirmation between neighboring IP layers.

RQ_COR_8525 Redirect Message [Generate]

RFC 2461 *Clause:* 8 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation knows of a better first-hop router for a specific destination.

Requirement: The implementation sends a Redirect message to the host that indicates the better first-hop router.

RFC text: `{{Redirect messages are sent by routers to redirect a host to a better first-hop router for a specific destination}}` or to inform hosts that a destination is in fact a neighbor (i.e., on-link). The latter is accomplished by having the ICMP Target Address be equal to the ICMP Destination Address.

RQ_COR_8526 Redirect Message [Generate]

RFC 2461 *Clause:* 8 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation wants to inform hosts that a destination is in fact a neighbor (i.e., on-link).

Requirement: The implementation sends a Redirect message to the host that indicates the destination is a neighbor. The redirect's ICMP Target Address field is set to the ICMP Destination Address.

RFC text: `{{Redirect messages are sent by routers }}`to redirect a host to a better first-hop router for a specific destination or `{{to inform hosts that a destination is in fact a neighbor (i.e., on-link). The latter is accomplished by having the ICMP Target Address be equal to the ICMP Destination Address.}}`

RQ_COR_8527 Redirect Target Address Field Determination

RFC 2461 *Clause:* 8 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation is using the link-local address of a neighbor router in the Target Address field of a Redirect message.

Requirement: The implementation is able to determine the link-local address for each of its neighboring routers.

RFC text: `{{A router MUST be able to determine the link-local address for each of its neighboring routers in order to ensure that the target address in a Redirect message identifies the neighbor router by its link-local address.}}` For static routing this requirement implies that the next-hop router's address should be specified using the link-local address of the router. For dynamic routing this requirement implies that all IPv6 routing protocols must somehow exchange the link-local addresses of neighboring routers.

RQ_COR_8528 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose IP Source Address is not a link-local address.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
`{{- IP Source Address is a link-local address.}}` Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8529 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose IP Hop Limit field is set to a value other than 255.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
`{{- The IP Hop Limit field has a value of 255,}}` i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8530 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message that includes an IP Authentication Header. The Redirect message does not authenticate.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
`{{- If the message includes an IP Authentication Header, the message authenticates correctly. }}`
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8531 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose calculated checksum does not match the value in the ICMP Checksum field.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
`{{- ICMP Checksum is valid.}}`
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8532 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose ICMP length as derived from the IP Length field is less than 40 octets.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
`{{- ICMP length (derived from the IP length) is 40 or more octets.}}`
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8533 Redirect Message [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose IP Source Address is not the address of the current first-hop router for the specified ICMP Destination Address.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
`{{- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address. }}`
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8534 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose ICMP Code is set to a value other than 0.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
`{{- ICMP Code is 0. }}`
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8535 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose ICMP Destination Address field contains a multicast field.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
`{{- The ICMP Destination Address field in the redirect message does not contain a multicast address.}}`
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8536 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message redirecting it to a router and whose ICMP Target Address field does not contain a link-local address.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
`{{- The ICMP Target Address is either a link-local address (when redirected to a router)}}` or the same as the ICMP Destination Address (when redirected to the on-link destination).
- All included options have a length that is greater than zero.

RQ_COR_8537 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message redirecting it to an on-link destination and whose ICMP Target Address field is not the same as the ICMP Destination Address.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
`{{- The ICMP Target Address is}}` either a link-local address (when redirected to a router) or `{{the same as the ICMP Destination Address (when redirected to the on-link destination).}}`
- All included options have a length that is greater than zero.

RQ_COR_8538 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message containing an option that has a Length field equal to 0.

Requirement: The implementation silently discards the Redirect message.

RFC text: `{{A host MUST silently discard any received Redirect message that does not satisfy all of the following validity checks:}}`
- IP Source Address is a link-local address. Routers must use their link-local address as the source for Router Advertisement and Redirect messages so that hosts can uniquely identify routers.
- The IP Hop Limit field has a value of 255, i.e., the packet could not possibly have been forwarded by a router.
- If the message includes an IP Authentication Header, the message authenticates correctly.
- ICMP Checksum is valid.
- ICMP Code is 0.
- ICMP length (derived from the IP length) is 40 or more octets.
- The IP source address of the Redirect is the same as the current first-hop router for the specified ICMP Destination Address.
- The ICMP Destination Address field in the redirect message does not contain a multicast address.
- The ICMP Target Address is either a link-local address (when redirected to a router) or the same as the ICMP Destination Address (when redirected to the on-link destination).
`{{- All included options have a length that is greater than zero.}}`

RQ_COR_8539 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message that has its Reserved field set to a value other than 0.

Requirement: The implementation ignores the contents of the Reserved field [and continues processing the rest of the Redirect message].

RFC text: `{{The contents of the Reserved field}}`, and of any unrecognized options `{{MUST be ignored.}}` Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8540 Redirect Message - Option Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message that includes an unrecognizable option.

Requirement: The implementation ignores the unrecognizable option [and continues processing the rest of the Redirect message].

RFC text: `{{The contents}}` of the Reserved field, and `{{of any unrecognized options MUST be ignored.}}` Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

RQ_COR_8541 Redirect Message - Option Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶3 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message that includes a Source Link-layer Address option.

Requirement: The implementation ignores the Source Link-layer Address option and processes the rest of the packet as normal.

RFC text: `{{The contents of any defined options that are not specified to be used with Redirect messages MUST be ignored and the packet processed as normal. The only defined options that may appear are the Target Link-Layer Address option and the Redirected Header option.}}`

RQ_COR_8542 Redirect Message - Option Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶3 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message that includes a Prefix Information option.

Requirement: The implementation ignores the Prefix Information option and processes the rest of the packet as normal.

RFC text: `{{The contents of any defined options that are not specified to be used with Redirect messages MUST be ignored and the packet processed as normal. The only defined options that may appear are the Target Link-Layer Address option and the Redirected Header option.}}`

RQ_COR_8543 Redirect Message - Option Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶3 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message that includes an MTU option.

Requirement: The implementation ignores the MTU option and processes the rest of the packet as normal.

RFC text: `{{The contents of any defined options that are not specified to be used with Redirect messages MUST be ignored and the packet processed as normal. The only defined options that may appear are the Target Link-Layer Address option and the Redirected Header option.}}`

RQ_COR_8544 Redirect Options [Generate]

RFC 2461 *Clause:* 8.1 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect message.

Requirement: Of the possible options, the implementation includes only the Target Link-Layer Address option and the Redirected Header option in the Redirect message.

RFC text: `{{The contents of any defined options that are not specified to be used with Redirect messages MUST be ignored and the packet processed as normal. The only defined options that may appear are the Target Link-Layer Address option and the Redirected Header option.}}`

RQ_COR_8545 Redirect Message - Field Anomalies [Process]

RFC 2461 *Clause:* 8.1 ¶4 *Type:* MUST *applies to:* Host

Context: The implementation receives a Redirect message whose address in the Target Address field is not one of the link's prefixes.

Requirement: The implementation processes the Redirect message. It does not silently discard the redirect.

RFC text: `{{A host MUST NOT consider a redirect invalid just because the Target Address of the redirect is not covered under one of the link's prefixes.}}` Part of the semantics of the Redirect message is that the Target Address is on-link.

RQ_COR_8546 Redirect Message [Generate]

RFC 2461 *Clause:* 8.2 ¶1 *Type:* SHOULD *applies to:* Router

Context: The implementation has received a packet for forwarding whose Source Address field identifies a neighbor. The packet's Destination Address is not a multicast address nor is it the implementation's address. The implementation determines that a better first-hop node resides on the same link as the sending node for the Destination Address of the packet being forwarded.

Requirement: The implementation sends a Redirect message subject to rate limiting.

RFC text:
```
{{A router SHOULD send a redirect message, subject to rate limiting,
whenever it forwards a packet that is not explicitly addressed to
itself (i.e. a packet that is not source routed through the router)
in which:
- the Source Address field of the packet identifies a neighbor, and
- the router determines that a better first-hop node resides on the
same link as the sending node for the Destination Address of the
packet being forwarded, and
- the Destination Address of the packet is not a multicast address}},
```
and

RQ_COR_8547

RFC 2461 *Clause:* 8.2 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect message.

Requirement: The implementation sets the Redirect message's fields to the following values: the Destination Address field is set to the Destination Address of the invoking IP packet; the Target Link-Layer Address field of the Target Link-Layer Address option is set to the target's link-layer address if known; the IP Header + Data field of the Redirected Header is set to as much of the forwarded packet as can fit without the redirect packet exceeding 1280 octets in size.

RFC text:
```
{{The transmitted redirect packet contains, consistent with the
message format given in section 4.5:}}
```
- In the Target Address field: the address to which subsequent packets for the destination SHOULD be sent. If the target is a router, that router's link-local address MUST be used. If the target is a host the target address field MUST be set to the same value as the Destination Address field. `{{`
```
- In the Destination Address field: the destination address of the
invoking IP packet.}} {{
- In the options:
o Target Link-Layer Address option: link-layer address of the target,
if known.
o Redirected Header: as much of the forwarded packet as can fit
without the redirect packet exceeding 1280 octets in size.}}
```

RQ_COR_8548

RFC 2461 *Clause:* 8.2 ¶2 *Type:* SHOULD *applies to:* Router

Context: The implementation is generating a Redirect message.

Requirement: The implementation sets the Redirect's Target Address field to the address to which subsequent packets for the destination are be sent.

RFC text: `{{The transmitted redirect packet contains, consistent with the message format given in section 4.5:`
`- In the Target Address field: the address to which subsequent packets for the destination SHOULD be sent.}}` If the target is a router, that router's link-local address MUST be used. If the target is a host the target address field MUST be set to the same value as the Destination Address field.
- In the Destination Address field: the destination address of the invoking IP packet.
- In the options:
o Target Link-Layer Address option: link-layer address of the target, if known.
o Redirected Header: as much of the forwarded packet as can fit without the redirect packet exceeding 1280 octets in size.

RQ_COR_8549 Redirect Target Address Field Determination

RFC 2461 *Clause:* 8.2 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect message. The target is a router.

Requirement: The implementation sets the Redirect's Target Address field to the target router's link-local address.

RFC text: The transmitted redirect packet contains, consistent with the message format given in section 4.5:
- In the Target Address field: the address to which subsequent packets for the destination SHOULD be sent. `{{If the target is a router, that router's link-local address MUST be used.}}` If the target is a host the target address field MUST be set to the same value as the Destination Address field.
- In the Destination Address field: the destination address of the invoking IP packet.
- In the options:
o Target Link-Layer Address option: link-layer address of the target, if known.
o Redirected Header: as much of the forwarded packet as can fit without the redirect packet exceeding 1280 octets in size.

RQ_COR_8550 Redirect Target Address Field Determination

RFC 2461 *Clause:* 8.2 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect message. The target is a host.

Requirement: The implementation sets the Redirect's Target Address field to the same value as the redirect's Destination Address field.

RFC text: The transmitted redirect packet contains, consistent with the message format given in section 4.5:
- In the Target Address field: the address to which subsequent packets for the destination SHOULD be sent. If the target is a router, that router's link-local address MUST be used. `{{If the target is a host the target address field MUST be set to the same value as the Destination Address field.}}`
- In the Destination Address field: the destination address of the invoking IP packet.
- In the options:
o Target Link-Layer Address option: link-layer address of the target, if known.
o Redirected Header: as much of the forwarded packet as can fit without the redirect packet exceeding 1280 octets in size.

RQ_COR_8551 Redirect Message [Generate]

RFC 2461 *Clause:* 8.2 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation is sending Redirect messages.

Requirement: The implementation limits the rate at which Redirect messages are sent according to [ICMPv6].

RFC text: `{{A router MUST limit the rate at which Redirect messages are sent}}`, in order to limit the bandwidth and processing costs incurred by the Redirect messages when the source does not correctly respond to the Redirects, or the source chooses to ignore unauthenticated Redirect messages. `{{More details on the rate-limiting of ICMP error messages can be found in [ICMPv6].}}`

RQ_COR_8552 Redirect Message [Process]

RFC 2461 *Clause:* 8.2 ¶4 *Type:* MUST *applies to:* Router

Context: The implementation receives a valid Redirect message.

Requirement: The implementation does not update its routing tables.

RFC text: `{{A router MUST NOT update its routing tables upon receipt of a Redirect.}}`

RQ_COR_8553 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Redirect message. The implementation already knows the redirect message's target.

Requirement: For packets whose next hop was changed by the Redirect message, the implementation sends the packets to the specificed target.

RFC text: `{{A host receiving a valid redirect SHOULD update its Destination Cache accordingly so that subsequent traffic goes to the specified target.}}` If no Destination Cache entry exists for the destination, an implementation SHOULD create such an entry.

RQ_COR_8554 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Redirect message. The implementation does not know the redirect message's target is a neighbor.

Requirement: The implementation adds the target to its neighbor list. For packets whose next hop was changed by the Redirect message, the implementation sends the packets to the specificed target.

RFC text: `{{A host receiving a valid redirect SHOULD update its Destination Cache accordingly so that subsequent traffic goes to the specified target. If no Destination Cache entry exists for the destination, an implementation SHOULD create such an entry.}}`

RQ_COR_8555 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶1-2 *Type:* SHOULD *applies to:* Host

Context: The implementation receives a valid Redirect message that includes a Target Link-Layer Address option. The link-layer address in the option is unchanged for this neighbor.

Requirement: For packets whose next hop was changed by the Redirect message, the implementation sends the packets to the specificed target.

RFC text: If the redirect contains a Target Link-Layer Address option the host either creates or updates the Neighbor Cache entry for the target. In both cases the cached link-layer address is copied from the Target Link-Layer Address option. If a Neighbor Cache entry is created for the target its reachability state MUST be set to STALE as specified in section 7.3.3. If a cache entry already existed and it is updated with a different link-layer address, its reachability state MUST also be set to STALE. `{{If the link-layer address is the same as that already in the cache, the cache entry's state remains unchanged.}}`

RQ_COR_8556 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Redirect message that includes a Target Link-Layer Address option. The link-layer address in the option is unchanged for this neighbor.

Requirement: The implementation does not change its link-layer information concerning this neighbor.

RFC text: If the redirect contains a Target Link-Layer Address option the host either creates or updates the Neighbor Cache entry for the target. In both cases the cached link-layer address is copied from the Target Link-Layer Address option. If a Neighbor Cache entry is created for the target its reachability state MUST be set to STALE as specified in section 7.3.3. If a cache entry already existed and it is updated with a different link-layer address, its reachability state MUST also be set to STALE. `{{If the link-layer address is the same as that already in the cache, the cache entry's state remains unchanged.}}`

RQ_COR_8557 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Redirect message that includes a Target Link-Layer Address option. The link-layer address in the option is changed for this neighbor.

Requirement: The implementation updates its link-layer/IP address associations concerning this neighbor and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: If the redirect contains a Target Link-Layer Address option the host either creates or updates the Neighbor Cache entry for the target. In both cases the cached link-layer address is copied from the Target Link-Layer Address option. If a Neighbor Cache entry is created for the target its reachability state MUST be set to STALE as specified in section 7.3.3. `{{If a cache entry already existed and it is updated with a different link-layer address, its reachability state MUST also be set to STALE.}}` If the link-layer address is the same as that already in the cache, the cache entry's state remains unchanged.

RQ_COR_8558 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Redirect message that includes a Target Link-Layer Address option. The link-layer address in the option is unknown.

Requirement: The implementation adds this neighbor and the link-layer/IP address associations to its neighbor list and waits ReachableTime milliseconds for reachability confirmation with the new neighbor.

RFC text: If the redirect contains a Target Link-Layer Address option the host either creates or updates the Neighbor Cache entry for the target. In both cases the cached link-layer address is copied from the Target Link-Layer Address option. `{{If a Neighbor Cache entry is created for the target its reachability state MUST be set to STALE as specified in section 7.3.3.}}` If a cache entry already existed and it is updated with a different link-layer address, its reachability state MUST also be set to STALE. If the link-layer address is the same as that already in the cache, the cache entry's state remains unchanged.

RQ_COR_8559 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶3 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Redirect message in which the Target and Destination Addresses are the same. The target is a known neighbor.

Requirement: The implementation treats the Target as on-link. The neighbor's status as a router or host remains unchanged.

RFC text: `{{If the Target and Destination Addresses are the same, the host MUST treat the Target as on-link.}}` If the Target Address is not the same as the Destination Address, the host MUST set IsRouter to TRUE for the target. If the Target and Destination Addresses are the same, however, one cannot reliably determine whether the Target Address is a router. Consequently, newly created Neighbor Cache entries should set the IsRouter flag to FALSE, `{{while existing cache entries should leave the flag unchanged.}}` If the Target is a router, subsequent Neighbor Advertisement or Router Advertisement messages will update IsRouter accordingly.

RQ_COR_8560 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶3 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Redirect message in which the Target and Destination Addresses are different.

Requirement: The implementation treats the Target as a router.

RFC text: `{{If the Target and Destination Addresses are the same, the host MUST treat the Target as on-link.}}` `{{If the Target Address is not the same as the Destination Address, the host MUST set IsRouter to TRUE for the target.}}` If the Target and Destination Addresses are the same, however, one cannot reliably determine whether the Target Address is a router. Consequently, newly created Neighbor Cache entries should set the IsRouter flag to FALSE, while existing cache entries should leave the flag unchanged. If the Target is a router, subsequent Neighbor Advertisement or Router Advertisement messages will update IsRouter accordingly.

RQ_COR_8561 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶3 *Type:* MUST *applies to:* Host

Context: The implementation receives a valid Redirect message in which the Target and Destination Addresses are the same. The target is unknown.

Requirement: The implementation adds the new neighbor to its neighbor list treats the Target as on-link. The Target is treated as a host.

RFC text: `{{<If the Target and Destination Addresses are the same, the host MUST treat the Target as on-link.}}` If the Target Address is not the same as the Destination Address, the host MUST set IsRouter to TRUE for the target. If the Target and Destination Addresses are the same, however, one cannot reliably determine whether the Target Address is a router. Consequently, `{{newly created Neighbor Cache entries should set the IsRouter flag to FALSE,}}` while existing cache entries should leave the flag unchanged. If the Target is a router, subsequent Neighbor Advertisement or Router Advertisement messages will update IsRouter accordingly.

RQ_COR_8562

RFC 2461 *Clause:* 8.3 ¶4 *Type:* MUST *applies to:* Host

Context: Different types of flows are being sent to a given destination. The implementation receives a valid Redirect message to change the first-hop for the given destination..

Requirement: The implementation sends all packets for the given destination to the Redirect's specified next hop regardless of the Flow Label field in the Redirected Header option's contents.

RFC text: `{{Redirect messages apply to all flows that are being sent to a given destination. That is, upon receipt of a Redirect for a Destination Address, all Destination Cache entries to that address should be updated to use the specified next-hop, regardless of the contents of the Flow Label field that appears in the Redirected Header option.}}`

RQ_COR_8563 Redirect Message [Process]

RFC 2461 *Clause:* 8.3 ¶5 *Type:* MAY *applies to:* Host

Context: The implementation has been configured by system management.

Requirement: The implementation has a configuration switch to allow the implementation to ignore a Redirect message that does not have an IP Authentication header.

RFC text: `{{A host MAY have a configuration switch that can be set to make it ignore a Redirect message that does not have an IP Authentication header.}}`

RQ_COR_8564 Redirect Message [Generate]

RFC 2461 *Clause:* 8.3 ¶6 *Type:* MUST *applies to:* Host

Context: The implementation performs Neighbor Discovery

Requirement: The implementation does not send Redirect messages.

RFC text: `{{A host MUST NOT send Redirect messages.}}`

RQ_COR_8565 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Discovery packet that contains an unrecognizable option.

Requirement: The implementation ignores the unrecognizable option and continues processing the packet.

RFC text: `{{In order to ensure that future extensions properly coexist with current implementations, all nodes MUST silently ignore any options they do not recognize in received ND packets and continue processing the packet.}}` All options specified in this document MUST be recognized. A node MUST NOT ignore valid options just because the ND message contains unrecognized ones.

RQ_COR_8566 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Discovery packet.

Requirement: The implementation recognizes the Source Link-layer Address option, Target Link-layer Address option, Prefix Information option, Redirected Header option, and the MTU option when the option is in the Neighbor Discovery packet.

RFC text: In order to ensure that future extensions properly coexist with current implementations, all nodes MUST silently ignore any options they do not recognize in received ND packets and continue processing the packet. `{{All options specified in this document MUST be recognized.}}` A node MUST NOT ignore valid options just because the ND message contains unrecognized ones.

RQ_COR_8567 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Discovery packet that includes both valid options and other unrecognizable options.

Requirement: The implementation does not ignore the valid options and ignores the unrecognizable options.

RFC text: `{{In order to ensure that future extensions properly coexist with current implementations, all nodes MUST silently ignore any options they do not recognize in received ND packets}}` and continue processing the packet. All options specified in this document MUST be recognized. `{{A node MUST NOT ignore valid options just because the ND message contains unrecognized ones.}}`

RQ_COR_8568 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation uses an option other than the the Source Link-layer Address, Target Link-layer Address, Prefix Information, Redirected Header, and the MTU options in a Neighbor Discovery packet.

Requirement: The semantics of the option depend only on the information in the fixed part of the Neighbor Discovery packet and on the information contained in the option itself.

RFC text: The option MUST NOT depend on the presence or absence of any other options. `{{The semantics of an option should depend only on the information in the fixed part of the ND packet and on the information contained in the option itself.}}`

RQ_COR_8569 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation uses an option other than the the Source Link-layer Address, Target Link-layer Address, Prefix Information, Redirected Header, and the MTU options in a Neighbor Discovery packet.

Requirement: The option does not depend on the presence or absence of any other options.

RFC text: `{{The option MUST NOT depend on the presence or absence of any other options.}}` The semantics of an option should depend only on the information in the fixed part of the ND packet and on the information contained in the option itself.

RQ_COR_8570 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶4 sub¶3 *Type:* MAY *applies to:* Node

Context: The implementation is generating a Neighbor Discovery packet that contains several options.

Requirement: The implementation splits the set of options into subsets and sends the subsets in different Neighbor Discovery packets.

RFC text: 3) `{{Senders MAY send a subset of options in different packets.}}` For instance, if a prefix's Valid and Preferred Lifetime are high enough, it might not be necessary to include the Prefix Information option in every Router Advertisement. In addition, different routers might send different sets of options. Thus, a receiver MUST NOT associate any action with the absence of an option in a particular packet. This protocol specifies that receivers should only act on the expiration of timers and on the information that is received in the packets.

RQ_COR_8571 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶4 sub¶3 *Type:* MAY *applies to:* Node

Context: The implementation receives a Neighbor Discovery packet that does not have an option normally associated with the packet.

Requirement: The implementation does not take any action with the absence of an option in a particular packet.

RFC text: 3) Senders MAY send a subset of options in different packets. For instance, if a prefix's Valid and Preferred Lifetime are high enough, it might not be necessary to include the Prefix Information option in every Router Advertisement. In addition, different routers might send different sets of options. `{{Thus, a receiver MUST NOT associate any action with the absence of an option in a particular packet. This protocol specifies that receivers should only act on the expiration of timers and on the information that is received in the packets.}}`

RQ_COR_8572 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation receives valid Neighbor Discovery packets containing more than one option.

Requirement: The implementation process the options independently of their order.

RFC text: `{{Options in Neighbor Discovery packets can appear in any order; receivers MUST be prepared to process them independently of their order.}}` There can also be multiple instances of the same option in a message (e.g., Prefix Information options).

RQ_COR_8573 Neighbor Discovery Messages Options

RFC 2461 *Clause:* 9 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation receives valid Neighbor Discovery packets containing multiple instances of the same option.

Requirement: The implementation processes the multiple instances.

RFC text: Options in Neighbor Discovery packets can appear in any order; receivers MUST be prepared to process them independently of their order. `{{There can also be multiple instances of the same option in a message (e.g., Prefix Information options).}}`

RQ_COR_8574 Router Advertisement [Process]

RFC 2461 *Clause:* 9 ¶6 *Type:* MUST *applies to:* Router

Context: The implementation is generating a Router Advertisement where the number of total options causes causes the advertisement's size to exceed the link MTU.

Requirement: The implementation sends multiple separate advertisements each containing a subset of the options.

RFC text: `{{If the number of included options in a Router Advertisement causes the advertisement's size to exceed the link MTU, the router can send multiple separate advertisements each containing a subset of the options.}}`

RQ_COR_8575 Redirect Options [Generate]

RFC 2461 *Clause:* 9 ¶7 *Type:* MUST *applies to:* Router

Context: The implementation is generating a Redirect packet.

Requirement: The implementation limits the amount of data to include in the Redirected Header option so that the entire redirect packet does not exceed 1280 octets.

RFC text: `{{The amount of data to include in the Redirected Header option MUST be limited so that the entire redirect packet does not exceed 1280 octets.}}`

RQ_COR_8576 Neighbor Discovery Messages [Generate]

RFC 2461 *Clause:* 9 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation is generating a Neighbor Discovery packet with or without options.

Requirement: The implementation limits the size of the Neighbor Discovery packet to the link MTU.

RFC text: `{{The size of an ND packet including the IP header is limited to the link MTU (which is at least 1280 octets). When adding options to an ND packet a node MUST NOT exceed the link MTU.}}`

RQ_COR_8577 Neighbor Discovery Protocol Constants and

RFC 2461 *Clause:* 10 "Routers" *Type:* MUST *applies to:* Router

Context: The implementation is functioning.

Requirement: The implementation uses the following protocol constants unless overridden by specific documents that describe how IPv6 operates over different link layers: MAX_INITIAL_RTR_ADVERT_INTERVAL - 16 seconds; MAX_INITIAL_RTR_ADVERTISEMENTS - 3 transmissions; MAX_FINAL_RTR_ADVERTISEMENTS - 3 transmissions; MIN_DELAY_BETWEEN_RAS - 3 seconds; MAX_RA_DELAY_TIME - .5 seconds

RFC text:
```
{{Router constants:
MAX_INITIAL_RTR_ADVERT_INTERVAL   16 seconds
MAX_INITIAL_RTR_ADVERTISEMENTS    3 transmissions
MAX_FINAL_RTR_ADVERTISEMENTS      3 transmissions
MIN_DELAY_BETWEEN_RAS             3 seconds
MAX_RA_DELAY_TIME                 .5 seconds}}
```
Host constants:
MAX_RTR_SOLICITATION_DELAY 1 second
RTR_SOLICITATION_INTERVAL 4 seconds
MAX_RTR_SOLICITATIONS 3 transmissions
Node constants:
MAX_MULTICAST_SOLICIT 3 transmissions
MAX_UNICAST_SOLICIT 3 transmissions
MAX_ANYCAST_DELAY_TIME 1 second
MAX_NEIGHBOR_ADVERTISEMENT 3 transmissions
REACHABLE_TIME 30,000 milliseconds
RETRANS_TIMER 1,000 milliseconds
DELAY_FIRST_PROBE_TIME 5 seconds
MIN_RANDOM_FACTOR .5
MAX_RANDOM_FACTOR 1.5

RQ_COR_8578 Neighbor Discovery Protocol Constants and

RFC 2461 *Clause:* 10 "Hosts" and *Type:* MUST *applies to:* Host

Context: The implementation is functioning.

Requirement: The implementation uses the following protocol constants unless overridden by specific documents that describe how IPv6 operates over different link layers: MAX_RTR_SOLICITATION_DELAY - 1 second; RTR_SOLICITATION_INTERVAL - 4 seconds; MAX_RTR_SOLICITATIONS - 3 transmissions

RFC text: Router constants:
MAX_INITIAL_RTR_ADVERT_INTERVAL 16 seconds
MAX_INITIAL_RTR_ADVERTISEMENTS 3 transmissions
MAX_FINAL_RTR_ADVERTISEMENTS 3 transmissions
MIN_DELAY_BETWEEN_RAS 3 seconds
MAX_RA_DELAY_TIME .5 seconds
```
{{Host constants:
MAX_RTR_SOLICITATION_DELAY        1 second
RTR_SOLICITATION_INTERVAL         4 seconds
MAX_RTR_SOLICITATIONS             3 transmissions}}
```
Node constants:
MAX_MULTICAST_SOLICIT 3 transmissions
MAX_UNICAST_SOLICIT 3 transmissions
MAX_ANYCAST_DELAY_TIME 1 second
MAX_NEIGHBOR_ADVERTISEMENT 3 transmissions
REACHABLE_TIME 30,000 milliseconds
RETRANS_TIMER 1,000 milliseconds
DELAY_FIRST_PROBE_TIME 5 seconds
MIN_RANDOM_FACTOR .5
MAX_RANDOM_FACTOR 1.5

RQ_COR_8579 Neighbor Discovery Protocol Constants and

RFC 2461 *Clause:* 10 "Nodes" and *Type:* MUST *applies to:* Node

Context: The implementation is functioning.

Requirement: The implementation uses the following protocol constants unless overridden by specific documents that describe how IPv6 operates over different link layers: Node constants: MAX_MULTICAST_SOLICIT - 3; transmissions; MAX_UNICAST_SOLICIT - 3 transmissions; MAX_ANYCAST_DELAY_TIME - 1 second; MAX_NEIGHBOR_ADVERTISEMENT - 3 transmissions; REACHABLE_TIME - 30,000 milliseconds; RETRANS_TIMER - 1,000 milliseconds; DELAY_FIRST_PROBE_TIME - 5 seconds; MIN_RANDOM_FACTOR - .5; >MAX_RANDOM_FACTOR - 1.5

RFC text: Router constants:

MAX_INITIAL_RTR_ADVERT_INTERVAL 16 seconds
MAX_INITIAL_RTR_ADVERTISEMENTS 3 transmissions
MAX_FINAL_RTR_ADVERTISEMENTS 3 transmissions
MIN_DELAY_BETWEEN_RAS 3 seconds
MAX_RA_DELAY_TIME .5 seconds

Host constants:

MAX_RTR_SOLICITATION_DELAY 1 second
RTR_SOLICITATION_INTERVAL 4 seconds
MAX_RTR_SOLICITATIONS 3 transmissions

```
{{Node constants:
MAX_MULTICAST_SOLICIT           3 transmissions
MAX_UNICAST_SOLICIT             3 transmissions
MAX_ANYCAST_DELAY_TIME          1 second
MAX_NEIGHBOR_ADVERTISEMENT      3 transmissions
REACHABLE_TIME                 30,000 milliseconds
RETRANS_TIMER                   1,000 milliseconds
DELAY_FIRST_PROBE_TIME          5 seconds
MIN_RANDOM_FACTOR              .5
MAX_RANDOM_FACTOR               1.5}}
```

RQ_COR_8580 Redirect Message [Process]

RFC 2461 *Clause:* 11 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation receives a Redirect message from a router other than the router currently being used as the destination's first hop.

Requirement: The implementation silently ignores the Redirect message.

RFC text: The trust model for redirects is the same as in IPv4. `{{A redirect is accepted only if received from the same router that is currently being used for that destination.}}` It is natural to trust the routers on the link. If a host has been redirected to another node (i.e., the destination is on-link) there is no way to prevent the target from issuing another redirect to some other destination. However, this exposure is no worse than it was; the target host, once subverted, could always act as a hidden router to forward traffic elsewhere.

RQ_COR_8581 Neighbor Discovery Messages [Generate]

RFC 2461 *Clause:* 11 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation has a security association for the Destination Address of a Neighbor Discovery packet.

Requirement: The implementation includes a valid IP Authentication Header in the Neighbor Discovery packets sent to the Destination Address for which the security association exists.

RFC text: Neighbor Discovery protocol packet exchanges can be authenticated using the IP Authentication Header [IPv6-AUTH]. `{{A node SHOULD include an Authentication Header when sending Neighbor Discovery packets if a security association for use with the IP Authentication Header exists for the destination address.}}` The security associations may have been created through manual configuration or through the operation of some key management protocol.

RQ_COR_8582 Neighbor Discovery Messages [Process]

RFC 2461 *Clause:* 11 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Discovery packet that includes an IP Authentication Header.

Requirement: The implementation verifies the Authentication Header for correctness.

RFC text: `{{Received Authentication Headers in Neighbor Discovery packets MUST be verified for correctness}}` and packets with incorrect authentication MUST be ignored.

RQ_COR_8583 Neighbor Discovery Messages [Process]

RFC 2461 *Clause:* 11 ¶8 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighbor Discovery packet that includes an IP Authentication Header. The implementation verifies the Authentication Header for correctness. The packet incorrectly authenticates.

Requirement: The implementation ignores the Neighbor Discovery packet.

RFC text: `{{Received Authentication Headers in Neighbor Discovery packets MUST be verified for correctness}}` and packets with incorrect authentication MUST be ignored.

RQ_COR_8584 Neighbor Discovery Messages [Process]

RFC 2461 *Clause:* 11 ¶9 *Type:* SHOULD *applies to:* Node

Context: The implementation is being configured by the system administrator.

Requirement: The implementation is capable via a switch to ignore any Neighbor Discovery messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload.

RFC text: `{{It SHOULD be possible for the system administrator to configure a node to ignore any Neighbor Discovery messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload.}}` The configuration technique for this MUST be documented. Such a switch SHOULD default to allowing unauthenticated messages.

RQ_COR_8585 Neighbor Discovery Messages [Process]

RFC 2461 *Clause:* 11 ¶9 *Type:* SHOULD *applies to:* Node

Context: The implementation is capable via a switch to ignore any Neighbor Discovery messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload. The switch is left in its default value.

Requirement: The implementation processes unauthenticated Neighbor Discovery messages.

RFC text: It SHOULD be possible for the system administrator to configure a node to ignore any Neighbor Discovery messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload. The configuration technique for this MUST be documented. `{{Such a switch SHOULD default to allowing unauthenticated messages.}}`

RQ_COR_8586 Router Solicitation [Process]

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Router Solicitation message.

Requirement: The implementation assumes that the solicitation's sender is a host.

RFC text: The background for these rules is that the ND messages contain, either implicitly or explicitly, information that indicates whether or not the sender (or Target Address) is a host or a router. `{{The following assumptions are used:}}`
`{{- The sender of a Router Solicitation is implicitly assumed to be a host since there is no need for routers to send such messages.}}`
- The sender of a Router Advertisement is implicitly assumed to be a router.
- Neighbor Solicitation messages do not contain either an implicit or explicit indication about the sender. Both hosts and routers send such messages.
- Neighbor Advertisement messages contain an explicit "IsRouter flag", the R-bit.
- The target of the redirect, when the target differs from the destination address in the packet being redirected, is implicitly assumed to be a router. This is a natural assumption since that node is expected to be able to forward the packets towards the destination.
- The target of the redirect, when the target is the same as the destination, does not carry any host vs. router information. All that is known is that the destination (i.e. target) is on-link but it could be either a host or a router.

RQ_COR_8587 Router Advertisement [Process]

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Router Advertisement message.

Requirement: The implementation assumes that the advertisement's sender is a router.

RFC text: The background for these rules is that the ND messages contain, either implicitly or explicitly, information that indicates whether or not the sender (or Target Address) is a host or a router. `{{The following assumptions are used:}}`
- The sender of a Router Solicitation is implicitly assumed to be a host since there is no need for routers to send such messages.
`{{- The sender of a Router Advertisement is implicitly assumed to be a router.}}`
- Neighbor Solicitation messages do not contain either an implicit or explicit indication about the sender. Both hosts and routers send such messages.
- Neighbor Advertisement messages contain an explicit "IsRouter flag", the R-bit.
- The target of the redirect, when the target differs from the destination address in the packet being redirected, is implicitly assumed to be a router. This is a natural assumption since that node is expected to be able to forward the packets towards the destination.
- The target of the redirect, when the target is the same as the destination, does not carry any host vs. router information. All that is known is that the destination (i.e. target) is on-link but it could be either a host or a router.

RQ_COR_8588

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Redirect message. The redirect's target differs from the Destination Address of the redirected packet.

Requirement: The implementation assumes that the target of the redirect is a router.

RFC text: The background for these rules is that the ND messages contain, either implicitly or explicitly, information that indicates whether or not the sender (or Target Address) is a host or a router. `{{The following assumptions are used:}}`
- The sender of a Router Solicitation is implicitly assumed to be a host since there is no need for routers to send such messages.
- The sender of a Router Advertisement is implicitly assumed to be a router.
- Neighbor Solicitation messages do not contain either an implicit or explicit indication about the sender. Both hosts and routers send such messages.
- Neighbor Advertisement messages contain an explicit "IsRouter flag", the R-bit.
`{{- The target of the redirect, when the target differs from the destination address in the packet being redirected, is implicitly assumed to be a router. This is a natural assumption since that node is expected to be able to forward the packets towards the destination.}}`
- The target of the redirect, when the target is the same as the destination, does not carry any host vs. router information. All that is known is that the destination (i.e. target) is on-link but it could be either a host or a router.

RQ_COR_8589 Redirect Message [Process]

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Redirect message. The redirect's target is the same as the Destination Address of the redirected packet.

Requirement: The implementation does not assume that the target of the redirect is a router or a host.

RFC text: The background for these rules is that the ND messages contain, either implicitly or explicitly, information that indicates whether or not the sender (or Target Address) is a host or a router. `{{The following assumptions are used:}}`
- The sender of a Router Solicitation is implicitly assumed to be a host since there is no need for routers to send such messages.
- The sender of a Router Advertisement is implicitly assumed to be a router.
- Neighbor Solicitation messages do not contain either an implicit or explicit indication about the sender. Both hosts and routers send such messages.
- Neighbor Advertisement messages contain an explicit "IsRouter flag", the R-bit.
- The target of the redirect, when the target differs from the destination address in the packet being redirected, is implicitly assumed to be a router. This is a natural assumption since that node is expected to be able to forward the packets towards the destination.
`{{- The target of the redirect, when the target is the same as the destination, does not carry any host vs. router information. All that is known is that the destination (i.e. target) is on-link but it could be either a host or a router.}}`

RQ_COR_8590 Neighbor Solicitation [Process]

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a valid Neighbor Solicitation message.

Requirement: The implementation does not assume that the target of the solicitation is a router or a host.

RFC text: The background for these rules is that the ND messages contain, either implicitly or explicitly, information that indicates whether or not the sender (or Target Address) is a host or a router. `{{The following assumptions are used:}}`
- The sender of a Router Solicitation is implicitly assumed to be a host since there is no need for routers to send such messages.
- The sender of a Router Advertisement is implicitly assumed to be a router.
`{{- Neighbor Solicitation messages do not contain either an implicit or explicit indication about the sender. Both hosts and routers send such messages.}}`
- Neighbor Advertisement messages contain an explicit "IsRouter flag", the R-bit.
- The target of the redirect, when the target differs from the destination address in the packet being redirected, is implicitly assumed to be a router. This is a natural assumption since that node is expected to be able to forward the packets towards the destination.
- The target of the redirect, when the target is the same as the destination, does not carry any host vs. router information. All that is known is that the destination (i.e. target) is on-link but it could be either a host or a router.

RQ_COR_8591 Neighbor Discovery Messages [Process]

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighber Discovery message containing explicit information concerning the sender being a host or a router.

Requirement: The implementation updates the sender's status to a host or rater based on the explicit information.

RFC text: The rules for setting the IsRouter flag are based on the information content above. `{{If an ND message contains explicit or implicit information the receipt of the message will cause the IsRouter flag to be updated.}}` But when there is no host vs. router information in the ND message the receipt of the message MUST NOT cause a change to the IsRouter state. When the receipt of such a message causes a Neighbor Cache entry to be created this document specifies that the IsRouter flag be set to FALSE. There is greater potential for mischief when a node incorrectly thinks a host is a router, than the other way around. In these cases a subsequent Neighbor Advertisement or Router Advertisement message will set the correct IsRouter value.

RQ_COR_8592 Neighbor Discovery Messages [Process]

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighber Discovery message containing implicit information concerning the sender being a host or a router.

Requirement: The implementation updates the sender's status to a host or rater based on the implicit information.

RFC text: The rules for setting the IsRouter flag are based on the information content above. `{{If an ND message contains explicit or implicit information the receipt of the message will cause the IsRouter flag to be updated.}}` But when there is no host vs. router information in the ND message the receipt of the message MUST NOT cause a change to the IsRouter state. When the receipt of such a message causes a Neighbor Cache entry to be created this document specifies that the IsRouter flag be set to FALSE. There is greater potential for mischief when a node incorrectly thinks a host is a router, than the other way around. In these cases a subsequent Neighbor Advertisement or Router Advertisement message will set the correct IsRouter value.

RQ_COR_8593 Neighbor Discovery Messages [Process]

RFC 2461 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a Neighber Discovery message containing no explicit nor implicit information concerning the sender being a host or a router.

Requirement: The implementation does not change the sender's status of being either a router or a host.

RFC text: The rules for setting the IsRouter flag are based on the information content above. If an ND message contains explicit or implicit information the receipt of the message will cause the IsRouter flag to be updated. `{{But when there is no host vs. router information in the ND message the receipt of the message MUST NOT cause a change to the IsRouter state. }}` When the receipt of such a message causes a Neighbor Cache entry to be created this document specifies that the IsRouter flag be set to FALSE. There is greater potential for mischief when a node incorrectly thinks a host is a router, than the other way around. In these cases a subsequent Neighbor Advertisement or Router Advertisement message will set the correct IsRouter value.

RQ_COR_8594 Address Resolution Behavior

RFC 2461 *Clause:* 7.2.5 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is on a link having addresses and the neighbor's link-layer address is known (i.e. in any other state than INCOMPLETE). The implementation considers the neighbor not to be reachable. It then receives a valid Neighbor Advertisement for Address Resolution. The advertisement includes the Target Link-Layer address option with a link-layer address the same as the one currently associated to the neighbor. The advertisement's Override flag is set to one and its Solicited flag is set to zero.

Requirement: The implementation does not change its information concerning the neighbor.

RFC text: `{{If the target's Neighbor Cache entry is in any state other than INCOMPLETE when the advertisement is received, processing becomes quite a bit more complex.}}` If the Override flag is clear and the supplied link-layer address differs from that in the cache, then one of two actions takes place: if the state of the entry is REACHABLE, set it to STALE, but do not update the entry in any other way; otherwise, the received advertisement should be ignored and MUST NOT update the cache. `{{If the Override flag is set}}`, both the Override flag is clear and the supplied link-layer address is the same as that in the cache, or no Target Link-layer address option was supplied, the received advertisement MUST update the Neighbor Cache entry as follows:
- The link-layer address in the Target Link-Layer Address option MUST be inserted in the cache (if one is supplied and is different than the already recorded address).
- If the Solicited flag is set, the state of the entry MUST be set to REACHABLE. `{{If the Solicited flag is zero and the link-layer address was updated with a different address the state MUST be set to STALE. Otherwise, the entry's state remains unchanged.}}`

RQ_COR_9033 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to prohibit the implementation from both sending periodic Router Advertisements and responding to Router Solicitations. This flag is left at its default value.

Requirement: The implementation does not respond to Router Solicitations.

RFC text: A router MUST allow for the following conceptual variables to be configured by system management.
...
AdvSendAdvertisements
A flag indicating whether or not the router sends periodic Router Advertisements and responds to Router Solicitations.
`{{Default: FALSE`
`Note that AdvSendAdvertisements MUST be FALSE by default so that a node will not accidentally start acting as a router unless it is explicitly configured by system management to send Router Advertisements}}`.

RQ_COR_9034 Router Solicitation [Process]

RFC 2461 *Clause:* 6.2.1 ¶1 and *Type:* MUST *applies to:* Router

Context: The implementation is being configured for operation. System management provides for each implementation's multicast interface a flag to prohibit the implementation from both sending periodic Router Advertisements and responding to Router Solicitations. This flag is set to allow both periodic generation of Router Advertisements and responding to Router Solicitations.

Requirement: The implementation responds to Router Solicitations.

RFC text: A router MUST allow for the following conceptual variables to be configured by system management.
...
AdvSendAdvertisements
A flag indicating whether or not the router sends periodic Router Advertisements and responds to Router Solicitations.
`{{Default: FALSE`
`Note that AdvSendAdvertisements MUST be FALSE by default so that a node will not accidentally start acting as a router unless it is explicitly configured by system management to send Router Advertisements}}`.

4.8 Requirements extracted from RFC 2462

RQ_COR_1200 Stateless and Stateful Autoconfig Simultaneous

RFC 2462 *Clause:* 1 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation uses both Stateful and Stateless Autoconfiguration. The implementation receives a Router Advertisment message requiring simultaneous stateful and stateless autoconfiguration.

Requirement: The implementation simultaneously executes both stateful and stateless autoconfiguration.

RFC text: The stateless approach is used when a site is not particularly concerned with the exact addresses hosts use, so long as they are unique and properly routable. The stateful approach is used when a site requires tighter control over exact address assignments. `{{Both stateful and stateless address autoconfiguration may be used simultaneously}}`. The site administrator specifies which type of autoconfiguration to use through the setting of appropriate fields in Router Advertisement messages [DISCOVERY].

RQ_COR_1202 Address Lifetime

RFC 2462 *Clause:* 1 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation receives a leased autoconfigured address for a fixed lifetime. The lifetime expires.

Requirement: The implementation's binding and address are invalid.

RFC text: `{{When a lifetime expires, the binding (and address) become invalid}}`.

RQ_COR_1204 address: Preferred Address

RFC 2462 *Clause:* 1 ¶5,2 ¶20 *Type:* MUST *applies to:* Node

Context: The implementation's address state is "preferred".

Requirement: The implementation uses the address as the source address of packets sent from the address's interface.

RFC text: To handle the expiration of address bindings gracefully, an address goes through two distinct phases while assigned to an interface. `{{Initially, an address is "preferred", meaning that its use in arbitrary communication is unrestricted}}`. Later, an address becomes "deprecated" in anticipation that its current interface binding will become invalid. While in a deprecated state, the use of an address is discouraged, but not strictly forbidden. `{{...The implementation uses the preferred address as the source (or destination) address of packets sent from (or to) the address' interface}}`.

RQ_COR_1205 address: Preferred Address

RFC 2462 *Clause:* 1 ¶5,2 ¶20 *Type:* MUST *applies to:* Node

Context: The implementation's address state is "preferred".

Requirement: The implementation uses the address as the destination address of packets sent to the address's interface.

RFC text: To handle the expiration of address bindings gracefully, an address goes through two distinct phases while assigned to an interface. `{{Initially, an address is "preferred", meaning that its use in arbitrary communication is unrestricted}}`. Later, an address becomes "deprecated" in anticipation that its current interface binding will become invalid. While in a deprecated state, the use of an address is discouraged, but not strictly forbidden. `{{...The implementation uses the preferred address as the source (or destination) address of packets sent from (or to) the address' interface}}`.

RQ_COR_1207 address: Deprecated Address Use

RFC 2462 *Clause:* 1 ¶5 *Type:* SHOULD *applies to:* Node

Context: The implementation begins new communication (e.g., the opening of a new TCP connection).

Requirement: The implementation uses its preferred address.

RFC text: `{{While in a deprecated state, the use of an address is discouraged, but not strictly forbidden. New communication (e.g., the opening of a new TCP connection) should use a preferred address when possible. A deprecated address should be used only by applications that have been using it and would have difficulty switching to another address without a service disruption}}`.

RQ_COR_1209 address: Deprecated Address Use

RFC 2462 *Clause:* 1 ¶5 *Type:* SHOULD *applies to:* Node

Context: An implementation's application cannot switch to another address without service disruption. The implementation's address state is deprecated.

Requirement: The implementation continues to use the deprecated address.

RFC text: `{{While in a deprecated state, the use of an address is discouraged, but not strictly forbidden. ...A deprecated address should be used only by applications that have been using it and would have difficulty switching to another address without a service disruption}}`.

RQ_COR_1210 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 2 ¶19,5.4 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation's address is tentative.

Requirement: The implementation accepts Neighbor Discovery packets related to Duplicate Address Detection for the tentative address and discards all other packets addressed to its tentative address.

RFC text: `{{tentative address - an address whose uniqueness on a link is being verified, prior to its assignment to an interface. A tentative address is not considered assigned to an interface in the usual sense. An interface discards received packets addressed to a tentative address, but accepts Neighbor Discovery packets related to Duplicate Address Detection for the tentative address}}`. `{{...An address on which the duplicate Address Detection Procedure is applied is said to be tentative until the procedure has completed successfully. A tentative address is not considered "assigned to an interface" in the traditional sense. That is, the interface must accept Neighbor Solicitation and Advertisement messages containing the tentative address in the Target Address field, but processes such packets differently from those whose Target Address matches an address assigned to the interface. Other packets addressed to the tentative address should be silently discarded}}`.

RQ_COR_1211 address: Deprecated Address Use

RFC 2462 *Clause:* 2 ¶21 *Type:* SHOULD *applies to:* Node

Context: The implementation has a "deprecated" address.

Requirement: The implementation does not use the "deprecated" address as a source address in new communications.

RFC text: deprecated address - An address assigned to an interface whose use is discouraged, but not forbidden. `{{A deprecated address should no longer be used as a source address in new communications}}`, but packets sent from or to deprecated addresses are delivered as expected.

RQ_COR_1212 address: Deprecated Address Use

RFC 2462 *Clause:* 2 ¶21 *Type:* MUST *applies to:* Node

Context: An address is deprecated.

Requirement: The implementation delivers the packets sent from the deprecated address.

RFC text: deprecated address - An address assigned to an interface whose use is discouraged, but not forbidden. A deprecated address should no longer be used as a source address in new communications, `{{but packets sent from or to deprecated addresses are delivered as expected}}`.

RQ_COR_1213 address: Invalid

RFC 2462 *Clause:* 2 ¶23 *Type:* SHOULD *applies to:* Node

Context: An address is invalid.

Requirement: The implementation does not generate packets with an invalid destination address.

RFC text: `{{invalid address - an address that is not assigned to any interface. A valid address becomes invalid when its valid lifetime expires. Invalid addresses should not appear as the destination or source address of a packet}}`. In the former case, the internet routing system will be unable to deliver the packet, in the later case the recipient of the packet will be unable to respond to it.

RQ_COR_1214 address: Invalid

RFC 2462 *Clause:* 2 ¶23 *Type:* SHOULD *applies to:* Node

Context: An address is invalid.

Requirement: The implementation does not generate packets with an invalid source address.

RFC text: `{{invalid address - an address that is not assigned to any interface. A valid address becomes invalid when its valid lifetime expires. Invalid addresses should not appear as the destination or source address of a packet}}`. In the former case, the internet routing system will be unable to deliver the packet, in the later case the recipient of the packet will be unable to respond to it.

RQ_COR_1215 address: Preferred Address

RFC 2462 *Clause:* 2 ¶24 *Type:* MUST *applies to:* Node

Context: The implementation is assigned a "preferred" address with a "preferred" lifetime.

Requirement: The implementation uses the preferred address for the lifetime starting from the time of assignment.

RFC text: `{{preferred lifetime - the length of time that a valid address is preferred (i.e., the time until deprecation). When the preferred lifetime expires, the address becomes deprecated}}`.

RQ_COR_1216 Address Lifetime

RFC 2462 *Clause:* 2 ¶24 *Type:* MUST *applies to:* Node

Context: The implementation is assigned a "preferred" address with a "preferred" lifetime. The preferred lifetime expires.

Requirement: The implementation deprecates the address.

RFC text: `{{preferred lifetime - the length of time that a valid address is preferred (i.e., the time until deprecation). When the preferred lifetime expires, the address becomes deprecated}}`.

RQ_COR_1217 address: Valid Address Use

RFC 2462 *Clause:* 2 ¶25 *Type:* MUST *applies to:* Node

Context: The implementation is assigned a "valid" address with a "valid" lifetime.

Requirement: The implementation uses the valid address for the valid lifetime starting from the time of assignment.

RFC text: `{{valid lifetime - the length of time an address remains in the valid state (i.e., the time until invalidation). The valid lifetime must be greater then or equal to the preferred lifetime. When the valid lifetime expires, the address becomes invalid}}`.

RQ_COR_1218 address: Valid Address Use

RFC 2462 *Clause:* 2 ¶25 *Type:* MUST *applies to:* Node

Context: The implementation is assigned a "valid" address with a "valid" lifetime. The "valid" lifetime expires.

Requirement: The implementation's address is invalid.

RFC text: valid lifetime - the length of time an address remains in the valid state (i.e., the time until invalidation). The valid lifetime must be greater then or equal to the preferred lifetime. `{{When the valid lifetime expires, the address becomes invalid}}`. See RQ_COR_1202.

RQ_COR_1219 Address Use

RFC 2462 *Clause:* 2 ¶25 *Type:* MUST *applies to:* Node

Context: The implementation assigns a "preferred" address with a "preferred" lifetime. The implementation also assigns a "valid" address with a "valid" lifetime.

Requirement: The valid lifetime is greater or equal to the preferred lifetime.

RFC text: valid lifetime - the length of time an address remains in the valid state (i.e., the time until invalidation). `{{The valid lifetime must be greater then or equal to the preferred lifetime}}`. When the valid lifetime expires, the address becomes invalid.

RQ_COR_1220 address: Link-local [Form]

RFC 2462 *Clause:* 2 ¶26 *Type:* MAY *applies to:* Node

Context: The implementation has an interface identifier and is on a link.

Requirement: The implementation's interface identifier is a link-dependent identifier.

RFC text: `{{interface identifier - a link-dependent identifier for an interface}}` that is (at least) unique per link [ADDR-ARCH]. `{{...The exact length of an interface identifier and the way it is created is defined in a separate link-type specific document that covers issues related to the transmission of IP over a particular link type (e.g., [IPv6-ETHER]). In many cases, the identifier will be the same as the interface's link- layer address}}`.

RQ_COR_1221 address: Link-local [Form]

RFC 2462 *Clause:* 2 *Type:* SHOULD *applies to:* Node

Context: The implementation has an interface identifier and is on a link.

Requirement: The implementation's interface identifier is (at least) unique per link.

RFC text: `{{interface identifier - a link-dependent identifier for an interface that is (at least) unique per link [ADDR-ARCH]}}`.

RQ_COR_1222 address: Link-local [Form]

RFC 2462 *Clause:* 2 *Type:* MAY *applies to:* Node

Context: The implementation has a link-layer address.

Requirement: The implementation's interface identifier is the same as the interface's link-layer address.

RFC text: `{{...In many cases, the identifier will be the same as the interface's link- layer address}}`.

RQ_COR_1223 Stateless Autoconfig

RFC 2462 *Clause:* 2 ¶26 *Type:* MUST *applies to:* Node

Context: The implementation implements Stateless address autoconfiguration.

Requirement: The implementation combines its interface identifier with a prefix to form an IPv6 address.

RFC text: `{{Stateless address autoconfiguration combines an interface identifier with a prefix to form an address}}`.

RQ_COR_1224 Stateless Autoconfig

RFC 2462 *Clause:* 3 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation is attached to a small site consisting of a set of machines attached to a single link.

Requirement: The implementation uses stateless autoconfiguration with link-local addresses.

RFC text: Small sites consisting of a set of machines attached to a single link should not require the presence of a stateful server or router as a prerequisite for communicating. `{{Plug-and-play communication is achieved through the use of link-local addresses. Link-local addresses have a well-known prefix that identifies the (single) shared link to which a set of nodes attach. A host forms a link-local address by appending its interface identifier to the link-local prefix}}`.

RQ_COR_1225 address: Link-local [Form]

RFC 2462 *Clause:* 3 ¶3,4 ¶1,5 *Type:* MUST *applies to:* Node

Context: The implementation generates its link-local address. The implementation interface identifier's length is less than 119 bits.

Requirement: The implementation forms a link-local address by appending its interface identifier to the link-local well-known prefix [FE80::0].

RFC text: Small sites consisting of a set of machines attached to a single link should not require the presence of a stateful server or router as a prerequisite for communicating. `{{Plug-and-play communication is achieved through the use of link-local addresses. Link-local addresses have a well-known prefix that identifies the (single) shared link to which a set of nodes attach. A host forms a link-local address by appending its interface identifier to the link-local prefix}}`. `{{...Nodes (both hosts and routers) begin the autoconfiguration process by generating a link-local address for the interface. A link-local address is formed by appending the interface's identifier to the well-known link-local prefix}}`. ...Autoconfiguration applies primarily to hosts, with two exceptions. Routers are expected to generate a link-local address using the procedure outlined below. In addition, routers perform Duplicate Address Detection on all addresses prior to assigning them to an interface. A link-local address is formed by prepending the well-known link- local prefix FE80::0 [ADDR-ARCH] (of appropriate length) to the interface identifier.

RQ_COR_1226 Stateless Autoconfig

RFC 2462 *Clause:* 3 ¶4 *Type:* SHOULD *applies to:* Node

Context: The implementation is attached to a large site with multiple networks and routers.

Requirement: The implementation uses Stateless Address autoconfiguration.

RFC text: `{{A large site with multiple networks and routers should not require the presence of a stateful address configuration server.}}` In order to generate site-local(deprecated) or global addresses, hosts must determine the prefixes that identify the subnets to which they attach. Routers generate periodic Router Advertisements that include options listing the set of active prefixes on a link.

RQ_COR_1228 address: Global [Assign]

RFC 2462 *Clause:* 3 ¶4 *Type:* MUST *applies to:* Host

Context: The implementation is attached to a large site with multiple networks and routers. The implementation uses Stateless Address autoconfiguration.

Requirement: The implementation generates its global address using the prefixes of the subnets to which they are attached.

RFC text: A large site with multiple networks and routers should not require the presence of a stateful address configuration server. `{{In order to generate site-local(deprecated) or global addresses, hosts must determine the prefixes that identify the subnets to which they attach. Routers generate periodic Router Advertisements that include options listing the set of active prefixes on a link.}}`.

RQ_COR_1229 Stateless Autoconfig

RFC 2462 *Clause:* 3 ¶4, 5.5.1 ¶1 *Type:* MUST *applies to:* Router

Context: The implementation is attached to large site with multiple networks and routers. Stateless autoconfiguration is used.

Requirement: The implementation generates periodic Router Advertisements, to the all-nodes multicast address, that include options listing the set of active prefixes on the link.

RFC text: A large site with multiple networks and routers should not require the presence of a stateful address configuration server. In order to generate site-local(deprecated) or global addresses, hosts must determine the prefixes that identify the subnets to which they attach. `{{Routers generate periodic Router Advertisements that include options listing the set of active prefixes on a link}}`. `{{...Router Advertisements are sent periodically to the all-nodes multicast address}}`.

RQ_COR_1230 Address Lifetime

RFC 2462 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation is assigned multiple addresses to the same interface. The lifetime periods of two addresses overlap thereby providing a transition period.

Requirement: The implementation simultaneously uses both the new address and the one being phased out during the transition period.

RFC text: Renumbering is achieved through the leasing of addresses to interfaces and the assignment of multiple addresses to the same interface. Lease lifetimes provide the mechanism through which a site phases out old prefixes. `{{The assignment of multiple addresses to an interface provides for a transition period during which both a new address and the one being phased out work simultaneously}}`.

RQ_COR_1231 Stateless Autoconfig

RFC 2462 *Clause:* 4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The implementation is on a multicast-capable link.

Requirement: The implementation starts Stateless address autoconfiguration when a multicast-capable interface is enabled.

RFC text: `{{Autoconfiguration is performed only on multicast-capable links and begins when a multicast-capable interface is enabled, e.g., during system startup}}`.

RQ_COR_1232 Stateless Autoconfig

RFC 2462 *Clause:* 4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation implements Stateless address autoconfiguration. The implementation begins the autoconfiguration process.

Requirement: The implementation generates a link-local address.

RFC text: `{{Nodes (both hosts and routers) begin the autoconfiguration process by generating a link-local address for the interface}}`.

RQ_COR_1235 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration and has a unique link-local address. The implementation receives a Neighbor Solicitation message with its link-local address in the Destination Address.

Requirement: The implementation sends a Neighbor Advertisement indicating that it already uses the link-local address received in the Neighbor Solicitation message's Destination Address.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target. `{{If another node is already using that address, it will return a Neighbor Advertisement saying so}}`. If another node is also attempting to use the same address, it will send a Neighbor Solicitation for the target as well. The exact number of times the Neighbor Solicitation is (re)transmitted and the delay time between consecutive solicitations is link-specific and may be set by system management.

RQ_COR_1237 address: DAD Timers and Counters

RFC 2462 *Clause:* 4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation implements Stateless address autoconfiguration. The implementation is generating a unique link-local address. The implementation retransmits the Neighbor Solicitation a specific number of times with a delay between consecutive transmissions.

Requirement: The number of transmissions and the delay time are link-specific.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target. If another node is already using that address, it will return a Neighbor Advertisement saying so. If another node is also attempting to use the same address, it will send a Neighbor Solicitation for the target as well. `{{The exact number of times the Neighbor Solicitation is (re)transmitted and the delay time between consecutive solicitations is link-specific}}` and may be set by system management.

RQ_COR_1238 address: DAD Timers and Counters

RFC 2462 *Clause:* 4 ¶2 *Type:* MAY *applies to:* Node

Context: The implementation implements Stateless address autoconfiguration. The implementation is generating a unique link-local address. The implementation retransmits the Neighbor Solicitation a specific number of times with a delay between consecutive transmissions. The number of transmissions and the delay time are link-specific.

Requirement: System management sets the number of transmissions and the delay time.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target. If another node is already using that address, it will return a Neighbor Advertisement saying so. If another node is also attempting to use the same address, it will send a Neighbor Solicitation for the target as well. `{{The exact number of times the Neighbor Solicitation is (re)transmitted and the delay time between consecutive solicitations is link-specific}}` and `{{may be set by system management}}`.

RQ_COR_1239 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 4 ¶2-3 *Type:* MUST *applies to:* Node

Context: The implementation implements Stateless address autoconfiguration. The implementation is generating a unique link-local address. The implementation sends a Neighbor Solicitation message containing its tentative unique link-local address in the Target field. The Destination Address of the solicitation is set to the solicited-node multicast address. The implementation then determines that its tentative link-local address is not unique.

Requirement: The implementation stops the autoconfiguration.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target. `{{If another node is already using that address, it will return a Neighbor Advertisement saying so}}`. `{{If a node determines that its tentative link-local address is not unique, autoconfiguration stops}}` and manual configuration of the interface is required.

RQ_COR_1240 Stateless Autoconfig

RFC 2462 *Clause:* 4 ¶2-3 *Type:* MAY *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The implementation is generating a unique link-local address. The implementation sends a Neighbor Solicitation message containing its tentative unique link-local address as the Destination Address. The implementation determines that its tentative link-local address is not unique. The implementation has an alternate interface identifier.

Requirement: The implementation sends a Neighbor Solicitation message containing a tentative unique link-local address using the alternate interface identifier as part of the Destination Address.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target. `{{If another node is already using that address, it will return a Neighbor Advertisement saying so}}`. If a node determines that its tentative link-local address is not unique, autoconfiguration stops and manual configuration of the interface is required. `{{To simplify recovery in this case, it should be possible for an administrator to supply an alternate interface identifier that overrides the default identifier in such a way that the autoconfiguration mechanism can then be applied using the new (presumably unique) interface identifier}}`. Alternatively, link-local and other addresses will need to be configured manually.

RQ_COR_1243 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 4 ¶2, 4 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The implementation is generating a unique link-local address. The implementation attempt to verify that this "tentative" address is not already in use by another node on the link. The implementation sends [one or more] Neighbor Solicitation message containing the tentative address as the target. The implementation does not receive Neighbor Advertisements or Neighbor Solicitations saying the tentative address is already used.

Requirement: The node ascertains that the tentative link-local address is unique.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. `{{Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target. If another node is already using that address, it will return a Neighbor Advertisement saying so. If another node is also attempting to use the same address, it will send a Neighbor Solicitation for the target as well. The exact number of times the Neighbor Solicitation is (re)transmitted and the delay time between consecutive solicitations is link-specific and may be set by system management.}}`. `{{Once a node ascertains that its tentative link-local address is unique, it assigns it to the interface}}`. At this point, the node has IP-level connectivity with neighboring nodes.

RQ_COR_1244 Stateless Autoconfig

RFC 2462 *Clause:* 4 ¶2, 4 *Type:* MUST *applies to:* Node

Context: The implementation implements Stateless address autoconfiguration. The implementation is generating a unique link-local address. The node ascertains using Duplicate Address Detection that the tentative link-local address is unique.

Requirement: The implementation assigns the tentative link-local address to the interface and uses the interface for IP-level connectivity.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. `{{Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target}}`. `{{Once a node ascertains that its tentative link-local address is unique, it assigns it to the interface}}`. At this point, the node has IP-level connectivity with neighboring nodes.

RQ_COR_1245 address: Global [Assign]

RFC 2462 *Clause:* 4 ¶4, 5 *Type:* MUST *applies to:* Host

Context: The implementation ascertains that its tentative link-local address for an interface is unique. The implementation assigns the link-local address to the interface.

Requirement: The implementation waits for a Router Advertisement.

RFC text: `{{The next phase of autoconfiguration involves obtaining a Router Advertisement or determining that no routers are present}}`.

RQ_COR_1246 address: [Configure]

RFC 2462 *Clause:* 4 ¶5 *Type:* MUST *applies to:* Router

Context: The implementation is configured for autoconfiguration.

Requirement: The implementation transmits Router Advertisements that specify the sort(s) autoconfiguration for a host on the link.

RFC text: `{{If routers are present, they will send Router Advertisements that specify what sort of autoconfiguration a host should do}}`.

RQ_COR_1248 address: Global [Assign]

RFC 2462 *Clause:* 4 ¶6 *Type:* MUST *applies to:* Host

Context: The implementation assigns the link-local address to the interface. A router is present in the same network link as the implementation. The delay between successive Router Advertisements is longer than the implementation wants to wait.

Requirement: The implementation sends one or more Router Solicitations to the all-routers multicast group.

RFC text: `{{Routers send Router Advertisements periodically, but the delay between successive advertisements will generally be longer than a host performing autoconfiguration will want to wait [DISCOVERY]. To obtain an advertisement quickly, a host sends one or more Router Solicitations to the all-routers multicast group}}`.

RQ_COR_1249 address: [Configure]

RFC 2462 *Clause:* 4 ¶8 *Type:* MUST *applies to:* Host

Context: The implementation assigns the link-local address to the interface. The implementation periodically receives Router Advertisements that specify what sort of autoconfiguration the implementation should do.

Requirement: The implementation adds to and refreshes the information received in the previous Router Advertisements.

RFC text: `{{Because routers generate Router Advertisements periodically, hosts will continually receive new advertisements. Hosts process the information contained in each advertisement as described above, adding to and refreshing information received in previous advertisements}}`.

RQ_COR_1250 address: Global [Assign]

RFC 2462 *Clause:* 4 ¶9, 5.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is assigning unicast addresses to an interface.

Requirement: The implementation tests all the unicast addresses for uniqueness using Duplicate Address Detection prior to their assignment to an interface.

RFC text: `{{For safety, all addresses must be tested for uniqueness prior to their assignment to an interface}}`. `{{Duplicate Address Detection MUST take place on all unicast addresses, regardless of whether they are obtained through stateful, stateless or manual configuration}}`

RQ_COR_1251 address: Global [Assign]

RFC 2462 *Clause:* 4 ¶9, 5.4 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation is assigning address to its interface and uses Stateless Autoconfiguration. The implementation has already verified the uniqueness of a link-local address.

Requirement: The implementation does not individually test additional addresses created from the same interface identifier used in the configuration of the unique link-local address.

RFC text: For safety, all addresses must be tested for uniqueness prior to their assignment to an interface. `{{In the case of addresses created through stateless autoconfig, however, the uniqueness of an address is determined primarily by the portion of the address formed from an interface identifier. Thus, if a node has already verified the uniqueness of a link-local address, additional addresses created from the same interface identifier need not be tested individually}}`.

RQ_COR_1252 Stateful Autoconfig

RFC 2462 *Clause:* 4 ¶9 *Type:* SHOULD *applies to:* Node

Context: The implementation is assigning addresses to an interface using Stateful Address Autoconfiguration.

Requirement: The implementation individually tests for uniqueness using Duplicate Address Detection all addresses obtained via Stateful address autoconfiguration.

RFC text: For safety, all addresses must be tested for uniqueness prior to their assignment to an interface. `{{In contrast, all addresses obtained manually or via stateful address autoconfiguration should be tested for uniqueness individually}}`.

RQ_COR_1253 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 4 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation is assigning address to its interface. The implementation can implement Duplicate Address Detection. The administrator disables use of Duplicate Address Detection on a given interface.

Requirement: The implementation does not implement Duplicate Address Detection on the given interface.

RFC text: `{{To accommodate sites that believe the overhead of performing Duplicate Address Detection outweighs its benefits, the use of Duplicate Address Detection can be disabled through the administrative setting of a per-interface configuration flag.}}`.

RQ_COR_1254 address: [Configure]

RFC 2462 *Clause:* 4 ¶10 *Type:* MAY *applies to:* Node

Context: The implementation is assigning an address to its interface.

Requirement: The implementation generates its link-local address and verifies its uniqueness in parallel with waiting for a Router Advertisement.

RFC text: `{{To speed the autoconfiguration process, a host may generate its link-local address (and verify its uniqueness) in parallel with waiting for a Router Advertisement. Because a router may delay responding to a Router Solicitation for a few seconds, the total time needed to complete autoconfiguration can be significantly longer if the two steps are done serially}}`.

RQ_COR_1255 address: [Configure]

RFC 2462 *Clause:* 5 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation is multihomed and it is assigning addresses to its interfaces.

Requirement: The implementation generates the autoconfigurated addresses independently on each interface.

RFC text: `{{For multihomed hosts, autoconfiguration is performed independently on each interface}}`.

RQ_COR_1256 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.1 ¶1-4 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration.

Requirement: For each multicast interface, the implementation uses the system management configured variable DupAddrDetectTransmits. This variable indicates the number of consecutive Neighbor Solicitation messages sent while performing Duplicate Address Detection on a tentative address.

RFC text: `{{A node MUST allow the following autoconfiguration-related variable to be configured by system management for each multicast interface: DupAddrDetectTransmits The number of consecutive Neighbor Solicitation messages sent while performing Duplicate Address Detection on a tentative address}}`. A value of zero indicates that Duplicate Address Detection is not performed on tentative addresses. A value of one indicates a single transmission with no follow up retransmissions. Default: 1, but may be overridden by a link-type specific value in the document that covers issues related to the transmission of IP over a particular link type (e.g., [IPv6-ETHER]).

RQ_COR_1257 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.1 ¶1-4 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. For a multicast interface, DupAddrDetectTransmits is not set by the system operator.

Requirement: The implementation uses DupAddrDetectTransmits set to the default value of 1 meaning a single transmission of a Neighbor Solicitation message for the tentative address with no follow up retransmissions.

RFC text: `{{A node MUST allow the following autoconfiguration-related variable to be configured by system management for each multicast interface: DupAddrDetectTransmits The number of consecutive Neighbor Solicitation messages sent while performing Duplicate Address Detection on a tentative address. A value of zero indicates that Duplicate Address Detection is not performed on tentative addresses. A value of one indicates a single transmission with no follow up retransmissions. Default: 1, but may be overridden by a link-type specific value in the document that covers issues related to the transmission of IP over a particular link type (e.g., [IPv6-ETHER])}}`.

RQ_COR_1258 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.1 ¶1-4 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. For a multicast interface, the system operator overrides the default value of DupAddrDetectTransmits with a link-type specific value.

Requirement: The implementation uses the link-type specific value of DupAddrDetectTransmits in Duplicate Address Detection.

RFC text: `{{A node MUST allow the following autoconfiguration-related variable to be configured by system management for each multicast interface: DupAddrDetectTransmits The number of consecutive Neighbor Solicitation messages sent while performing Duplicate Address Detection on a tentative address. A value of zero indicates that Duplicate Address Detection is not performed on tentative addresses. A value of one indicates a single transmission with no follow up retransmissions. Default: 1, but may be overridden by a link-type specific value in the document that covers issues related to the transmission of IP over a particular link type (e.g., [IPv6-ETHER])}}`.

RQ_COR_1259 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.1 ¶1-4 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. For a multicast interface, the system operator overrides the default value of DupAddrDetectTransmits variable setting it to zero.

Requirement: The implementation does not perform Duplicate Address Detection on the tentative address.

RFC text: A node MUST allow the following autoconfiguration-related variable to be configured by system management for each multicast interface: DupAddrDetectTransmits The number of consecutive Neighbor Solicitation messages sent while performing Duplicate Address Detection on a tentative address. `{{A value of zero indicates that Duplicate Address Detection is not performed on tentative addresses}}`. A value of one indicates a single transmission with no follow up retransmissions. Default: 1, but may be overridden by a link-type specific value in the document that covers issues related to the transmission of IP over a particular link type (e.g., [IPv6-ETHER]).

RQ_COR_1262 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.1 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The system operator has configured the DupAddrDetectTransmits value greater than 1.

Requirement: The implementation uses RetransTimer as the delay between consecutive Neighbor Solicitation transmissions performed during Duplicate Address Detection.

RFC text:
```
{{Autoconfiguration also assumes the presence of the variable
RetransTimer as defined in [DISCOVERY]. For autoconfiguration
purposes, RetransTimer specifies the delay between consecutive
Neighbor Solicitation transmissions performed during Duplicate
Address Detection (if DupAddrDetectTransmits is greater than 1), as
well as the time a node waits after sending the last Neighbor
Solicitation before ending the Duplicate Address Detection process.}}.
```

RQ_COR_1263 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.1 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation uses the variable named RetransTimer.

Requirement: The implementation uses RetransTimer as the waiting time after sending the last Neighbor Solicitation before ending the Duplicate Address Detection process.

RFC text:
```
{{Autoconfiguration also assumes the presence of the variable
RetransTimer as defined in [DISCOVERY]. For autoconfiguration
purposes, RetransTimer specifies the delay between consecutive
Neighbor Solicitation transmissions performed during Duplicate
Address Detection (if DupAddrDetectTransmits is greater than 1), as
well as the time a node waits after sending the last Neighbor
Solicitation before ending the Duplicate Address Detection process.}}.
```

RQ_COR_1271 address: [Configure]

RFC 2462 *Clause:* 5.2 ¶7 *Type:* MUST *applies to:* Host

Context: The implementation uses both autoconfigured and manually configured addresses.

Requirement: The implementation maintains a list of both autoconfigured addresses and manually configured addresses together with their corresponding lifetimes.

RFC text:
```
{{A host also maintains a list of addresses together with their
corresponding lifetimes. The address list contains both
autoconfigured addresses and those configured manually}}.
```

RQ_COR_1272 Stateless Autoconfig

RFC 2462 *Clause:* 5.3 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The IPv6 system initializes.

Requirement: The implementation enables the interface and forms a link-local address.

RFC text: `{{A node forms a link-local address whenever an interface becomes enabled. An interface may become enabled after any of the following events: - The interface is initialized at system startup time. - The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management. - The interface attaches to a link for the first time. - The interface becomes enabled by system management after having been administratively disabled}}`. See RQ_COR_1231.

RQ_COR_1274 Invalid Stateless Address Autoconfig Syntax

RFC 2462 *Clause:* 5.3 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The implementation interface identifier's length is more than 118 bits.

Requirement: The implementation stops Stateless Address Autoconfiguration. Manual configuration is required.

RFC text: `{{A link-local address is formed by prepending the well-known link-local prefix FE80::0 [ADDR-ARCH] (of appropriate length) to the interface identifier. If the interface identifier has a length of N bits, the interface identifier replaces the right-most N zero bits of the link-local prefix. If the interface identifier is more than 118 bits in length, autoconfiguration fails and manual configuration is required. Note that interface identifiers will typically be 64-bits long and based on EUI-64 identifiers as described in [ADDR-ARCH]}}`.

RQ_COR_1275 address: Link-local [Form]

RFC 2462 *Clause:* 2 ¶26,5.3 ¶6 *Type:* MAY *applies to:* Node

Context: The implementation forms a link-layer address.

Requirement: The implementation's interface identifier is 64-bits long and based on EUI-64 identifiers.

RFC text: `{{In many cases, the identifier will be the same as the interface's link- layer address}}`. `{{...Note that interface identifiers will typically be 64-bits long and based on EUI-64 identifiers as described in [ADDR-ARCH]}}`. See RQ_COR_1222.

RQ_COR_1276 address: Link-local [Form]

RFC 2462 *Clause:* 5.3 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation forms a link-layer address.

Requirement: The implementation's link-local address has both an infinite preferred and a valid lifetime.

RFC text: `{{A link-local address has an infinite preferred and an infinite valid lifetime; it is never timed out}}`.

RQ_COR_1277 Stateless Autoconfig

RFC 2462 *Clause:* 4 ¶9, 5.4 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation is assigning an anycast address to an interface and uses Stateless Autoconfiguration.

Requirement: The implementation does not perform Duplicate Address Detection on the anycast address.

RFC text: {{For safety, all addresses must be tested for uniqueness prior to their assignment to an interface}}. {{Duplicate Address Detection MUST take place on all unicast addresses, regardless of whether they are obtained through stateful, stateless or manual configuration, with the exception of the following cases: Duplicate Address Detection MUST NOT be performed on anycast addresses}}.

RQ_COR_1278 Neighbor Discovery - Invalid ND Message

RFC 2462 *Clause:* 5.4.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration and receives Neighbor Solicitation messages that do not pass the validity checks specified in [DISCOVERY].

Requirement: The implementation silently discards the messages.

RFC text: {{A node MUST silently discard any Neighbor Solicitation or Advertisement message that does not pass the validity checks specified in [DISCOVERY]}}. A solicitation that passes these validity checks is called a valid solicitation or valid advertisement.

RQ_COR_1279 Stateless Autoconfig

RFC 2462 *Clause:* 5.4.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration.

Requirement: The implementation joins the all-nodes multicast address before sending a Neighbor Solicitation.

RFC text: {{Before sending a Neighbor Solicitation, an interface MUST join the all-nodes multicast address and the solicited-node multicast address of the tentative address}}. The former insures that the node receives Neighbor Advertisements from other nodes already using the address; the latter insures that two nodes attempting to use the same address simultaneously detect each other's presence.

RQ_COR_1280 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 4 ¶2, 5.4.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration to verify that its tentative address is not already in use by another implementation.

Requirement: The implementation sends Neighbor Solicitations for DupAddrDetectTransmits times, each separated by RetransTimer milliseconds. The solicitation's Target Address is set to the address being checked, the IP source is set to the Unspecified Address (0::0) and the IP destination is set to the solicited-node multicast address of the target.

RFC text: Before the link-local address can be assigned to an interface and used, however, a node must attempt to verify that this "tentative" address is not already in use by another node on the link. `{{Specifically, it sends a Neighbor Solicitation message containing the tentative address as the target}}`. If another node is already using that address, it will return a Neighbor Advertisement saying so. If another node is also attempting to use the same address, it will send a Neighbor Solicitation for the target as well. The exact number of times the Neighbor Solicitation is (re)transmitted and the delay time between consecutive solicitations is link-specific and may be set by system management. `{{...To check an address, a node sends DupAddrDetectTransmits Neighbor Solicitations, each separated by RetransTimer milliseconds. The solicitation's Target Address is set to the address being checked, the IP source is set to the unspecified address and the IP destination is set to the solicited-node multicast address of the target address}}`.

RQ_COR_1281 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.4.2 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. A Neighbor Solicitation message is the first message to be sent from the implementation's interface after interface (re)initialization.

Requirement: The implementation delays sending the Neighbor Solicitation message by a random delay between 0 and MAX_RTR_SOLICITATION_DELAY as specified in [DISCOVERY].

RFC text: `{{If the Neighbor Solicitation is the first message to be sent from an interface after interface (re)initialization, the node should delay sending the message by a random delay between 0 and MAX_RTR_SOLICITATION_DELAY as specified in [DISCOVERY]}}`. This serves to alleviate congestion when many nodes start up on the link at the same time, such as after a power failure, and may help to avoid race conditions when more than one node is trying to solicit for the same address at the same time.

RQ_COR_1282 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.2 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. A Neighbor Solicitation message is the first message to be sent from implementation's interface after interface (re)initialization. The implementation is waiting to send the Neighbor Solicitation during the random delay period between 0 and MAX_RTR_SOLICITATION_DELAY.

Requirement: The implementation receives and process datagrams sent to the all-nodes multicast address or solicited-node multicast address of the tentative during the random delay period.

RFC text: `{{If the Neighbor Solicitation is the first message to be sent from an interface after interface (re)initialization, the node should delay sending the message by a random delay between 0 and MAX_RTR_SOLICITATION_DELAY as specified in [DISCOVERY]}}`. `{{In order to improve the robustness of the Duplicate Address Detection algorithm, an interface MUST receive and process datagrams sent to the all-nodes multicast address or solicited-node multicast address of the tentative address while delaying transmission of the initial Neighbor Solicitation}}`.

RQ_COR_1283 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.4.3 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation receives a valid Neighbor Solicitation message. The Target field of the solicitation is set to the implementation's tentative address. The Source Address of the solicitation is a unicast address indicating that the solicitation's sender is performing address resolution for the address in the Target field.

Requirement: The implementation ignores the solicitation.

RFC text: On receipt of a valid Neighbor Solicitation message on an interface, node behavior depends on whether the target address is tentative or not. If the target address is not tentative (i.e., it is assigned to the receiving interface), the solicitation is processed as described in [DISCOVERY]. `{{If the target address is tentative, and the source address is a unicast address, the solicitation's sender is performing address resolution on the target; the solicitation should be silently ignored}}`. Otherwise, processing takes place as described below. In all cases, a node MUST NOT respond to a Neighbor Solicitation for a tentative address.

RQ_COR_1284 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.3 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation receives a Neighbor Solicitation message with a tentative target address. The source address of the message is a unicast address.

Requirement: The implementation ignores the solicitation.

RFC text: On receipt of a valid Neighbor Solicitation message on an interface, node behavior depends on whether the target address is tentative or not. `{{If the target address is tentative, and the source address is a unicast address, the solicitation's sender is performing address resolution on the target; the solicitation should be silently ignored}}`.

RQ_COR_1285 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.3 ¶1-2 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation receives a valid Neighbor Solicitation message from a node other than the implementation; i.e. this is not a looped back solicitation. The Target field of the solicitation is set to the implementation's tentative address. The source address of the Neighbor Solicitation is the Unspecified Address (0::0) indicating that the solicitation is from a node performing Duplicate Address Detection.

Requirement: The implementation's tentative address is not used (because it is duplicate).

RFC text: On receipt of a valid Neighbor Solicitation message on an interface, node behavior depends on whether the target address is tentative or not. If the target address is not tentative (i.e., it is assigned to the receiving interface), the solicitation is processed as described in [DISCOVERY]. `{{If the target address is tentative}}`, and the source address is a unicast address, the solicitation's sender is performing address resolution on the target; the solicitation should be silently ignored. Otherwise, processing takes place as described below. In all cases, a node MUST NOT respond to a Neighbor Solicitation for a tentative address. `{{If the source address of the Neighbor Solicitation is the unspecified address, the solicitation is from a node performing Duplicate Address Detection. If the solicitation is from another node, the tentative address is a duplicate and should not be used (by either node)}}`. If the solicitation is from the node itself (because the node loops back multicast packets), the solicitation does not indicate the presence of a duplicate address.

RQ_COR_1286 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.3 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation receives a valid Neighbor Solicitation message from itself; i.e. this is a looped back solicitation. The Target field of the solicitation is set to the implementation's tentative address. The source address of the Neighbor Solicitation is the Unspecified Address (0::0) indicating that the solicitation is from a node performing Duplicate Address Detection.

Requirement: The implementation does not consider that the received loop-back Neighbor Solicitation indicates a duplicate address.

RFC text: On receipt of a valid Neighbor Solicitation message on an interface, node behavior depends on whether the target address is tentative or not. If the target address is not tentative (i.e., it is assigned to the receiving interface), the solicitation is processed as described in [DISCOVERY]. `{{If the target address is tentative}}`, and the source address is a unicast address, the solicitation's sender is performing address resolution on the target; the solicitation should be silently ignored. Otherwise, processing takes place as described below. In all cases, a node MUST NOT respond to a Neighbor Solicitation for a tentative address. `{{If the source address of the Neighbor Solicitation is the unspecified address, the solicitation is from a node performing Duplicate Address Detection}}`. If the solicitation is from another node, the tentative address is a duplicate and should not be used (by either node). `{{If the solicitation is from the node itself (because the node loops back multicast packets), the solicitation does not indicate the presence of a duplicate address}}`.

RQ_COR_1287 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.3 ¶4-6 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation receives a valid Neighbor Solicitation with the solicitation's Target field set to the implementation's tentative address prior to the implementation sending the Neighbor Solicitation for this same tentative address.

Requirement: The implementation's tentative address is discarded and autoconfiguration stops.

RFC text: `{{The following tests identify conditions under which a tentative address is not unique: If a Neighbor Solicitation for a tentative address is received prior to having sent one, the tentative address is a duplicate. This condition occurs when two nodes run Duplicate Address Detection simultaneously, but transmit initial solicitations at different times (e.g., by selecting different random delay values before transmitting an initial solicitation)}}`.

RQ_COR_1288 address: DAD Timers and Counters

RFC 2462 *Clause:* 5.4.3 ¶4-6 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation receives more Neighbor Solicitations whose Target Address field contains the implementation's tentative address than the number expected based on the loopback semantics.

Requirement: The implementation's tentative address is discarded.

RFC text: `{{The following tests identify conditions under which a tentative address is not unique: If the actual number of Neighbor Solicitations received exceeds the number expected based on the loopback semantics (e.g., the interface does not loopback packet, yet one or more solicitations was received), the tentative address is a duplicate. This condition occurs when two nodes run Duplicate Address Detection simultaneously and transmit solicitations at roughly the same time.}}`.

RQ_COR_1290 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation receives a valid Neighbor Advertisement message with the Target field set to the implementation's tentative address.

Requirement: The implementation's tentative address is not unique and discarded.

RFC text: `{{On receipt of a valid Neighbor Advertisement message on an interface, node behavior depends on whether the target address is tentative or matches a unicast or anycast address assigned to the interface. If the target address is assigned to the receiving interface, the solicitation is processed as described in [DISCOVERY]. If the target address is tentative, the tentative address is not unique}}`.

RQ_COR_1291 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.5 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation discovers that a tentative address is duplicated.

Requirement: The implementation logs a system management error.

RFC text: `{{A tentative address that is determined to be a duplicate as described above, MUST NOT be assigned to an interface and the node SHOULD log a system management error. If the address is a link-local address formed from an interface identifier, the interface SHOULD be disabled}}`.

RQ_COR_1292 address: Global [Assign]

RFC 2462 *Clause:* 5.5 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is creating a global address.

Requirement: The implementation uses by default the global address Stateless autoconfiguration mechanism.

RFC text: `{{Creation of global and site-local (deprecated) addresses and configuration of other parameters as described in this section SHOULD be locally configurable. However, the processing described below MUST be enabled by default.}}`.

RQ_COR_1293 address: Global [Assign]

RFC 2462 *Clause:* 5.5 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation is creating a global address.

Requirement: The implementation allows local configuration of global addresses.

RFC text: `{{Creation of global and site-local (deprecated) addresses and configuration of other parameters as described in this section SHOULD be locally configurable. However, the processing described below MUST be enabled by default.}}`.

RQ_COR_1294 Stateful Autoconfig

RFC 2462 *Clause:* 5.5.2 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is creating a global address and has determined that there are no routers on the link. There is no option for disabling stateful autoconfiguration for this situation.

Requirement: The implementation attempts to use stateful autoconfiguration to obtain addresses and other configuration information.

RFC text: `{{If a link has no routers, a host MUST attempt to use stateful autoconfiguration to obtain addresses and other configuration information. An implementation MAY provide a way to disable the invocation of stateful autoconfiguration in this case, but the default SHOULD be enabled}}`.

RQ_COR_1296 M-bit [Use of]

RFC 2462 *Clause:* 5.5.3 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation then receives a valid Router Advertisement with the M-bit set to TRUE.

Requirement: The implementation invokes the stateful address autoconfiguration protocol requesting address and other information.

RFC text: `{{On receipt of a valid Router Advertisement (as defined in [DISCOVERY]), a host copies the value of the advertisement's M bit into ManagedFlag. If the value of ManagedFlag changes from FALSE to TRUE, and the host is not already running the stateful address autoconfiguration protocol, the host should invoke the stateful address autoconfiguration protocol, requesting both address information and other information}}`. If the value of the ManagedFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration, i.e., the change in the value of the ManagedFlag has no effect. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful address configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

RQ_COR_1297 M-bit [Use of]

RFC 2462 *Clause:* 5.5.3 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateful Autoconfiguration. The implementation receives a valid Router Advertisement with the M-bit set to FALSE.

Requirement: The implementation continues running the Stateful autoconfiguration.

RFC text: `{{On receipt of a valid Router Advertisement (as defined in [DISCOVERY]), a host copies the value of the advertisement's M bit into ManagedFlag}}`. If the value of ManagedFlag changes from FALSE to TRUE, and the host is not already running the stateful address autoconfiguration protocol, the host should invoke the stateful address autoconfiguration protocol, requesting both address information and other information. `{{If the value of the ManagedFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration, i.e., the change in the value of the ManagedFlag has no effect}}`. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful address configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

RQ_COR_1298 M-bit [Use of]

RFC 2462 *Clause:* 5.5.3 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement with the M-bit set to the same value as the previous Router Advertisements.

Requirement: The implementation does not take a special action; i.e. the implementation continues Stateless Autoconfiguration.

RFC text: `{{On receipt of a valid Router Advertisement (as defined in [DISCOVERY]), a host copies the value of the advertisement's M bit into ManagedFlag}}`. If the value of ManagedFlag changes from FALSE to TRUE, and the host is not already running the stateful address autoconfiguration protocol, the host should invoke the stateful address autoconfiguration protocol, requesting both address information and other information. If the value of the ManagedFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration, i.e., the change in the value of the ManagedFlag has no effect. `{{If the value of the flag stays unchanged, no special action takes place}}`. In particular, a host MUST NOT reinvoke stateful address configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

RQ_COR_1299 M-bit [Use of]

RFC 2462 *Clause:* 5.5.3 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using using Stateful Autoconfiguration as a result of an earlier advertisement. The implementation receives a valid Router Advertisement with its M-bit set to TRUE.

Requirement: The implementation does not reinvoke stateful address configuration.

RFC text: On receipt of a valid Router Advertisement (as defined in [DISCOVERY]), a host copies the value of the advertisement's M bit into ManagedFlag. If the value of ManagedFlag changes from FALSE to TRUE, and the host is not already running the stateful address autoconfiguration protocol, the host should invoke the stateful address autoconfiguration protocol, requesting both address information and other information. If the value of the ManagedFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration, i.e., the change in the value of the ManagedFlag has no effect. If the value of the flag stays unchanged, no special action takes place. `{{In particular, a host MUST NOT reinvoke stateful address configuration if it is already participating in the stateful protocol as a result of an earlier advertisement}}`.

RQ_COR_1300 O-Flag [Use of]

RFC 2462 *Clause:* 5.5.3 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement with the O-flag set to TRUE. Previous O-flags were set to FALSE. The M-flag is set to FALSE.

Requirement: The implementation invokes the stateful autoconfiguration protocol requesting information without addresses.

RFC text: `{{An advertisement's O flag field is processed in an analogous manner. A host copies the value of the O flag into OtherConfigFlag. If the value of OtherConfigFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol, requesting information (excluding addresses if ManagedFlag is set to FALSE)}}`. If the value of the OtherConfigFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration protocol, i.e., the change in the value of OtherConfigFlag has no effect. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

RQ_COR_1301 O-Flag [Use of]

RFC 2462 *Clause:* 5.5.3 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement with the O-flag set to TRUE. Previous O-flags were set to FALSE. The M-flag is set to TRUE.

Requirement: The implementation invokes the stateful autoconfiguration protocol requesting address and other information.

RFC text: `{{An advertisement's O flag field is processed in an analogous manner. A host copies the value of the O flag into OtherConfigFlag. If the value of OtherConfigFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol, requesting information (excluding addresses if ManagedFlag is set to FALSE)}}`. If the value of the OtherConfigFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration protocol, i.e., the change in the value of OtherConfigFlag has no effect. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

RQ_COR_1302 O-Flag [Use of]

RFC 2462 *Clause:* 5.5.3 ¶2 *Type:* SHOULD *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateful autoconfiguration. The implementation receives a valid Router Advertisement with the O-flag set to FALSE. Previous O-flags were set to TRUE.

Requirement: The implementation continues running the stateful address autoconfiguration.

RFC text: An advertisement's O flag field is processed in an analogous manner. A host copies the value of the O flag into OtherConfigFlag. If the value of OtherConfigFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol, requesting information (excluding addresses if ManagedFlag is set to FALSE). `{{If the value of the OtherConfigFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration protocol, i.e., the change in the value of OtherConfigFlag has no effect}}`. If the value of the flag stays unchanged, no special action takes place. In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

RQ_COR_1303 O-Flag [Use of]

RFC 2462 *Clause:* 5.5.3 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid Router Advertisement with the O-flag remaining the same as in previous advertisements.

Requirement: The implementation continues using Stateless Autoconfiguration.

RFC text: An advertisement's O flag field is processed in an analogous manner. A host copies the value of the O flag into OtherConfigFlag. If the value of OtherConfigFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol, requesting information (excluding addresses if ManagedFlag is set to FALSE). If the value of the OtherConfigFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration protocol, i.e., the change in the value of OtherConfigFlag has no effect. `{{If the value of the flag stays unchanged, no special action takes place}}`. In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

RQ_COR_1304 O-Flag [Use of]

RFC 2462 *Clause:* 5.5.3 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateful Autoconfiguration. The implementation receives a valid Router Advertisement with the O-flag set to FALSE. Previous O-flags were set to TRUE.

Requirement: The implementation does not reinvoke stateful address configuration.

RFC text: An advertisement's O flag field is processed in an analogous manner. A host copies the value of the O flag into OtherConfigFlag. If the value of OtherConfigFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol, requesting information (excluding addresses if ManagedFlag is set to FALSE). If the value of the OtherConfigFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration protocol, i.e., the change in the value of OtherConfigFlag has no effect. If the value of the flag stays unchanged, no special action takes place. `{{In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement}}`.

RQ_COR_1305 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶3-4 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the Autonomous flag in the Prefix-Information Option is not set.

Requirement: The implementation silently ignores the Prefix-Information Option.

RFC text: `{{For each Prefix-Information option in the Router Advertisement: ...a) If the Autonomous flag is not set, silently ignore the Prefix Information option}}`.

RQ_COR_1306 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶3, 5 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the prefix in the Prefix-Information Option is the link-local prefix.

Requirement: The implementation silently ignores the Prefix-Information Option.

RFC text: `{{For each Prefix-Information option in the Router Advertisement: ...b) If the prefix is the link-local prefix, silently ignore the Prefix Information option}}`.

RQ_COR_1307 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶3, 6 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the preferred lifetime in the Prefix-Information Option is greater than the valid lifetime.

Requirement: The implementation silently ignores the Prefix Information Option.

RFC text: `{{For each Prefix-Information option in the Router Advertisement: ...c) If the preferred lifetime is greater than the valid lifetime, silently ignore the Prefix Information option. A node MAY wish to log a system management error in this case}}`.

RQ_COR_1308 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶3, 6 *Type:* MAY *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the preferred lifetime in the Prefix-Information Option is greater than the valid lifetime.

Requirement: The implementation logs a system management error.

RFC text: `{{For each Prefix-Information option in the Router Advertisement: ...c) If the preferred lifetime is greater than the valid lifetime, silently ignore the Prefix Information option. A node MAY wish to log a system management error in this case}}`.

RQ_COR_1309 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶3, ¶7, ¶9 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using only Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the prefix advertised in the Prefix-Information Option does not match the prefix of an address already in the list and the Valid Lifetime is not 0.

Requirement: The implementation forms an address by combining the advertised prefix in the Prefix-Information Option with the link's interface identifier. It adds the address to the list of addresses assigned to the interface, initializing its preferred and valid lifetime values from the Prefix Information Option.

RFC text: `{{For each Prefix-Information option in the Router Advertisement: ...d) If the prefix advertised does not match the prefix of an address already in the list, and the Valid Lifetime is not 0, form an address (and add it to the list) by combining the advertised prefix with the link's interface identifier}}`.

RQ_COR_1310 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶8 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid Router Advertisement. The sum of the length of the prefix advertised in the Prefix-Information Option and the length of the link's interface identifier does not equal 128 bits.

Requirement: The implementation ignores the Prefix-Information Option.

RFC text: `{{If the sum of the prefix length and interface identifier length does not equal 128 bits, the Prefix Information option MUST be ignored. An implementation MAY wish to log a system management error in this case}}`.

RQ_COR_1311 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶8 *Type:* MAY *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid Router Advertisement. The sum of the length of the prefix advertised in the Prefix-Information Option and the length of the link's interface identifier does not equal 128 bits.

Requirement: The implementation logs a system management error.

RFC text: `{{If the sum of the prefix length and interface identifier length does not equal 128 bits, the Prefix Information option MUST be ignored. An implementation MAY wish to log a system management error in this case}}`.

RQ_COR_1313 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶10-11 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the prefix advertised in the Prefix-Information Option matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface. The received Valid Lifetime in the received advertisement is greater than 2 hours.

Requirement: The implementation updates the stored Lifetime [StoredLifetime] of the corresponding address.

RFC text: `{{...e) If the advertised prefix matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface, the specific action to perform depends on the Valid Lifetime in the received advertisement and the Lifetime associated with the previously autoconfigured address (which we call StoredLifetime in the discussion that follows): ...1) If the received Lifetime is greater than 2 hours or greater than StoredLifetime, update the stored Lifetime of the corresponding address}}`.

RQ_COR_1314 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶10-11 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the prefix advertised in the Prefix-Information Option matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface. The received Valid Lifetime in the received advertisement is greater than StoredLifetime in the implementation.

Requirement: The implementation updates the stored Lifetime [StoredLifetime] of the corresponding address.

RFC text: `{{...e) If the advertised prefix matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface, the specific action to perform depends on the Valid Lifetime in the received advertisement and the Lifetime associated with the previously autoconfigured address (which we call StoredLifetime in the discussion that follows): ...1) If the received Lifetime is greater than 2 hours or greater than StoredLifetime, update the stored Lifetime of the corresponding address}}`.

RQ_COR_1315 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶10, 12 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid and NOT authenticated (e.g., via IPSec [RFC 2402]) Router Advertisement where the prefix advertised in the Prefix-Information Option matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface. The StoredLifetime in the implementation is less than or equal to 2 hours and the received Lifetime in the Prefix Information Option is less than or equal to StoredLifetime.

Requirement: The implementation ignores the Prefix-Information Option.

RFC text: {{...e) If the advertised prefix matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface, the specific action to perform depends on the Valid Lifetime in the received advertisement and the Lifetime associated with the previously autoconfigured address (which we call StoredLifetime in the discussion that follows): ...2) If the StoredLifetime is less than or equal to 2 hours and the received Lifetime is less than or equal to StoredLifetime, ignore the prefix, unless the Router Advertisement from which his Prefix Information option was obtained has been authenticated (e.g., via IPSec [RFC 2402]). If the Router Advertisment was authenticated, the StoredLifetime should be set to the Lifetime in the received option}}.

RQ_COR_1316 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶10, 12 *Type:* SHOULD *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid and authenticated (e.g., via IPSec [RFC 2402]) Router Advertisement where the prefix advertised in the Prefix-Information Option matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface. The StoredLifetime is less than or equal to 2 hours and the received Lifetime is less than or equal to StoredLifetime.

Requirement: The implementation sets the StoredLifetime to the Lifetime in the received option.

RFC text: {{...e) If the advertised prefix matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface, the specific action to perform depends on the Valid Lifetime in the received advertisement and the Lifetime associated with the previously autoconfigured address (which we call StoredLifetime in the discussion that follows): ...2) If the StoredLifetime is less than or equal to 2 hours and the received Lifetime is less than or equal to StoredLifetime, ignore the prefix, unless the Router Advertisement from which his Prefix Information option was obtained has been authenticated (e.g., via IPSec [RFC 2402]). If the Router Advertisment was authenticated, the StoredLifetime should be set to the Lifetime in the received option}}.

RQ_COR_1317 Prefix Information Option [Process]

RFC 2462 *Clause:* 5.5.3 ¶10, 13 *Type:* SHOULD *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateless Autoconfiguration. The implementation receives a valid Router Advertisement where the prefix advertised in the Prefix-Information Option matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface. The received Valid Lifetime in the received advertisement is lower than 2 hours.

Requirement: The implementation sets the stored Lifetime [StoredLifetime] in the corresponding address to two hours.

RFC text: `{{...e) If the advertised prefix matches the prefix of an autoconfigured address (i.e., one obtained via stateless or stateful address autoconfiguration) in the list of addresses associated with the interface, the specific action to perform depends on the Valid Lifetime in the received advertisement and the Lifetime associated with the previously autoconfigured address (which we call StoredLifetime in the discussion that follows): ...3) Otherwise, reset the stored Lifetime in the corresponding address to two hours}}`.

RQ_COR_1318 address: Deprecated Address Use

RFC 2462 *Clause:* 5.5.4 ¶1 *Type:* MAY *applies to:* Node

Context: An address is deprecated.

Requirement: The implementation prevents any new communication from using a deprecated address.

RFC text: `{{An implementation MAY prevent any new communication from using a deprecated address, but system management MUST have the ability to disable such a facility, and the facility MUST be disabled by default.}}`.

RQ_COR_1319 address: Deprecated Address Use

RFC 2462 *Clause:* 5.5.4 ¶1 *Type:* MUST *applies to:* Node

Context: An address is deprecated. The implementation has the ability to prevent any new communication from using a deprecated address.

Requirement: The implementation's system management has the ability to disable preventing any new communication's use of a deprecated address.

RFC text: `{{An implementation MAY prevent any new communication from using a deprecated address, but system management MUST have the ability to disable such a facility, and the facility MUST be disabled by default}}`.

RQ_COR_1320 address: Deprecated Address Use

RFC 2462 *Clause:* 5.5.4 ¶1 *Type:* MUST *applies to:* Node

Context: An address is deprecated. The implementation prevents any new communication from using a deprecated address.

Requirement: By default, the implementation does not prevent any new communication's use of a deprecated address.

RFC text: `{{An implementation MAY prevent any new communication from using a deprecated address, but system management MUST have the ability to disable such a facility, and the facility MUST be disabled by default}}`.

RQ_COR_1323 Stateless and Stateful Autoconfig Simultaneous

RFC 2462 *Clause:* 5.6 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation simultaneously simultaneously uses both Stateful and Stateless Autoconfiguration.

Requirement: The implementation makes the union of all information received via the stateless and stateful protocols.

RFC text: It is possible for hosts to obtain address information using both stateless and stateful protocols since both may be enabled at the same time. It is also possible that the values of other configuration parameters such as MTU size and hop limit will be learned from both Router Advertisements and the stateful autoconfiguration protocol. If the same configuration information is provided by multiple sources, the value of this information should be consistent. However, it is not considered a fatal error if information received from multiple sources is inconsistent. `{{Hosts accept the union of all information received via the stateless and stateful protocols}}`. If inconsistent information is learned different sources, the most recently obtained values always have precedence over information learned earlier.

RQ_COR_1324 Stateless and Stateful Autoconfig Simultaneous

RFC 2462 *Clause:* 5.6 ¶1 *Type:* MUST *applies to:* Host

Context: The implementation simultaneously uses both Stateful and Stateless Autoconfiguration. The implementation receives inconsistent values of address and other configuration parameters from multiple sources of both Stateful and Stateless Autoconfiguration and makes their union.

Requirement: The implementation uses the most recently obtained values in forming the union of the address and other configuration parameters.

RFC text: It is possible for hosts to obtain address information using both stateless and stateful protocols since both may be enabled at the same time. It is also possible that the values of other configuration parameters such as MTU size and hop limit will be learned from both Router Advertisements and the stateful autoconfiguration protocol. If the same configuration information is provided by multiple sources, the value of this information should be consistent. However, it is not considered a fatal error if information received from multiple sources is inconsistent. Hosts accept the union of all information received via the stateless and stateful protocols. `{{If inconsistent information is learned different sources, the most recently obtained values always have precedence over information learned earlier}}`.

RQ_COR_1326 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 6 ¶2 *Type:* MAY *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation sends Neighbor Solicitations for a tentative address. Denial of Service attacks are possible.

Requirement: The implementation uses authenticated [RFC 2402] Neighbor Discovery packets.

RFC text: `{{The use of Duplicate Address Detection opens up the possibility of denial of service attacks. Any node can respond to Neighbor Solicitations for a tentative address, causing the other node to reject the address as a duplicate. This attack is similar to other attacks involving the spoofing of Neighbor Discovery messages and can be addressed by requiring that Neighbor Discovery packets be authenticated [RFC 2402]}}`.

RQ_COR_9009 address: Deprecated Address Use

RFC 2462 *Clause:* 2 ¶21 *Type:* MUST *applies to:* Node

Context: An address is deprecated.

Requirement: The implementation delivers the packets to the deprecated address.

RFC text: deprecated address - An address assigned to an interface whose use is discouraged, but not forbidden. A deprecated address should no longer be used as a source address in new communications, `{{but packets sent from or to deprecated addresses are delivered as expected}}`.

RQ_COR_9012 address: [Configure]

RFC 2462 *Clause:* 4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is capable of Stateless address autoconfiguration. The implementation is not on a multicast-capable link.

Requirement: The implementation does not start Stateless address autoconfiguration.

RFC text: `{{Autoconfiguration is performed only on multicast-capable links and begins when a multicast-capable interface is enabled, e.g., during system startup}}`.

RQ_COR_9013 address: Manual Configuration

RFC 2462 *Clause:* 4 ¶9 *Type:* SHOULD *applies to:* Node

Context: The implementation's addresses for an interface are assigned manually.

Requirement: The implementation individually tests all the manually assigned addresses for uniqueness using Duplicate Address Detection.

RFC text: For safety, all addresses must be tested for uniqueness prior to their assignment to an interface. `{{...In contrast, all addresses obtained manually or via stateful address autoconfiguration should be tested for uniqueness individually}}`.

RQ_COR_9014 Stateless Autoconfig

RFC 2462 *Clause:* 1 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation is capable of both Stateful and Stateless Autoconfiguration. The implementation receives a Router Advertisment message requiring stateless autoconfiguration.

Requirement: The implementation executes stateless autoconfiguration.

RFC text: The stateless approach is used when a site is not particularly concerned with the exact addresses hosts use, so long as they are unique and properly routable. The stateful approach is used when a site requires tighter control over exact address assignments. `{{Both stateful and stateless address autoconfiguration may be used simultaneously}}`. The site administrator specifies which type of autoconfiguration to use through the setting of appropriate fields in Router Advertisement messages [DISCOVERY].

RQ_COR_9015 Stateless and Stateful Autoconfig Simultaneous

RFC 2462 *Clause:* 1 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation uses both Stateful and Stateless Autoconfiguration. The implementation receives a Router Advertisment message requiring stateful autoconfiguration.

Requirement: The implementation executes Stateful Autoconfiguration.

RFC text: The stateless approach is used when a site is not particularly concerned with the exact addresses hosts use, so long as they are unique and properly routable. The stateful approach is used when a site requires tighter control over exact address assignments. `{{Both stateful and stateless address autoconfiguration may be used simultaneously}}`. The site administrator specifies which type of autoconfiguration to use through the setting of appropriate fields in Router Advertisement messages [DISCOVERY].

RQ_COR_9016 Stateless Autoconfig

RFC 2462 *Clause:* 5.3 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The interface is reinitialized after a temporary interface failure.

Requirement: The implementation enables the interface and forms a link-local address.

RFC text: `{{A node forms a link-local address whenever an interface becomes enabled. An interface may become enabled after any of the following events: - The interface is initialized at system startup time. - The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management. - The interface attaches to a link for the first time. - The interface becomes enabled by system management after having been administratively disabled}}`.

RQ_COR_9017 Stateless Autoconfig

RFC 2462 *Clause:* 5.3 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The interface is reinitialized after after being temporarily disabled by system management.

Requirement: The implementation enables the interface and forms a link-local address.

RFC text: `{{A node forms a link-local address whenever an interface becomes enabled. An interface may become enabled after any of the following events: - The interface is initialized at system startup time. - The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management. - The interface attaches to a link for the first time. - The interface becomes enabled by system management after having been administratively disabled}}`.

RQ_COR_9018 Stateless Autoconfig

RFC 2462 *Clause:* 5.3 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. The interface attaches to a link for the first time.

Requirement: The implementation enables the interface and forms a link-local address.

RFC text: `{{A node forms a link-local address whenever an interface becomes enabled. An interface may become enabled after any of the following events: - The interface is initialized at system startup time. - The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management. - The interface attaches to a link for the first time. - The interface becomes enabled by system management after having been administratively disabled}}`.

RQ_COR_9019 Stateless Autoconfig

RFC 2462 *Clause:* 5.3 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration. System management re-enables an interface after being administratively disabled.

Requirement: The implementation enables the interface and forms a link-local address.

RFC text: `{{A node forms a link-local address whenever an interface becomes enabled. An interface may become enabled after any of the following events: - The interface is initialized at system startup time. - The interface is reinitialized after a temporary interface failure or after being temporarily disabled by system management. - The interface attaches to a link for the first time. - The interface becomes enabled by system management after having been administratively disabled}}`.

RQ_COR_9020 Neighbor Discovery - Invalid ND Message

RFC 2462 *Clause:* 5.4.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration and receives Advertisement messages that do not pass the validity checks specified in [DISCOVERY].

Requirement: The implementation silently discards the messages.

RFC text: `{{A node MUST silently discard any Neighbor Solicitation or Advertisement message that does not pass the validity checks specified in [DISCOVERY]}}`. A solicitation that passes these validity checks is called a valid solicitation or valid advertisement.

RQ_COR_9021 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration.

Requirement: The implementation joins the solicited-node multicast address before sending a Neighbor Solicitation.

RFC text: `{{Before sending a Neighbor Solicitation, an interface MUST join the all-nodes multicast address and the solicited-node multicast address of the tentative address}}`. The former insures that the node receives Neighbor Advertisements from other nodes already using the address; the latter insures that two nodes attempting to use the same address simultaneously detect each other's presence.

RQ_COR_9022 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 5.4.5 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation discovers that a tentative address is duplicated.

Requirement: The implementation does not assign the tentative address to the interface.

RFC text: `{{A tentative address that is determined to be a duplicate as described above, MUST NOT be assigned to an interface and the node SHOULD log a system management error. If the address is a link-local address formed from an interface identifier, the interface SHOULD be disabled}}`.

RQ_COR_9023 Stateless Autoconfig

RFC 2462 *Clause:* 5.4.5 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Duplicate Address Detection in Stateless autoconfiguration. The implementation discovers that a tentative address is duplicated. The tentative address is link-local formed from an interface identifier.

Requirement: The link-local interface is disabled.

RFC text: `{{A tentative address that is determined to be a duplicate as described above, MUST NOT be assigned to an interface and the node SHOULD log a system management error. If the address is a link-local address formed from an interface identifier, the interface SHOULD be disabled}}`.

RQ_COR_9024 Stateful Autoconfig

RFC 2462 *Clause:* 5.5.2 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation is creating a global address and has determined that there are no routers on the link. An option exists for disabling stateful autoconfiguration for this situation but the system operator has not chosen to use the option.

Requirement: By default, the implementation attempts to use stateful autoconfiguration obtain addresses and other configuration information.

RFC text: `{{If a link has no routers, a host MUST attempt to use stateful autoconfiguration to obtain addresses and other configuration information. An implementation MAY provide a way to disable the invocation of stateful autoconfiguration in this case, but the default SHOULD be enabled}}`.

RQ_COR_9025 address: Global [Assign]

RFC 2462 *Clause:* 5.5.2 ¶1 *Type:* SHOULD *applies to:* Host

Context: The implementation is creating a global address and has determined that there are no routers on the link. An option exists for disabling stateful autoconfiguration for this situation. The system operator has chosen to use the option.

Requirement: The implementation does not attempt to use stateful autoconfiguration obtain addresses and other configuration information.

RFC text: `{{If a link has no routers, a host MUST attempt to use stateful autoconfiguration to obtain addresses and other configuration information. An implementation MAY provide a way to disable the invocation of stateful autoconfiguration in this case, but the default SHOULD be enabled}}`.

RQ_COR_9026 address: Duplicate Address Detection (DAD)

RFC 2462 *Clause:* 4 ¶2, 4, 5.4 *Type:* SHOULD *applies to:* Node

Context: The implementation uses Stateless address autoconfiguration and is generating a unique address. It has transmitted DupAddrDetectTransmits Neighbor Solicitations and waited RetransTimer milliseconds after the last Neighbor Solicitation transmission. No responses indicate the presence of a duplicate address.

Requirement: The implementation considers the address unique. The address is no longer tentative.

RFC text: {{An address is considered unique if none of the tests indicate the presence of a duplicate address within RetransTimer milliseconds after having sent DupAddrDetectTransmits Neighbor Solicitations...}}. See RQ_COR_1243.

RQ_COR_9027 O-Flag [Use of]

RFC 2462 *Clause:* 5.5.3 ¶2 *Type:* MUST *applies to:* Host

Context: The implementation is determining its global address for an interface using Stateful Autoconfiguration. The implementation receives a valid Router Advertisement with the O-flag remaining the same as in previous advertisements.

Requirement: The implementation continues using Stateful Autoconfiguration.

RFC text: An advertisement's O flag field is processed in an analogous manner. A host copies the value of the O flag into OtherConfigFlag. If the value of OtherConfigFlag changes from FALSE to TRUE, the host should invoke the stateful autoconfiguration protocol, requesting information (excluding addresses if ManagedFlag is set to FALSE). If the value of the OtherConfigFlag changes from TRUE to FALSE, the host should continue running the stateful address autoconfiguration protocol, i.e., the change in the value of OtherConfigFlag has no effect. {{If the value of the flag stays unchanged, no special action takes place}}. In particular, a host MUST NOT reinvoke stateful configuration if it is already participating in the stateful protocol as a result of an earlier advertisement.

4.9 Requirements extracted from RFC 2463

RQ_COR_1400 ICMPv6 Messages [Generate]

RFC 2463 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6.

Requirement: The implementation fully implements ICMPv6.

RFC text: {{ICMPv6 is an integral part of IPv6 and MUST be fully implemented by every IPv6 node}}.

RQ_COR_1402 ICMPv6 Error Messages [Generate]

RFC 2463 *Clause:* 2.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 error message.

Requirement: The highest-order bit of the ICMPv6 error message Type field is set to 0. Error messages have Type values in the range 0 to 127.

RFC text: {{ICMPv6 messages are grouped into two classes: error messages and informational messages. Error messages are identified as such by having a zero in the high-order bit of their message Type field values. Thus, error messages have message Types from 0 to 127; informational messages have message Types from 128 to 255}}.

RQ_COR_1403 ICMPv6 Information Messages [Generate]

RFC 2463 *Clause:* 2.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 informational message.

Requirement: The highest-order bit of the ICMPv6 informational message Type field is set to 1. Informational messages have Type values in the range 128 to 255.

RFC text: `{{ICMPv6 messages are grouped into two classes: error messages and informational messages. Error messages are identified as such by having a zero in the high-order bit of their message Type field values. Thus, error messages have message Types from 0 to 127; informational messages have message Types from 128 to 255}}`.

RQ_COR_1404 ICMPv6 Messages [Generate]

RFC 2463 *Clause:* 2.1 ¶7-11 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 message.

Requirement: The ICMPv6 message is preceded by an IPv6 header and zero or more IPv6 extension headers. The ICMPv6 header is identified by a Next Header value of 58 in the header immediately preceding the ICMPv6 header.

RFC text: `{{Every ICMPv6 message is preceded by an IPv6 header and zero or more IPv6 extension headers. The ICMPv6 header is identified by a Next Header value of 58 in the immediately preceding header}}`. `{{The ICMPv6 messages have the following general format: The type field indicates the type of the message. Its value determines the format of the remaining data. The code field depends on the message type. It is used to create an additional level of message granularity. The checksum field is used to detect data corruption in the ICMPv6 message and parts of the IPv6 header}}`.

RQ_COR_1405 Checksum [Compute]

RFC 2463 *Clause:* 2.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 message.

Requirement: The implementation calculates the checksum including both the Source and Destination IPv6 Addresses in the IPv6 header.

RFC text: `{{A node that sends an ICMPv6 message has to determine both the Source and Destination IPv6 Addresses in the IPv6 header before calculating the checksum}}`.

RQ_COR_1406 ICMPv6 Message [Determine Source Address]

RFC 2463 *Clause:* 2.2 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation has more than one unicast address. The implementation receives a message with one of these unicast addresses as a Destination Address. The message provokes an ICMPv6 message as a response.

Requirement: The implementation uses the same received message's Destination Address as the Source Address for the ICMPv6 reply.

RFC text: If the node has more than one unicast address, it must choose the Source Address of the message as follows: `{{(a) If the message is a response to a message sent to one of the node's unicast addresses, the Source Address of the reply must be that same address}}`. (b) If the message is a response to a message sent to a multicast or anycast group in which the node is a member, the Source Address of the reply must be a unicast address belonging to the interface on which the multicast or anycast packet was received. (c) If the message is a response to a message sent to an address that does not belong to the node, the Source Address should be that unicast address belonging to the node that will be most helpful in diagnosing the error. For example, if the message is a response to a packet forwarding action that cannot complete successfully, the Source Address should be a unicast address belonging to the interface on which the packet forwarding failed. (d) Otherwise, the node's routing table must be examined to determine which interface will be used to transmit the message to its destination, and a unicast address belonging to that interface must be used as the Source Address of the message.

RQ_COR_1407 ICMPv6 Message [Determine Source Address]

RFC 2463 *Clause:* 2.2 ¶1, 3 *Type:* MUST *applies to:* Node

Context: The implementation has more than one unicast address. The implementation receives a message with a Destination Address of a multicast or anycast group to which the implementation belongs. The received message provokes an ICMPv6 message as a response.

Requirement: The Source Address of the ICMPv6 reply generated by the implementation is a unicast address belonging to the interface on which the multicast or anycast packet was received.

RFC text: If the node has more than one unicast address, it must choose the Source Address of the message as follows: (a) If the message is a response to a message sent to one of the node's unicast addresses, the Source Address of the reply must be that same address. `{{(b) If the message is a response to a message sent to a multicast or anycast group in which the node is a member, the Source Address of the reply must be a unicast address belonging to the interface on which the multicast or anycast packet was received}}`. (c) If the message is a response to a message sent to an address that does not belong to the node, the Source Address should be that unicast address belonging to the node that will be most helpful in diagnosing the error. For example, if the message is a response to a packet forwarding action that cannot complete successfully, the Source Address should be a unicast address belonging to the interface on which the packet forwarding failed. (d) Otherwise, the node's routing table must be examined to determine which interface will be used to transmit the message to its destination, and a unicast address belonging to that interface must be used as the Source Address of the message.

RQ_COR_1408 ICMPv6 Message [Determine Source Address]

RFC 2463 *Clause:* 2.2 ¶1, 4 *Type:* SHOULD *applies to:* Node

Context: The implementation has more than one unicast address. The implementation receives a message with a Destination Address that does not belong to the implementation. The received message provokes an ICMPv6 message as a response.

Requirement: The Source Address of the ICMPv6 reply generated by the implementation is the unicast address belonging to the node that will be most helpful in diagnosing the error.

RFC text: If the node has more than one unicast address, it must choose the Source Address of the message as follows: (a) If the message is a response to a message sent to one of the node's unicast addresses, the Source Address of the reply must be that same address. (b) If the message is a response to a message sent to a multicast or anycast group in which the node is a member, the Source Address of the reply must be a unicast address belonging to the interface on which the multicast or anycast packet was received. `{{(c) If the message is a response to a message sent to an address that does not belong to the node, the Source Address should be that unicast address belonging to the node that will be most helpful in diagnosing the error. For example, if the message is a response to a packet forwarding action that cannot complete successfully, the Source Address should be a unicast address belonging to the interface on which the packet forwarding failed}}`. (d) Otherwise, the node's routing table must be examined to determine which interface will be used to transmit the message to its destination, and a unicast address belonging to that interface must be used as the Source Address of the message.

RQ_COR_1409 ICMPv6 Message [Determine Source Address]

RFC 2463 *Clause:* 2.2 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation has more than one unicast address. An ICMPv6 message is required as a response to a received message. The received message does not fall into one of the following categories. (1) The received message's Destination Address is not one of the implementations assigned unicast addresses. (2) The received message's Destination Address is not the address of any multicast or anycast group to which the implementation belongs. (3) The received message's Destination Address does not belong to the implementation.

Requirement: The implementation examines its routing table to determine the interface to transmit the message to its destination. The Source Address of the reply is a unicast address belonging to the selected interface.

RFC text: If the node has more than one unicast address, it must choose the Source Address of the message as follows: (a) If the message is a response to a message sent to one of the node's unicast addresses, the Source Address of the reply must be that same address. (b) If the message is a response to a message sent to a multicast or anycast group in which the node is a member, the Source Address of the reply must be a unicast address belonging to the interface on which the multicast or anycast packet was received. (c) If the message is a response to a message sent to an address that does not belong to the node, the Source Address should be that unicast address belonging to the node that will be most helpful in diagnosing the error. For example, if the message is a response to a packet forwarding action that cannot complete successfully, the Source Address should be a unicast address belonging to the interface on which the packet forwarding failed. `{{(d) Otherwise, the node's routing table must be examined to determine which interface will be used to transmit the message to its destination, and a unicast address belonging to that interface must be used as the Source Address of the message}}`.

RQ_COR_1410 Checksum [Compute]

RFC 2463 *Clause:* 2.3 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 message.

Requirement: The implementation calculates the checksum as the 16-bit one's complement of the one's complement sum of the entire ICMPv6 message starting with the ICMPv6 message type field, prepended with a "pseudo-header" [RFC 2460] of IPv6 header fields. When computing the checksum, the Next Header value in the pseudo-header set to 58 and the checksum field is set to zero.

RFC text: `{{The checksum is the 16-bit one's complement of the one's complement sum of the entire ICMPv6 message starting with the ICMPv6 message type field, prepended with a "pseudo-header" of IPv6 header fields, as specified in [IPv6, section 8.1]. The Next Header value used in the pseudo-header is 58. (NOTE: the inclusion of a pseudo-header in the ICMPv6 checksum is a change from IPv4; see [IPv6] for the rationale for this change.) ...For computing the checksum, the checksum field is set to zero}}`.

RQ_COR_1411 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 error message of unknown type.

Requirement: The implementation passes the message to the upper layer.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC-1122]): `{{...(a) If an ICMPv6 error message of unknown type is received, it MUST be passed to the upper layer}}`.

RQ_COR_1412 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 3 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 informational message of unknown type.

Requirement: The implementation silently discards the message.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC-1122]): `{{...(b) If an ICMPv6 informational message of unknown type is received, it MUST be silently discarded}}`.

RQ_COR_1413 ICMPv6 Error Messages [Generate]

RFC 2463 *Clause:* 2.4 ¶1, 4 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 error message.

Requirement: The implementation includes in every ICMPv6 error message (type < 128) as much of the IPv6 offending packet as will fit without making the error message packet exceed the minimum IPv6 MTU.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC-1122]): `{{...(c) Every ICMPv6 error message (type < 128) includes as much of the IPv6 offending (invoking) packet (the packet that caused the error) as will fit without making the error message packet exceed the minimum IPv6 MTU [IPv6]}}`.

RQ_COR_1414 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 5 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 error message where the implementation is required to pass the same message to the upper layer.

Requirement: The implementation extracts the upper-layer protocol type from the body of the ICMPv6 error message in order to select the appropriate upper-layer process to handle the error.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): `{{...(d) In those cases where the internet-layer protocol is required to pass an ICMPv6 error message to the upper-layer process, the upper-layer protocol type is extracted from the original packet (contained in the body of the ICMPv6 error message) and used to select the appropriate upper-layer process to handle the error}}`.

RQ_COR_1415 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 5-6 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 error message where the implementation is required to pass the same message to the upper layer. The implementation is unable to extract the upper-layer protocol type from the original packet because of truncation of the original packet to meet the minimum IPv6 MTU [IPv6] limit.

Requirement: The implementation silently drops the error message after any IPv6-layer processing.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): `{{...(d) In those cases where the internet-layer protocol is required to pass an ICMPv6 error message to the upper-layer process, the upper-layer protocol type is extracted from the original packet (contained in the body of the ICMPv6 error message) and used to select the appropriate upper-layer process to handle the error}}`. `{{...If the original packet had an unusually large amount of extension headers, it is possible that the upper-layer protocol type may not be present in the ICMPv6 message, due to truncation of the original packet to meet the minimum IPv6 MTU [IPv6] limit. In that case, the error message is silently dropped after any IPv6-layer processing}}`.

RQ_COR_1416 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7-8 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 error message.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: `{{...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message}}`, or (e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10), or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or (e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address.

RQ_COR_1417 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 9 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet destined to an IPv6 multicast address. The received packet is not a Packet Too Big Message.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or `{{(e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10)}}`, or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or (e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address.

RQ_COR_1418 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 9 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet destined to an IPv6 multicast address. The received packet is not a Parameter Problem Message reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or `{{(e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10)}}`, or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or (e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address.

RQ_COR_1419 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 10 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet sent on a link-layer multicast address. The packet is neither a Packet Too Big Message nor a Parameter Problem Message reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or (e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - Section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - Section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10), or `{{(e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too)}}`, or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or (e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address.

RQ_COR_1421 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 11 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet to an link-layer broadcast address. The packet is neither a Packet Too Big Message nor a Parameter Problem Message reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or (e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - Section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - Section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10), or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or `{{(e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too)}}`, or (e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address.

RQ_COR_1423 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 12 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet whose source address does not uniquely identify a single node.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or (e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - Section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - Section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10), or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or `{{(e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address}}`.

RQ_COR_1424 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 12 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet whose source address is the IPv6 Unspecified Address.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or (e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - Section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - Section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10), or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or

RQ_COR_1425 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 12 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet message whose source address is an IPv6 multicast address.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or (e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - Section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - Section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10), or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or `{{(e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address}}`.

RQ_COR_1426 ICMPv6 Messages [Process]

RFC 2463 *Clause:* 2.4 ¶1, 7, 12 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet whose source address is an address known by the implementation to be an IPv6 anycast address.

Requirement: The implementation does not send an ICMPv6 error message as response.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(e) An ICMPv6 error message MUST NOT be sent as a result of receiving: (e.1) an ICMPv6 error message, or (e.2) a packet destined to an IPv6 multicast address (there are two exceptions to this rule: (1) the Packet Too Big Message - Section 3.2 - to allow Path MTU discovery to work for IPv6 multicast, and (2) the Parameter Problem Message, Code 2 - Section 3.4 - reporting an unrecognized IPv6 option that has the Option Type highest-order two bits set to 10), or (e.3) a packet sent as a link-layer multicast, (the exception from e.2 applies to this case too), or (e.4) a packet sent as a link-layer broadcast, (the exception from e.2 applies to this case too), or `{{(e.5) a packet whose source address does not uniquely identify a single node -- e.g., the IPv6 Unspecified Address, an IPv6 multicast address, or an address known by the ICMP message sender to be an IPv6 anycast address}}`.

RQ_COR_1427 ICMPv6 Bandwidth and Forwarding Costs [Limit]

RFC 2463 *Clause:* 2.4 ¶1, 13 *Type:* MUST *applies to:* Node

Context: The implementation sends ICMPv6 error messages.

Requirement: The implementation controls the rate of ICMPv6 error messages it sends in order to limit the bandwidth and forwarding costs incurred with sending ICMPv6 error messages.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...`{{(f) Finally, in order to limit the bandwidth and forwarding costs incurred sending ICMPv6 error messages, an IPv6 node MUST limit the rate of ICMPv6 error messages it sends. This situation may occur when a source sending a stream of erroneous packets fails to heed the resulting ICMPv6 error messages}}`. There are a variety of ways of implementing the rate-limiting function, for example: (f.1) Timer-based - for example, limiting the rate of transmission of error messages to a given source, or to any source, to at most once every T milliseconds. (f.2) Bandwidth-based - for example, limiting the rate at which error messages are sent from a particular interface to some fraction F of the attached link's bandwidth. The limit parameters (e.g., T or F in the above examples) MUST be configurable for the node, with a conservative default value (e.g., T = 1 second, NOT 0 seconds, or F = 2 percent, NOT 100 percent).

RQ_COR_1428 ICMPv6 Bandwidth and Forwarding Costs [Limit]

RFC 2463 *Clause:* 2.4 ¶1, 13-17 *Type:* MAY *applies to:* Node

Context: The implementation controls the rate of ICMPv6 error messages it sends in order to limit the bandwidth and forwarding costs incurred sending ICMPv6 error messages.

Requirement: The implementation uses a rate-limiting function which is Timer-based, limiting the rate of transmission of error messages to a given source, or to any source, to at most once every T milliseconds.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(f) Finally, in order to limit the bandwidth and forwarding costs incurred sending ICMPv6 error messages, an IPv6 node MUST limit the rate of ICMPv6 error messages it sends. This situation may occur when a source sending a stream of erroneous packets fails to heed the resulting ICMPv6 error messages. There are a variety of ways of implementing the rate-limiting function, for example: {{(f.1) Timer-based - for example, limiting the rate of transmission of error messages to a given source, or to any source, to at most once every T milliseconds. (f.2) Bandwidth-based - for example, limiting the rate at which error messages are sent from a particular interface to some fraction F of the attached link's bandwidth. The limit parameters (e.g., T or F in the above examples) MUST be configurable for the node, with a conservative default value (e.g., T = 1 second, NOT 0 seconds, or F = 2 percent, NOT 100 percent)}}.

RQ_COR_1429 ICMPv6 Bandwidth and Forwarding Costs [Limit]

RFC 2463 *Clause:* 2.4 ¶1, 13-17 *Type:* MUST *applies to:* Node

Context: The implementation limits the rate of ICMPv6 error messages it sends using a rate-limiting function which is Timer-based, limiting the rate of transmission of error messages to a given source, or to any source, to at most once every T milliseconds.

Requirement: The limit parameter T is configurable for the implementation, with a conservative default value.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(f) Finally, in order to limit the bandwidth and forwarding costs incurred sending ICMPv6 error messages, an IPv6 node MUST limit the rate of ICMPv6 error messages it sends. This situation may occur when a source sending a stream of erroneous packets fails to heed the resulting ICMPv6 error messages. There are a variety of ways of implementing the rate-limiting function, for example: {{(f.1) Timer-based - for example, limiting the rate of transmission of error messages to a given source, or to any source, to at most once every T milliseconds. (f.2) Bandwidth-based - for example, limiting the rate at which error messages are sent from a particular interface to some fraction F of the attached link's bandwidth. The limit parameters (e.g., T or F in the above examples) MUST be configurable for the node, with a conservative default value (e.g., T = 1 second, NOT 0 seconds, or F = 2 percent, NOT 100 percent)}}.

RQ_COR_1430 ICMPv6 Bandwidth and Forwarding Costs [Limit]

RFC 2463 *Clause:* 2.4 ¶1, 13-17 *Type:* MAY *applies to:* Node

Context: The implementation limits the rate of ICMPv6 error messages it sends in order to limit the bandwidth and forwarding costs incurred sending ICMPv6 error messages.

Requirement: The implementation uses a rate-limiting function which is Bandwidth-based, limiting the rate at which error messages are sent from a particular interface to some fraction F of the attached link's bandwidth.

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(f) Finally, in order to limit the bandwidth and forwarding costs incurred sending ICMPv6 error messages, an IPv6 node MUST limit the rate of ICMPv6 error messages it sends. This situation may occur when a source sending a stream of erroneous packets fails to heed the resulting ICMPv6 error messages. There are a variety of ways of implementing the rate-limiting function, for example: `{{(f.1) Timer-based - for example, limiting the rate of transmission of error messages to a given source, or to any source, to at most once every T milliseconds. (f.2) Bandwidth-based - for example, limiting the rate at which error messages are sent from a particular interface to some fraction F of the attached link's bandwidth. The limit parameters (e.g., T or F in the above examples) MUST be configurable for the node, with a conservative default value (e.g., T = 1 second, NOT 0 seconds, or F = 2 percent, NOT 100 percent)}}`.

RQ_COR_1431 ICMPv6 Bandwidth and Forwarding Costs [Limit]

RFC 2463 *Clause:* 2.4 ¶1, 13-17 *Type:* MUST *applies to:* Node

Context: The implementation limits the rate of ICMPv6 error messages it sends using a rate-limiting function which is Bandwidth-based, limiting the rate at which error messages are sent from a particular interface to some fraction F of the attached link's bandwidth.

Requirement: The limit parameter F is configurable for the implementation, with a conservative default value (e.g., F = 2 percent, NOT 100 percent).

RFC text: Implementations MUST observe the following rules when processing ICMPv6 messages (from [RFC 1122]): ...(f) Finally, in order to limit the bandwidth and forwarding costs incurred sending ICMPv6 error messages, an IPv6 node MUST limit the rate of ICMPv6 error messages it sends. This situation may occur when a source sending a stream of erroneous packets fails to heed the resulting ICMPv6 error messages. There are a variety of ways of implementing the rate-limiting function, for example: `{{(f.1) Timer-based - for example, limiting the rate of transmission of error messages to a given source, or to any source, to at most once every T milliseconds. (f.2) Bandwidth-based - for example, limiting the rate at which error messages are sent from a particular interface to some fraction F of the attached link's bandwidth. The limit parameters (e.g., T or F in the above examples) MUST be configurable for the node, with a conservative default value (e.g., T = 1 second, NOT 0 seconds, or F = 2 percent, NOT 100 percent)}}`.

RQ_COR_1432 Destination Unreachable [Generate]

RFC 2463 *Clause:* 3.1 ¶1-8 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Destination Unreachable Message.

Requirement: The implementation includes in such ICMPv6 error message the following information: In IPv6 Fields: (a) Destination Address copied from the Source Address field of the invoking packet; In ICMPv6 Fields: (b) Type Field: 1, (c) Code Field: Description of the encountered problem (0 - no route to destination 1 - communication with destination administratively prohibited 2 - (not assigned) 3 - address unreachable 4 - port unreachable); (d) Checksum Field: the calculated checksum; (e) Unused Field: This field is unused and initialized to zero; (f) Message Body Field: as much of invoking packet as will fit without the ICMPv6 packet exceeding the minimum IPv6 MTU.

RFC text: `{{3.1 Destination Unreachable Message. IPv6 Fields: Destination Address Copied from the Source Address field of the invoking packet. ICMPv6 Fields: Type 1, Code 0 - no route to destination 1 - communication with destination administratively prohibited 2 - (not assigned) 3 - address unreachable 4 - port unreachable, Unused This field is unused for all code values. It must be initialized to zero by the sender and ignored by the receiver, Message Body As much of invoking packet as will fit without the ICMPv6 packet exceeding the minimum IPv6 MTU [IPv6]}}`.

RQ_COR_1433 Destination Unreachable [Process]

RFC 2463 *Clause:* 3.1 ¶1-8 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Destination Unreachable Message with a value not equal to zero in the Unused field.

Requirement: The implementation ignores the Unused field in the such ICMPv6 error message.

RFC text: `{{3.1 Destination Unreachable Message. IPv6 Fields: Destination Address Copied from the Source Address field of the invoking packet. ICMPv6 Fields: Type 1, Code 0 - no route to destination 1 - communication with destination administratively prohibited 2 - (not assigned) 3 - address unreachable 4 - port unreachable, Unused This field is unused for all code values. It must be initialized to zero by the sender and ignored by the receiver, Message Body As much of invoking packet as will fit without the ICMPv6 packet exceeding the minimum IPv6 MTU [IPv6]}}`.

RQ_COR_1434 Destination Unreachable [Generate]

RFC 2463 *Clause:* 3.1 ¶10 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a packet that cannot be delivered to its destination address for reasons other than congestion.

Requirement: The implementation generates an ICMPv6 Destination Unreachable Message.

RFC text: `{{A Destination Unreachable message SHOULD be generated by a router, or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion. (An ICMPv6 message MUST NOT be generated if a packet is dropped due to congestion.) }}`.

RQ_COR_1435 Destination Unreachable [Generate]

RFC 2463 *Clause:* 3.1 ¶10 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet that cannot be delivered to its destination address due to congestion.

Requirement: The implementation does not generates an ICMPv6 Destination Unreachable Message.

RFC text: `{{A Destination Unreachable message SHOULD be generated by a router, or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion. (An ICMPv6 message MUST NOT be generated if a packet is dropped due to congestion.)}}`.

RQ_COR_1436 Destination Unreachable Code Field Value

RFC 2463 *Clause:* 3.1 ¶10-11 *Type:* MUST *applies to:* Node

Context: The implementation does not hold a "default route" in its routing table. The implementation receives a packet that cannot be delivered to its destination address for reasons other than congestion. The implementation lacks a matching entry in the forwarding implementation's routing table.

Requirement: The implementation generates an ICMPv6 Destination Unreachable Message and sets the Code field to 0.

RFC text: `{{A Destination Unreachable message SHOULD be generated by a router, or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion. ...If the reason for the failure to deliver is lack of a matching entry in the forwarding node's routing table, the Code field is set to 0 (NOTE: this error can occur only in nodes that do not hold a "default route" in their routing tables)}}`.

RQ_COR_1437 Destination Unreachable Code Field Value

RFC 2463 *Clause:* 3.1 ¶10, 12 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet that cannot be delivered to its destination address due to administrative prohibition, e.g., a "firewall filter".

Requirement: The implementation generates an ICMPv6 Destination Unreachable Message where the Code field is set to 1.

RFC text: `{{A Destination Unreachable message SHOULD be generated by a router, or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion. ...If the reason for the failure to deliver is administrative prohibition, e.g., a "firewall filter", the Code field is set to 1}}`.

RQ_COR_1438 Destination Unreachable Code Field Value

RFC 2463 *Clause:* 3.1 ¶10, 13 *Type:* MUST *applies to:* Node

Context: The implementation receives a packet that cannot be delivered to its destination address due to any other reason than the lacking of a matching entry in the routing table or administrative prohibition.

Requirement: The implementation generates an ICMPv6 Destination Unreachable Message where the Code field is set to 3.

RFC text: `{{A Destination Unreachable message SHOULD be generated by a router, or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion. ...If there is any other reason for the failure to deliver, e.g., inability to resolve the IPv6 destination address into a corresponding link address, or a link-specific problem of some sort, then the Code field is set to 3}}`.

RQ_COR_1441 Destination Unreachable Code Field Value

RFC 2463 *Clause:* 3.1 ¶10, 14 *Type:* SHOULD *applies to:* Node

Context: The implementation receives a packet for which the transport protocol (e.g., UDP) has no listener and the transport protocol has no alternative means to inform the sender.

Requirement: The implementation generates an ICMPv6 Destination Unreachable Message where the Code field is set to 4.

RFC text: `{{A destination node SHOULD send a Destination Unreachable message with Code 4 in response to a packet for which the transport protocol (e.g., UDP) has no listener, if that transport protocol has no alternative means to inform the sender}}`.

RQ_COR_1442 Destination Unreachable [Process]

RFC 2463 *Clause:* 3.1 ¶16 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Destination Unreachable message.

Requirement: The implementation notifies the upper-layer process.

RFC text: `{{A node receiving the ICMPv6 Destination Unreachable message MUST notify the upper-layer process}}`.

RQ_COR_1443 Packet Too Big Message [Generate]

RFC 2463 *Clause:* 3.2 ¶1-7 *Type:* MUST *applies to:* Router

Context: The implementation generates an ICMPv6 Packet Too Big Message.

Requirement: The implementation includes in such ICMPv6 error message the following information: In IPv6 Fields: (a) Destination Address copied from the Source Address field of the received offending packet. ICMPv6 Fields: (b) Type Field set to 2. (c) Code Field set to zero. (d) Checksum Field set to the calculated checksum. (e) MTU Field set to Maximum Transmission Unit of the next-hop link. (f) Message Body Field contains the copy of as much of invoking packet as will fit without the ICMPv6 packet exceeding the minimum IPv6 MTU.

RFC text: `{{3.2 Packet Too Big Message IPv6 Fields: Destination Address Copied from the Source Address field of the invoking packet. ICMPv6 Fields: Type 2, Code Set to 0 (zero) by the sender and ignored by the receiver, MTU The Maximum Transmission Unit of the next-hop link}}`.

RQ_COR_1444 Packet Too Big Message [Process]

RFC 2463 *Clause:* 3.2 ¶1-7 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Packet Too Big Message with the Code field set a value other than zero.

Requirement: The implementation ignores the Code field.

RFC text: `{{3.2 Packet Too Big Message IPv6 Fields: Destination Address Copied from the Source Address field of the invoking packet. ICMPv6 Fields: Type 2, Code Set to 0 (zero) by the sender and ignored by the receiver, MTU The Maximum Transmission Unit of the next-hop link}}`.

RQ_COR_1445 Packet Too Big Message [Generate]

RFC 2463 *Clause:* 3.2 ¶9 *Type:* MUST *applies to:* Router

Context: The implementation receives a packet that it cannot forward because the packet is larger than the MTU of the outgoing link.

Requirement: The implementation generates an ICMPv6 Packet Too Big Message.

RFC text: {{A Packet Too Big MUST be sent by a router in response to a packet that it cannot forward because the packet is larger than the MTU of the outgoing link. The information in this message is used as part of the Path MTU Discovery process [PMTU]}}.

RQ_COR_1446 Packet Too Big Message [Process]

RFC 2463 *Clause:* 3.1 ¶12 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Packet Too Big Message.

Requirement: The implementation passes it to the upper-layer process.

RFC text: {{An incoming Packet Too Big message MUST be passed to the upper-layer process}}.

RQ_COR_1447 Time Exceeded Message [Generate]

RFC 2463 *Clause:* 3.3 ¶1-6 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Time Exceeded Message.

Requirement: The implementation includes in such ICMPv6 error message the following information: In the IPv6 Fields: (a) Destination Address copied from the Source Address field of the offending packet. In the ICMPv6 Fields: (b) Type Field set to 3. (c) Code Field set to either 0 (hop limit exceeded in transit) or 1 (fragment reassembly time exceeded). (d) Checksum Field set to the calculated checsum. (e) Unused Field set to zero. (f) Message Body Field contains as much of invoking packet as will fit without the ICMPv6 packet exceeding the minimum IPv6 MTU.

RFC text: {{3.3 Time Exceeded Message IPv6 Fields: Destination Address Copied from the Source Address field of the invoking packet. ICMPv6 Fields: Type 3, Code 0 - hop limit exceeded in transit [or] 1 - fragment reassembly time exceeded, Unused This field is unused for all code values. It must be initialized to zero by the sender and ignored by the receiver. ...Message Body Field: copied as much of invoking packet as will fit without the ICMPv6 packet exceeding the minimum IPv6 MTU}}.

RQ_COR_1448 Time Exceeded Message [Process]

RFC 2463 *Clause:* 3.3 ¶1-6 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Time Exceeded Message with a value other than zero in the Unused field.

Requirement: The implementation ignores the Unused field in the such ICMPv6 error message.

RFC text: {{3.3 Time Exceeded Message IPv6 Fields: Destination Address Copied from the Source Address field of the invoking packet. ICMPv6 Fields: Type 3, Code 0 - hop limit exceeded in transit [or] 1 - fragment reassembly time exceeded, Unused This field is unused for all code values. It must be initialized to zero by the sender and ignored by the receiver}}.

RQ_COR_1449 Time Exceeded Message [Generate]

RFC 2463 *Clause:* 3.3 ¶8 *Type:* MUST *applies to:* Router

Context: The implementation receives a packet with a Hop Limit of zero.

Requirement: The implementation discards the packet and sends an ICMPv6 Time Exceeded message with Code 0 to the source of the packet.

RFC text: `{{If a router receives a packet with a Hop Limit of zero, or a router decrements a packet's Hop Limit to zero, it MUST discard the packet and send an ICMPv6 Time Exceeded message with Code 0 to the source of the packet. This indicates either a routing loop or too small an initial Hop Limit value}}`.

RQ_COR_1450 Time Exceeded Message [Generate]

RFC 2463 *Clause:* 3.3 ¶8 *Type:* MUST *applies to:* Router

Context: The implementation receives a packet with the Hop Limit set to 1.

Requirement: The implementation discards the packet and sends an ICMPv6 Time Exceeded message with Code 0 to the source of the packet.

RFC text: `{{If a router receives a packet with a Hop Limit of zero, or a router decrements a packet's Hop Limit to zero, it MUST discard the packet and send an ICMPv6 Time Exceeded message with Code 0 to the source of the packet. This indicates either a routing loop or too small an initial Hop Limit value}}`.

RQ_COR_1451 Time Exceeded Message [Generate]

RFC 2463 *Clause:* 3.3 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Time Exceeded Message.

Requirement: The implementation selects the Source Address of this message following the rules defined in RFC 2463 section 2.2 [Related to implementations that have more than one unicast address].

RFC text: `{{The rules for selecting the Source Address of this message are defined in section 2.2.}}`. The corresponding requirements for these rules are RQ_COR_1406, RQ_COR_1407, RQ_COR_1408, and RQ_COR_1409.

RQ_COR_1452 Time Exceeded Message [Process]

RFC 2463 *Clause:* 3.3 ¶11 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Time Exceeded Message.

Requirement: The implementation passes it to the upper-layer process.

RFC text: `{{An incoming Time Exceeded message MUST be passed to the upper-layer process}}`.

RQ_COR_1453 Parameter Problem Message [Generate]

RFC 2463 *Clause:* 3.4 ¶1-7 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Parameter Problem Message.

Requirement: The implementation includes in such ICMPv6 error message the following information: IPv6 Fields: (a) Destination Address copied from the Source Address field of the received offending packet. ICMPv6 Fields: (b) Type Field set to 4. (c) Code Field set according the description of the encountered problem (0 - erroneous header field encountered 1 - unrecognized Next Header type encountered 2 - unrecognized IPv6 option encountered). (d) Checksum Field set to the calculated checksum. (e) Pointer Field Identifies the octet offset within the invoking packet where the error was detected. (f) Message Body Field As much of invoking packet as will fit without the ICMPv6 packet exceeding the minimum IPv6 MTU.

RFC text: {{3.4 Parameter Problem Message IPv6 Fields: Destination Address Copied from the Source Address field of the invoking packet. ICMPv6 Fields: Type 4, Code 0 - erroneous header field encountered 1 - unrecognized Next Header type encountered 2 - unrecognized IPv6 option encountered, Pointer Identifies the octet offset within the invoking packet where the error was detected. The pointer will point beyond the end of the ICMPv6 packet if the field in error is beyond what can fit in the maximum size of an ICMPv6 error message}}.

RQ_COR_1454 Parameter Problem Message [Generate]

RFC 2463 *Clause:* 3.4 ¶7 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Parameter Problem Message where the detected error is beyond what can fit in the maximum size of an ICMPv6 error message.

Requirement: The implementation calculates Pointer field of the ICMPv6 error message to point to the error's location even if the location is beyond the end of the ICMPv6 error message.

RFC text: Pointer Identifies the octet offset within the invoking packet where the error was detected. {{The pointer will point beyond the end of the ICMPv6 packet if the field in error is beyond what can fit in the maximum size of an ICMPv6 error message}}.

RQ_COR_1455 IPv6 Header [Process]

RFC 2463 *Clause:* 3.4 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet. The implementation finds a problem with a field in the IPv6 header such that it cannot complete processing the packet.

Requirement: The implementation discards the packet.

RFC text: {{If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it MUST discard the packet and SHOULD send an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem}}.

RQ_COR_1456 IPv6 Header [Process]

RFC 2463 *Clause:* 3.4 ¶9 *Type:* SHOULD *applies to:* Node

Context: The implementation processes a packet. The implementation finds a problem with a field in the IPv6 header such that it cannot complete processing the packet. The implementation discards the packet.

Requirement: The implementation sends an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem.

RFC text: {{If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it MUST discard the packet and SHOULD send an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem}}.

RQ_COR_1457 Extension Headers [Process]

RFC 2463 *Clause:* 3.4 ¶10 *Type:* SHOULD *applies to:* Node

Context: The implementation processes a packet. The implementation finds that the IPv6 extension header following the IPv6 header of the packet holds an unrecognized Next Header field value. The implementation discards the packet.

Requirement: The implementation sends an ICMPv6 Parameter Problem message to the packet's source, containing Type field = 4, Code field = 1, and Pointer field = 40.

RFC text: {{The pointer identifies the octet of the original packet's header where the error was detected. For example, an ICMPv6 message with Type field = 4, Code field = 1, and Pointer field = 40 would indicate that the IPv6 extension header following the IPv6 header of the original packet holds an unrecognized Next Header field value}}.

RQ_COR_1458 Parameter Problem Message [Process]

RFC 2463 *Clause:* 3.4 ¶12 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Parameter Problem Message.

Requirement: The implementation notifies the upper-layer process.

RFC text: {{A node receiving this ICMPv6 message MUST notify the upper-layer process}}.

RQ_COR_1459 Echo Request [Generate]

RFC 2463 *Clause:* 4.1 ¶1-8 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Echo Request Message.

Requirement: The implementation includes in the Echo Request the following information: IPv6 Fields: (a) Destination Address - Any legal IPv6 address. ICMPv6 Fields: (b) Type Field set to 128. (c) Code Field set to 0. (d) Checksum Field set to the calculated checksum. (e) Identifier Field - An identifier to aid in matching Echo Replies to this Echo Request. (f) Sequence Number Field - A sequence number to aid in matching Echo Replies to this Echo Request. (g) Data Field Zero or more octets of arbitrary data.

RFC text: {{4.1 Echo Request Message IPv6 Fields: Destination Address Any legal IPv6 address, ICMPv6 Fields: Type 128, Code 0, Identifier An identifier to aid in matching Echo Replies to this Echo Request. May be zero., Sequence Number A sequence number to aid in matching Echo Replies to this Echo Request. May be zero., Data Zero or more octets of arbitrary data}}.

RQ_COR_1460 Echo Request [Process]

RFC 2463 *Clause:* 4.1 ¶10 *Type:* MUST *applies to:* Node

Context: The implementation receives an Echo Request.

Requirement: The implementation sends an Echo Reply in response to the request.

RFC text: `{{Every node MUST implement an ICMPv6 Echo responder function that receives Echo Requests and sends corresponding Echo Replies}}`. A node SHOULD also implement an application-layer interface for sending Echo Requests and receiving Echo Replies, for diagnostic purposes.

RQ_COR_1461 ICMPv6 Messages [Generate]

RFC 2463 *Clause:* 4.1 ¶10 *Type:* SHOULD *applies to:* Node

Context: The implementation is IPv6 capable.

Requirement: The implementation has an application-layer interface for sending Echo Requests and receiving Echo Replies for diagnostic purposes.

RFC text: Every node MUST implement an ICMPv6 Echo responder function that receives Echo Requests and sends corresponding Echo Replies. `{{A node SHOULD also implement an application-layer interface for sending Echo Requests and receiving Echo Replies, for diagnostic purposes}}`.

RQ_COR_1462 Echo Request [Process]

RFC 2463 *Clause:* 4.1 ¶12 *Type:* MAY *applies to:* Node

Context: The implementation receives an ICMPv6 Echo Request Message.

Requirement: The implementation passes the Echo Request message to [upper-layer] processes receiving ICMP messages.

RFC text: `{{Echo Request messages MAY be passed to processes receiving ICMP messages}}`.

RQ_COR_1463 Echo Request [Process]

RFC 2463 *Clause:* 4.2 ¶1-8 *Type:* MUST *applies to:* Node

Context: The implementation has received an Echo Request message and generates an ICMPv6 Echo Reply Message.

Requirement: The implementation includes the following information in the Echo Reply: IPv6 Fields: (a) Destination Address - Copied from the Source Address field of the invoking Echo Request packet. In ICMPv6 Fields: (b) Type Field set to 129. (c) Code Field set to 0, (d) Checksum Field set to the calculated checksum. (e) Identifier Field - The identifier from the invoking Echo Request message. (f) Sequence Number Field - The sequence number from the invoking Echo Request message. (g) Data Field - The data from the invoking Echo Request message.

RFC text: `{{4.2 Echo Reply Message IPv6 Fields: Destination Address Copied from the Source Address field of the invoking Echo Request packet, ICMPv6 Fields: Type 129, Code 0, Identifier: The identifier from the invoking Echo Request message. Sequence Number: The sequence number from the invoking Echo Request message. Data: The data from the invoking Echo Request message}}`.

RQ_COR_1464 Echo Request [Process]

RFC 2463 *Clause:* 4.2 ¶11 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Echo Reply message in response to an Echo Request message sent to one of the implementation's unicast addresses.

Requirement: The implementation uses as Source Address of the Echo Reply the value of the Destination Address of the Echo Request message.

RFC text: `{{The source address of an Echo Reply sent in response to a unicast Echo Request message MUST be the same as the destination address of that Echo Request message}}`.

RQ_COR_1465 Echo Request [Process]

RFC 2463 *Clause:* 4.2 ¶12 *Type:* SHOULD *applies to:* Node

Context: The implementation receives an Echo Request message with a multicast address in the request's Destination Address field.

Requirement: The implementation sends an Echo Reply in response to the request.

RFC text: `{{An Echo Reply SHOULD be sent in response to an Echo Request message sent to an IPv6 multicast address}}`. The source address of the reply MUST be a unicast address belonging to the interface on which the multicast Echo Request message was received.

RQ_COR_1466 Echo Request [Process]

RFC 2463 *Clause:* 4.2 ¶12 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Echo Reply message in response to an Echo Request message sent to an multicast address.

Requirement: The implementation places into the reply's IP Header Source Address field a unicast address belonging to the Echo Request receiving interface.

RFC text: An Echo Reply SHOULD be sent in response to an Echo Request message sent to an IPv6 multicast address. `{{The source address of the reply MUST be a unicast address belonging to the interface on which the multicast Echo Request message was received}}`.

RQ_COR_1467 Echo Request [Process]

RFC 2463 *Clause:* 4.2 ¶13 *Type:* MUST *applies to:* Node

Context: The implementation generates an ICMPv6 Echo Reply message in response to an Echo Request message.

Requirement: The implementation returns entirely and unmodified in the ICMPv6 Echo Reply message the data received in the ICMPv6 Echo Request message.

RFC text: `{{The data received in the ICMPv6 Echo Request message MUST be returned entirely and unmodified in the ICMPv6 Echo Reply message}}`.

RQ_COR_1468 Echo Reply [Process]

RFC 2463 *Clause:* 4.1 ¶15 *Type:* MUST *applies to:* Node

Context: The implementation receives an ICMPv6 Echo Reply messages in response of an Echo Request message sent by the implementation.

Requirement: The implementation passes the Echo Reply messages to the [upper-layer] process that originated an Echo Request message.

RFC text: {{Echo Reply messages MUST be passed to the process that originated an Echo Request message}}. It may be passed to processes that did not originate the Echo Request message.

RQ_COR_1469 Echo Reply [Process]

RFC 2463 *Clause:* 4.1 ¶15 *Type:* MAY *applies to:* Node

Context: The implementation receives an ICMPv6 Echo Reply messages in response of an Echo Request message sent by the implementation.

Requirement: The implementation passes the Echo Reply messages to [upper-layer] processes that did not originate the Echo Request message.

RFC text: Echo Reply messages MUST be passed to the process that originated an Echo Request message. {{It may be passed to processes that did not originate the Echo Request message}}.

RQ_COR_1470 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.1 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation implements ICMP protocol.

Requirement: The implementation authenticates the ICMP messages using the IP Authentication Header.

RFC text: {{ICMP protocol packet exchanges can be authenticated using the IP Authentication Header [IPv6-AUTH]}}.

RQ_COR_1471 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.1 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation sends ICMP messages. There exists a security association between the implementation and the destination address.

Requirement: The implementation uses an Authentication Header in the ICMP messages.

RFC text: {{A node SHOULD include an Authentication Header when sending ICMP messages if a security association for use with the IP Authentication Header exists for the destination address}}.

RQ_COR_1472 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives an Authentication Header in an ICMP packet.

Requirement: The implementation verifies [the packet authentication] for correctness.

RFC text: {{Received Authentication Headers in ICMP packets MUST be verified for correctness and packets with incorrect authentication MUST be ignored and discarded}}.

RQ_COR_1473 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives an Authentication Header in an ICMP packet. The packet fails the authentication.

Requirement: The implementation ignores and discards the packet.

RFC text: `{{Received Authentication Headers in ICMP packets MUST be verified for correctness and packets with incorrect authentication MUST be ignored and discarded}}`.

RQ_COR_1474 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.1 ¶3 *Type:* SHOULD *applies to:* Node

Context: The system administrator is configuring the implementation.

Requirement: The implementation has a switch to ignore any ICMP messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload.

RFC text: `{{It SHOULD be possible for the system administrator to configure a node to ignore any ICMP messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload. Such a switch SHOULD default to allowing unauthenticated messages}}`.

RQ_COR_1475 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.1 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation has a switch to ignore any ICMP messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload. This switch is left in its default value. The implementation receives unauthenticated ICMP messages.

Requirement: The implementation processes unauthenticated ICMP messages; i.e. the default value of the switch is to process unauthenticated ICMP messages.

RFC text: `{{It SHOULD be possible for the system administrator to configure a node to ignore any ICMP messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload. Such a switch SHOULD default to allowing unauthenticated messages}}`.

RQ_COR_1476 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.2 ¶2 *Type:* MAY *applies to:* Node

Context: The implementation is subject to actions intended to cause the receiver to believe the ICMP messages come from a different source than the message originator [spoofing].

Requirement: The implementation applies IPv6 Authentication to ICMP messages to protect against these [spoofing] attacks.

RFC text: `{{1. ICMP messages may be subject to actions intended to cause the receiver believe the message came from a different source than the message originator. The protection against this attack can be achieved by applying the IPv6 Authentication mechanism [IPv6-Auth] to the ICMP message}}`.

RQ_COR_1477 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.2 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation is subject to actions intended to cause ICMP messages or their replies to go to a destination different than the message originator's intention.

Requirement: The implementation uses authentication and the ICMP checksum to protect against these types of attacks..

RFC text: {{2. ICMP messages may be subject to actions intended to cause the message or the reply to it go to a destination different than the message originator's intention. The ICMP checksum calculation provides a protection mechanism against changes by a malicious interceptor in the destination and source address of the IP packet carrying that message, provided the ICMP checksum field is protected against change by authentication [IPv6-Auth] or encryption [IPv6-ESP] of the ICMP message.}}.

RQ_COR_1479 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.2 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation subject to ICMP message fields or payload being changed while the packet is in transit.

Requirement: The implementation applies the IPv6 Authentication or Encryption mechanisms to such ICMP messages as protection against this attack.

RFC text: {{3. ICMP messages may be subject to changes in the message fields, or payload. The authentication [IPv6-Auth] or encryption [IPv6-ESP] of the ICMP message is a protection against such actions}}.

RQ_COR_9030 Extension Headers [Process]

RFC 2463 *Clause:* 3.4 ¶9 *Type:* MUST *applies to:* Node

Context: The implementation processes a packet. The implementation finds a problem with a field in the extension headers such that it cannot complete processing the packet.

Requirement: The implementation discards the packet.

RFC text: {{If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it MUST discard the packet and SHOULD send an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem}}.

RQ_COR_9031 Extension Headers [Process]

RFC 2463 *Clause:* 3.4 ¶9 *Type:* SHOULD *applies to:* Node

Context: The implementation processes a packet. The implementation finds a problem with a field in the extension headers such that it cannot complete processing the packet. The implementation discards the packet.

Requirement: The implementation sends an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem.

RFC text: {{If an IPv6 node processing a packet finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it MUST discard the packet and SHOULD send an ICMPv6 Parameter Problem message to the packet's source, indicating the type and location of the problem}}.

RQ_COR_9032 ICMPv6 Messages [Protect From Attacks]

RFC 2463 *Clause:* 5.1 ¶3 *Type:* SHOULD *applies to:* Node

Context: The implementation has a switch to ignore any ICMP messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload. This switch is set to ignore any ICMP messsages that are not authenticated. The implementation receives unauthenticated ICMP messages.

Requirement: The implementation ignores the unauthenticated ICMP messages.

RFC text: `{{It SHOULD be possible for the system administrator to configure a node to ignore any ICMP messages that are not authenticated using either the Authentication Header or Encapsulating Security Payload. Such a switch SHOULD default to allowing unauthenticated messages}}`.

4.10 Requirements extracted from RFC 2464

RQ_COR_8000

RFC 2464 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is not manually configured. The implementation has not received a Router Advertisement message containing an MTU option specifiying an MTU size smaller than the default size of 1500 octets.

Requirement: The implementation uses the default MTU size of 1500 octets.

RFC text: `{{The default MTU size for IPv6 [IPV6] packets on an Ethernet is 1500 octets.}}` This size may be reduced by a Router Advertisement [DISC] containing an MTU option which specifies a smaller MTU, or by manual configuration of each node. If a Router Advertisement received on an Ethernet interface has an MTU option specifying an MTU larger than 1500, or larger than a manually configured value, that MTU option may be logged to system management but must be otherwise ignored.

RQ_COR_8001

RFC 2464 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is not manually configured. It receives a Router Advertisement message containing an MTU option that specifies an MTU smaller than the default size of 1 500 bytes.

Requirement: The implementation sets the default MTU size to the smaller size in the Router Advertisement's MTU option.

RFC text: The default MTU size for IPv6 [IPV6] packets on an Ethernet is 1 500 octets. `{{This size may be reduced by a Router Advertisement [DISC] containing an MTU option which specifies a smaller MTU}}`, or by manual configuration of each node. If a Router Advertisement received on an Ethernet interface has an MTU option specifying an MTU larger than 1 500, or larger than a manually configured value, that MTU option may be logged to system management but must be otherwise ignored.

RQ_COR_8002

RFC 2464 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is manually configured for using IPv6 over Ethernet.

Requirement: The default MTU size in the manual configuration is less than or equal to 1 500 octets.

RFC text: The default MTU size for IPv6 [IPV6] packets on an Ethernet is 1 500 octets. `{{This size may be reduced}}` by a Router Advertisement [DISC] containing an MTU option which specifies a smaller MTU, or `{{by manual configuration of each node.}}` If a Router Advertisement received on an Ethernet interface has an MTU option specifying an MTU larger than 1 500, or larger than a manually configured value, that MTU option may be logged to system management but must be otherwise ignored.

RQ_COR_8003

RFC 2464 *Clause:* 2 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation is using the default MTU size of 1 500 bytes. It receives a Router Advertisement message containing an MTU option specifying an MTU larger than than the default size of 1500 bytes.

Requirement: The implementation logs the MTU option to system management.

RFC text: The default MTU size for IPv6 [IPV6] packets on an Ethernet is 1 500 octets. This size may be reduced by a Router Advertisement [DISC] containing an MTU option which specifies a smaller MTU, or by manual configuration of each node. `{{If a Router Advertisement received on an Ethernet interface has an MTU option specifying an MTU larger than 1 500}}`, or larger than a manually configured value, `{{that MTU option may be logged to system management}}` but must be otherwise ignored.

RQ_COR_8004

RFC 2464 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is using the default MTU size of 1 500 bytes. It receives a Router Advertisement message containing an MTU option specifying an MTU larger than than the default size of 1500 bytes.

Requirement: The implementation continues using the default MTU size of 1 500 bytes.

RFC text: The default MTU size for IPv6 [IPV6] packets on an Ethernet is 1 500 octets. This size may be reduced by a Router Advertisement [DISC] containing an MTU option which specifies a smaller MTU, or by manual configuration of each node. `{{If a Router Advertisement received on an Ethernet interface has an MTU option specifying an MTU larger than 1 500}}`, or larger than a manually configured value, that MTU option may be logged to system management but `{{must be otherwise ignored}}`.

RQ_COR_8005

RFC 2464 *Clause:* 2 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation is using a manually configured default MTU size. The implementation receives a Router Advertisement message containing an MTU option specifying an MTU larger than the manually configured default size.

Requirement: The implementation logs the Router Advertisement's MTU option to system management.

RFC text: The default MTU size for IPv6 [IPV6] packets on an Ethernet is 1 500 octets. This size may be reduced by a Router Advertisement [DISC] containing an MTU option which specifies a smaller MTU, or by manual configuration of each node. `{{If a Router Advertisement received on an Ethernet interface has an MTU option specifying an MTU}}` larger than 1500, or `{{larger than a manually configured value, that MTU option may be logged to system management}}` but must be otherwise ignored.

RQ_COR_8006

RFC 2464 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is using a manually configured default MTU size. The implementation receives a Router Advertisement message containing an MTU option specifying an MTU larger than the manually configured MTU.

Requirement: The implementation continues using the manually configured MTU size as the default.

RFC text: The default MTU size for IPv6 [IPV6] packets on an Ethernet is 1 500 octets. This size may be reduced by a Router Advertisement [DISC] containing an MTU option which specifies a smaller MTU, or by manual configuration of each node. `{{If a Router Advertisement received on an Ethernet interface has an MTU option specifying an MTU}}` larger than 1500, or `{{larger than a manually configured value, that MTU option }}`may be logged to system management but `{{must be otherwise ignored.}}`

RQ_COR_8007 IPv6 in Ethernet Frame

RFC 2464 *Clause:* 3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is transmitting IPv6 Packets over Ethernet.

Requirement: The implementation transmits the IPv6 packets in standard Ethernet frames with the Ethernet header containing the Destination and Source Ethernet addresses. The Ethernet type code is set to the value 86DD hexadecimal.

RFC text: `{{IPv6 packets are transmitted in standard Ethernet frames. The Ethernet header contains the Destination and Source Ethernet addresses and the Ethernet type code, which must contain the value 86DD hexadecimal.}}` The data field contains the IPv6 header followed immediately by the payload, and possibly padding octets to meet the minimum frame size for the Ethernet link.

RQ_COR_8008 IPv6 in Ethernet Frame

RFC 2464 *Clause:* 3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is transmitting IPv6 Packets over Ethernet.

Requirement: The Ethernet data field contains the IPv6 header followed immediately by the IPv6 payload. The Ethernet data field also contains padding octets to meet the Ethernet link's minimum frame size if required.

RFC text: IPv6 packets are transmitted in standard Ethernet frames. The Ethernet header contains the Destination and Source Ethernet addresses and the Ethernet type code, which must contain the value 86DD hexadecimal. `{{The data field contains the IPv6 header followed immediately by the payload, and possibly padding octets to meet the minimum frame size for the Ethernet link.}}.`

RQ_COR_8009 address: [Autoconfigure]

RFC 2464 *Clause:* 4 ¶2-3 *Type:* MUST *applies to:* Node

Context: The IPv6 implementation is configured for Stateless Autoconfiguration to transmit IPv6 packets over Ethernet. The Ethernet implementation has a built-in 48-bit IEEE 802 address.

Requirement: The implementation forms the EUI-64 IPv6 interface identifier in the following manner. (1) The first three octets of the 48-bit IEEE 802 address become the first three octets of the EUI-64 interface identifier except for a change of one bit. The next-to-lowest order bit of the first octet, the U/L bit, of the EUI-64 interface identifier is complemented. (2) The fourth and fifth octets of the EUI identifiers are set to FFFE hexadecimal. (3) The last three octets of the 48-bit IEEE 802 address become the last three octets of the EUI-64 interface identifier.

RFC text: `{{The OUI of the Ethernet address (the first three octets) becomes the company_id of the EUI-64 (the first three octets). The fourth and fifth octets of the EUI are set to the fixed value FFFE hexadecimal. The last three octets of the Ethernet address become the last three octets of the EUI-64. The Interface Identifier is then formed from the EUI-64 by complementing the "Universal/Local" (U/L) bit, which is the next-to- lowest order bit of the first octet of the EUI-64.}}`

RQ_COR_8010 address: [Autoconfigure]

RFC 2464 *Clause:* 4 ¶5 *Type:* SHOULD *applies to:* Node

Context: The IPv6 implementation over Ethernet is configuring its IPv6 EUI-64 interface identifier for Stateless Autoconfiguration.

Requirement: Do not use a manually or software-set MAC address different from the Ethernet implementation's built-in 48-bit IEEE 802 MAC address to form an implementation's IPv6 EUI-64 interface identifier.

RFC text: `{{A different MAC address set manually or by software should not be used to derive the Interface Identifier.}}` If such a MAC address must be used, its global uniqueness property should be reflected in the value of the U/L bit.

RQ_COR_8011 address: [Autoconfigure]

RFC 2464 *Clause:* 4 ¶5 *Type:* SHOULD *applies to:* Node

Context: The IPv6 implementation over Ethernet is configuring its IPv6 EUI-64 interface identifier for Stateless Autoconfiguration. It uses a manually or software-set MAC address different from the Ethernet implementation's built-in 48-bit IEEE 802 MAC address to form its IPv6 EUI-64 interface identifier.

Requirement: The implementation reflects the local uniqueness of the IPv6 interface identifier in the U/L bit. The U/L bit is the next-to-lowest order bit of the first octet of the EUI-64 interface identifier.

RFC text: {{Complementing this bit will generally change a 0 value to a 1, since an interface's built-in address is expected to be from a universally administered address space and hence have a globally unique value... A different MAC address set manually or by software should not be used to derive the Interface Identifier. If such a MAC address must be used, its global uniqueness property should be reflected in the value of the U/L bit.}}

RQ_COR_8012 address: [Autoconfigure]

RFC 2464 *Clause:* 4 ¶6 *Type:* MUST *applies to:* Node

Context: The IPv6 implementation is over Ethernet and and uses Stateless Autoconfiguration.

Requirement: The IPv6 address prefix for stateless autoconfiguration over Ethernet has a length of 64 bits.

RFC text: {{An IPv6 address prefix used for stateless autoconfiguration [ACONF] of an Ethernet interface must have a length of 64 bits.}}

RQ_COR_8013 address: Link-local [Form]

RFC 2464 *Clause:* 5 *Type:* MUST *applies to:* Node

Context: The implementation needs to form a link local address.

Requirement: The implementation forms the link local address by appending the 64-bit IPv6 Interface Identifier of the device concerned to the prefix FE80 0000 0000 0000. If '++' is the concatenation operation, the link local address is (FE80 0000 0000 000 ++ IPv6 Interface Identifier).

RFC text: {{The IPv6 link-local address [AARCH] for an Ethernet interface is formed by appending the Interface Identifier, as defined above (4), to the prefix FE80::/64.}}

RQ_COR_8014 Extension Header Options [Generate]

RFC 2464 *Clause:* 6 *Type:* MUST *applies to:* Node

Context: The implementation is using IPv6 over Ethernet and the Source Link-layer Address option in unicast addressing for an interface.

Requirement: The implementation sets the first octet of the Source Link-layer Address option to 0x01, the second octet to 0x01, and the last six octets to the 48 bit Ethernet IEEE 802 address (in canonical bit order) to which the interface currently responds.

RFC text: Option fields:
{{Type 1 for Source Link-layer address.}}
            2 for Target Link-layer address.
{{Length 1 (in units of 8 octets).}}
{{Ethernet Address The 48 bit Ethernet IEEE 802 address, in canonical bit order. This is the address the interface currently responds to, and may be different from the built-in address used to derive the Interface Identifier.}}

RQ_COR_8015 Extension Header Options [Generate]

RFC 2464 *Clause:* 6 *Type:* MUST *applies to:* Node

Context: The implementation is using IPv6 over Ethernet and the Target Link-layer Address option in unicast addressing for an interface.

Requirement: The implementation sets the first octet of the Source Link-layer Address option to 0x02, the second octet to 0x01, and the last six octets to the 48 bit Ethernet IEEE 802 address (in canonical bit order) to which the interface currently responds.

RFC text: Option fields:
Type 1 for Source Link-layer address.
`{{2 for Target Link-layer address.}}`
`{{Length 1 (in units of 8 octets).}}`
`{{Ethernet Address The 48 bit Ethernet IEEE 802 address, in canonical bit order. This is the address the interface currently responds to, and may be different from the built-in address used to derive the Interface Identifier.}}`

RQ_COR_8016 IPv6 in Ethernet Frame

RFC 2464 *Clause:* 7 *Type:* MUST *applies to:* Node

Context: The IPv6 over Ethernet implementation is using a 16 octet IPv6 multicast address.

Requirement: The implementation converts the IPv6 multicast address to the 6 octet Ethernet multicast address by appending the last 4 octets of the IPv6 multicast address to two octets having the value 0x3333; i.e. (0x3333 ++ DST[13 - 16]) where '++' is the concatenation operator and DST[n] is the nth byte of the IPv6 multicast address.

RFC text: `{{An IPv6 packet with a multicast destination address DST, consisting of the sixteen octets DST[1] through DST[16], is transmitted to the Ethernet multicast address whose first two octets are the value 3333 hexadecimal and whose last four octets are the last four octets of DST.}}`

4.11 Requirements extracted from RFC 2675

RQ_COR_8800 Jumbograms

RFC 2675 *Clause:* Appendix D ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6 Jumbograms.

Requirement: The implementation is attached to a link whose MTU is greater than 65,575 octets.

RFC text: `{{Jumbograms are relevant only to IPv6 nodes that may be attached to links with a link MTU greater than 65,575 octets}}`, and need not be implemented or understood by IPv6 nodes that do not support attachment to links with such large MTUs.

RQ_COR_8801 Jumbograms

RFC 2675 *Clause:* 1 ¶2 *Type:* MAY *applies to:* Node

Context: The implementation does not support attachement to links with MTU greater than 65,575 octets.

Requirement: The implementation does not implement the Jumbo Payload option.

RFC text: The Jumbo Payload option is relevant only for IPv6 nodes that may be attached to links with a link MTU greater than 65,575 octets (that is, 65,535 + 40, where 40 octets is the size of the IPv6 header).
`{{The Jumbo Payload option need not be implemented or understood by IPv6 nodes that do not support attachment to links with MTU greater than 65,575.}}`

RQ_COR_8802 Jumbograms

RFC 2675 *Clause:* 1 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation is on a link with a configurable MTU. Nodes on this link do not support the Jumbo Payload option.

Requirement: The implementation does not transmit in its Router Advertisements an MTU value greater than 65,575 octets.

RFC text: `{{On links with configurable MTUs, the MTU must not be configured to a value greater than 65,575 octets if there are nodes attached to that link that do not support the Jumbo Payload option}}` and it can not be guaranteed that the Jumbo Payload option will not be sent to those nodes.

RQ_COR_8803 Jumbograms

RFC 2675 *Clause:* 1 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation supports the Jumbo Payload option and is using TCP and UDP.

Requirement: The implementation incorporates the TCP and UDP enhancements contained in RFC 2675.

RFC text: The UDP header [UDP] has a 16-bit Length field which prevents it from making use of jumbograms, and though the TCP header [TCP] does not have a Length field, both the TCP MSS option and the TCP Urgent field are constrained to 16 bits. This document specifies some simple enhancements to TCP and UDP to enable them to make use of jumbograms. `{{An implementation of TCP or UDP on an IPv6 node that supports the Jumbo Payload option must include the enhancements specified here.}}`

RQ_COR_8804 Jumbograms [Generate]

RFC 2675 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation supports the Jumbo Payload Option and is on a link whose MTU is greater than 65,575 octets.

Requirement: The implementation places the Jumbo Payload option in an IPv6 Hop-by-Hop Options header immediately following the IPv6 header. The Option Type field is set to 0xC2 (hex, 8 bits) and the Opt Dat Len field is set to 4 (8 bits). The option's Length field is set to the length of the IPv6 packet in octets excluding the IPv6 header but including the Hop-by-Hop Options header and any other extension headers present. The Length value must be greater than 65,535.

RFC text:

```
{{The Jumbo Payload option is carried in an IPv6 Hop-by-Hop Options
header, immediately following the IPv6 header. This option has an
alignment requirement of 4n + 2. (See [IPv6, section 4.2] for
discussion of option alignment.) The option has the following format:
                                   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   |  Option Type  |  Opt Data Len |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Jumbo Payload Length                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Option Type          8-bit value C2 (hexadecimal).
Opt Data Len         8-bit value 4.
Jumbo Payload Length  32-bit unsigned integer. Length of the IPv6
packet in octets, excluding the IPv6 header but including the
Hop-by-Hop Options header and any other extension headers present.
Must be greater than 65,535.}}
```

RQ_COR_8805 Jumbograms [Generate]

RFC 2675 *Clause:* 3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is generating an IPv6 packet that contains a Jumbo Payload option.

Requirement: The implementation sets to zero the Payload Length field in the IPv6 header.

RFC text: `{{The Payload Length field in the IPv6 header must be set to zero in every packet that carries the Jumbo Payload option.}}`

RQ_COR_8806 Jumbograms [Process]

RFC 2675 *Clause:* 3 ¶2 *Type:* MUST *applies to:* Node

Context: The node implements the Jumbo Payload option. The node receives a packet whose IPv6 header carries a Payload Length of zero and a Next Header value of zero. The link-layer framing indicates octets beyond the IPv6 header.

Requirement: The implementation processes the Hop-by-Hop Options header to determine the payload's actual length.

RFC text: `{{If a node that understands the Jumbo Payload option receives a packet whose IPv6 header carries a Payload Length of zero and a Next Header value of zero (meaning that a Hop-by-Hop Options header follows), and whose link-layer framing indicates the presence of octets beyond the IPv6 header, the node must proceed to process the Hop-by-Hop Options header in order to determine the actual length of the payload from the Jumbo Payload option.}}`

RQ_COR_8807 Jumbograms [Generate]

RFC 2675 *Clause:* 3 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation supports the Jumbo Payload Option and is on a link whose MTU is greater than 65,575 octets. The implementation is generating a packet that carries a Fragment header.

Requirement: The implementation does not include the Jumbo Payload option in the packet.

RFC text: `{{The Jumbo Payload option must not be used in a packet that carries a Fragment header.}}`

RQ_COR_8808 Jumbograms [Generate]

RFC 2675 *Clause:* 3 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation is generating a packet that contains a Jumbo Payload Option and is on a link whose MTU is greater than 65,575 octets.

Requirement: The implementation uses the Jumbo Payload Length field for computing the checksum pseudo-header described in RFC 2460, 8.1.

RFC text: Higher-layer protocols that use the IPv6 Payload Length field to compute the value of the Upper-Layer Packet Length field in the checksum pseudo-header described in [IPv6, section 8.1] must instead `{{use the Jumbo Payload Length field for that computation, for packets that carry the Jumbo Payload option.}}`

RQ_COR_8809 Jumbograms [Process]

RFC 2675 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses the Jumbo Payload option and is on a link whose MTU is greater than 65,575 octets. It receives an IPv6 packet whose IPv6 Payload Length is set to 0 and the IPv6 Next Header is set to the Hop-by-Hop Option. The Jumbo Payload option is not in the packet. The packet's Destination Address is not a multicast address.

Requirement: The implementation transmits an ICMP Parameter Problem message with the Code set to zero and the Pointer pointing to the high-order octet of the IPv6 Payload Length.

RFC text:
```
{{error: IPv6 Payload Length = 0 and IPv6 Next Header = Hop-by-Hop
Options and Jumbo Payload option not present
Code: 0
Pointer: high-order octet of the IPv6 Payload Length}}
```

RQ_COR_8810 Jumbograms [Process]

RFC 2675 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses the Jumbo Payload option and is on a link whose MTU is greater than 65,575 octets. It receives an IPv6 packet whose IPv6 Payload Length is set to a value other than 0. The Jumbo Payload option is in the packet. The packet's Destination Address is not a multicast address.

Requirement: The implementation transmits an ICMP Parameter Problem message with the Code set to zero and the Pointer pointing to the Option Type field of the Jumbo Payload option.

RFC text:
```
{{error: IPv6 Payload Length != 0 and Jumbo Payload option present
Code: 0
Pointer: Option Type field of the Jumbo Payload option}}
```

RQ_COR_8811 Jumbograms [Process]

RFC 2675 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses the Jumbo Payload option and is on a link whose MTU is greater than 65,575 octets. It receives an IPv6 packet with Jumbo Payload option whose Jumbo Payload length is less than 65,536. The packet's Destination Address is not a multicast address.

Requirement: The implementation transmits an ICMP Parameter Problem message with the Code set to zero and the Pointer pointing to the high-order octet of the Jumbo Payload Length.

RFC text:
```
{{error: Jumbo Payload option present and Jumbo Payload Length <
65,536
Code: 0
Pointer: high-order octet of the Jumbo Payload Length}}
```

RQ_COR_8812 Jumbograms [Process]

RFC 2675 *Clause:* 3 ¶5 *Type:* MUST *applies to:* Node

Context: The implementation uses the Jumbo Payload option and is on a link whose MTU is greater than 65,575 octets. It receives an IPv6 packet with both a Jumbo Payload option and a Fragement header present. The packet's Destination Address is not a multicast address.

Requirement: The implementation transmits an ICMP Parameter Problem message with the Code set to zero and the Pointer pointing to the high-order octet of the Fragment header.

RFC text:
```
{{error: Jumbo Payload option present and Fragment header present
Code: 0
Pointer: high-order octet of the Fragment header.}}
```

RQ_COR_8813 Hop by Hop Header [Process]

RFC 2675 *Clause:* 3 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation does not recognize the Jumbo Payload option. It receives an IPv6 packet whose IPv6 Payload Length = 0 and the IPv6 Next Header = Hop-by-Hop Option.

Requirement: The implementation transmits an ICMP Parameter Problem message with the Code set to zero and the Pointer pointing to the high-order octet of the IPv6 Payload Length.

RFC text:
```
{{A node that does not understand the Jumbo Payload option is
expected to respond to erroneously-received jumbograms as follows,
according to the IPv6 specification:
error: IPv6 Payload Length = 0 and IPv6 Next Header = Hop-by-Hop
Options
Code: 0
Pointer: high-order octet of the IPv6 Payload Length}}
```

RQ_COR_8814 Process IPv6 Packet

RFC 2675 *Clause:* 3 ¶6 *Type:* MUST *applies to:* Node

Context: The implementation does not recognize the Jumbo Payload option. It receives an IPv6 packet whose IPv6 Payload Length is other than 0 and the Jumbo Payload option is present.

Requirement: The implementation transmits an ICMP Parameter Problem message with the Code set to 2 and the Pointer pointing to the Option Type field of the Jumbo Payload option.

RFC text:
```
{{error: IPv6 Payload Length != 0 and Jumbo Payload option present
Code: 2
Pointer: Option Type field of the Jumbo Payload option}}
```

RQ_COR_8815 Jumbograms - UDP

RFC 2675 *Clause:* 4 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is generating a Jumbogram carrying a UDP header plus UDP greater than 65,535 octets.

Requirement: The implementation sets the Length field in the UDP header to zero and sets the Jumbo Payload Length field of the Jumbo Payload Option to the length of the UDP header plus the UDP data plus the length of all IPv6 extension headers present between the IPv6 header and the UDP header. The implementation calculates the UDP checksum using the actual length of the UDP header plus UDP data in the checksum pseudo-header.

RFC text:
```
{{The specific requirements for sending a UDP jumbogram are as
follows:
When sending a UDP packet, if and only if the length of the UDP
header plus UDP data is greater than 65,535, set the Length field in
the UDP header to zero.
The IPv6 packet carrying such a large UDP packet will necessarily
include a Jumbo Payload option in a Hop-by-Hop Options header; set
the Jumbo Payload Length field of that option to be the actual length
of the UDP header plus data, plus the length of all IPv6 extension
headers present between the IPv6 header and the UDP header.
 For generating the UDP checksum, use the actual length of the UDP
header plus data, NOT zero, in the checksum pseudo-header [IPv6,
section 8.1].}}
```

RQ_COR_8816 Jumbograms - UDP

RFC 2675 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a UDP packet whose Length field in the UDP header is zero. A Jumbo Payload option is present in the packet.

Requirement: The implementation determines the length of the UDP header plus data by the length of all extension headers present between the IPv6 header and the UDP header from the IPv6 Jumbo Payload Length field. The implementation verifies the received UDP checksum suing the calculated length of the UDP header plus UDP data in the checksum pseudo-header.

RFC text:
```
{{When receiving a UDP packet, if and only if the Length field in the
UDP header is zero, calculate the actual length of the UDP header
plus data from the IPv6 Jumbo Payload Length field minus the length
of all extension headers present between the IPv6 header and the UDP
header.
For verifying the received UDP checksum, use the calculated length of
the UDP header plus data, NOT zero, in the checksum pseudo-header.}}
```

RQ_COR_8817 Jumbograms - UDP

RFC 2675 *Clause:* 4 ¶3 *Type:* MUST *applies to:* Node

Context: The implementation receives a UDP packet whose Length field in the UDP header is zero. No Jumbo Payload option is present in the packet.

Requirement: The implementation determines the length of the UDP header plus data by the length of all extension headers present between the IPv6 header and the UDP header from the IPv6 header Payload Length field. The implementation verifies the received UDP checksum suing the calculated length of the UDP header plus UDP data in the checksum pseudo-header.

RFC text:
```
{{In the unexpected case that the UDP Length field is zero but no
Jumbo Payload option is present (i.e., the IPv6 packet is not a
jumbogram), use the Payload Length field in the IPv6 header, in place
of the Jumbo Payload Length field, in the above calculation.
For verifying the received UDP checksum, use the calculated length of
the UDP header plus data, NOT zero, in the checksum pseudo-header.}}
```

RQ_COR_8818 Jumbograms - TCP

RFC 2675 *Clause:* 5.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation is generating a TCP Jumbogram. The MTU of the directly attached interface minus 60 is greater than or equal to 65,535 bytes.

Requirement: The implementations sets the MSS value of the TCP header to 65,535.

RFC text:
```
{{When determining what MSS value to send, if the MTU of the directly
attached interface minus 60 [IPv6, section 8.3] is greater than or
equal to 65,535, then set the MSS value to 65,535.}}
```

RQ_COR_8819 Jumbograms - TCP

RFC 2675 *Clause:* 5.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation receives a TCP Jumbogram with the MSS value in the TCP header = 65,535.

Requirement: The implementation treats the received MSS value of 65,535 as infinity. The implementation calculates the actual MSS by subtracting 60 from the value learned by performing Path MTU Discovery [MTU-DISC] over the path to the TCP peer.

RFC text:
```
{{When an MSS value of 65,535 is received, it is to be treated as
infinity. The actual MSS is determined by subtracting 60 from the
value learned by performing Path MTU Discovery [MTU-DISC] over the
path to the TCP peer.}}
```

RQ_COR_8820 Jumbograms - TCP

RFC 2675 *Clause:* 5.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is generating a TCP Jumbogram. The MTU of the directly attached interface minus 60 is greater than or equal to 65,535. The implementation is generating a TCP that contains an Urgent Pointer (i.e. the URG bit is set to one). The offset from the Sequence Number to the Urgent Pointer is less than 65,535.

Requirement: The implenentation sets the Urgent Pointer to the offset value and continues normal TCP processing.

RFC text: `{{When a TCP packet is to be sent with an Urgent Pointer (i.e., the URG bit set), first calculate the offset from the Sequence Number to the Urgent Pointer. If the offset is less than 65,535, fill in the Urgent field and continue with the normal TCP processing.}}` If the offset is greater than 65,535, and the offset is greater than or equal to the length of the TCP data, fill in the Urgent Pointer with 65,535 and continue with the normal TCP processing. Otherwise, the TCP packet must be split into two pieces. The first piece contains data up to, but not including the data pointed to by the Urgent Pointer, and the Urgent field is set to 65,535 to indicate that the Urgent Pointer is beyond the end of this packet. The second piece can then be sent with the Urgent field set normally.

RQ_COR_8821 Jumbograms - TCP

RFC 2675 *Clause:* 5.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is generating a TCP Jumbogram. The MTU of the directly attached interface minus 60 is greater than or equal to 65,535. The implementation is generating a TCP that contains an Urgent Pointer (i.e. the URG bit is set to one). The offset from the Sequence Number to the Urgent Pointer is greater than 65,535 and greater than or equal to the length of the TCP data.

Requirement: The implenentation sets the Urgent Pointer to the 65,535 and continues normal TCP processing.

RFC text: `{{When a TCP packet is to be sent with an Urgent Pointer (i.e., the URG bit set), first calculate the offset from the Sequence Number to the Urgent Pointer.}}` If the offset is less than 65,535, fill in the Urgent field and continue with the normal TCP processing. `{{If the offset is greater than 65,535, and the offset is greater than or equal to the length of the TCP data, fill in the Urgent Pointer with 65,535 and continue with the normal TCP processing.}}` Otherwise, the TCP packet must be split into two pieces. The first piece contains data up to, but not including the data pointed to by the Urgent Pointer, and the Urgent field is set to 65,535 to indicate that the Urgent Pointer is beyond the end of this packet. The second piece can then be sent with the Urgent field set normally.

RQ_COR_8822 Jumbograms - TCP

RFC 2675 *Clause:* 5.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is generating a TCP Jumbogram. The MTU of the directly attached interface minus 60 is greater than or equal to 65,535. The implementation is generating a TCP that contains an Urgent Pointer (i.e. the URG bit is set to one). The offset from the Sequence Number to the Urgent Pointer is greater than 65,535 and less than the length of the TCP data.

Requirement: The implementation splits the TCP packet into two pieces. The first piece contains data up to, but not including, the data pointed to by the Urgent Pointer. The implementation sets the first piece's Urgent Pointer field is set to 65,535. The implementations sets the second piece's Urgent Pointer field as usual.

RFC text: `{{When a TCP packet is to be sent with an Urgent Pointer (i.e., the URG bit set), first calculate the offset from the Sequence Number to the Urgent Pointer.}}` If the offset is less than 65,535, fill in the Urgent field and continue with the normal TCP processing. If the offset is greater than 65,535, and the offset is greater than or equal to the length of the TCP data, fill in the Urgent Pointer with 65,535 and continue with the normal TCP processing. `{{Otherwise, the TCP packet must be split into two pieces. The first piece contains data up to, but not including the data pointed to by the Urgent Pointer, and the Urgent field is set to 65,535 to indicate that the Urgent Pointer is beyond the end of this packet. The second piece can then be sent with the Urgent field set normally.}}`

RQ_COR_8823

RFC 2675 *Clause:* 5.2 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation receives a TCP Jumbogram with the URG bit set and an Urgent [Pointer] field fo 65,535.

Requirement: The implementation calculates the Urgent Pointer using an offset equal to the length of the TCP data.

RFC text: `{{For TCP input processing, when a TCP packet is received with the URG bit set and an Urgent field of 65,535, the Urgent Pointer is calculated using an offset equal to the length of the TCP data, rather than the offset in the Urgent field.}}`

4.12 Requirements extracted from RFC 3513

RQ_COR_1600 Interface Identifiers

RFC 3513 *Clause:* 2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6.

Requirement: The implementation uses 128-bit identifiers as IPv6 addresses for interfaces and sets of interfaces.

RFC text: `{{IPv6 addresses are 128-bit identifiers for interfaces and sets of interfaces (where "interface" is as defined in section 2 of [IPV6])}}`.

RQ_COR_1601 Address Architecture

RFC 3513 *Clause:* 2 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6.

Requirement: The implementation uses three types of IPv6 addresses: Unicast, Anycast and Multicast. The implementation does not use broadcast addresses, their function being superseded by multicast addresses.

RFC text: `{{There are three types of addresses: Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance). Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address}}`. `{{There are no broadcast addresses in IPv6, their function being superseded by multicast addresses}}`.

RQ_COR_1602 address: Unicast

RFC 3513 *Clause:* 2 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses a Unicast IPv6 address.

Requirement: The implementation uses that address as an identifier for a single interface.

RFC text: There are three types of addresses: `{{Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address}}`. Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance). Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

RQ_COR_1603 address: Unicast

RFC 3513 *Clause:* 2 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses a Unicast IPv6 address as an identifier for a single interface.

Requirement: The implementation receives the packets sent to that Unicast address.

RFC text: There are three types of addresses: `{{Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address}}`. Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance). Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

RQ_COR_1604 address: Anycast

RFC 3513 *Clause:* 2 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation uses an Anycast IPv6 address.

Requirement: The implementation shares the use of that address as an identifier for a set of interfaces (typically belonging to different nodes).

RFC text: There are three types of addresses: Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. `{{Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance)}}`. Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

RQ_COR_1605 address: Anycast

RFC 3513 *Clause:* 2 ¶1-5, 2.6 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation shares the use of an Anycast IPv6 address as an identifier for a set of interfaces (typically belonging to different nodes). The implementation is the "nearest" interface to the given source generating packets for this Anycast address.

Requirement: The implementation receives the packets sent to that Anycast address.

RFC text: There are three types of addresses: Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. `{{Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance)}}`. Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

RQ_COR_1606 address: Anycast

RFC 3513 *Clause:* 2 ¶1-5, 2.6 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation shares the use of an Anycast IPv6 address as an identifier for a set of interfaces (typically belonging to different nodes). The implementation is not the "nearest" interface to a given source generating packets for this Anycast address.

Requirement: The implementation does not receive the packets sent to that Anycast address.

RFC text: There are three types of addresses: Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. `{{Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance)}}`. Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

RQ_COR_1607 address: Multicast

RFC 3513 *Clause:* 2 ¶1-5, 2.7 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses a Multicast IPv6 address.

Requirement: The implementation shares the use of that address as an identifier for a set of interfaces (typically belonging to different nodes).

RFC text: There are three types of addresses: Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance). `{{Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address}}`.

RQ_COR_1608 address: Multicast

RFC 3513 *Clause:* 2 ¶1-5 *Type:* MUST *applies to:* Node

Context: The implementation shares the use of an Multicast IPv6 address as an identifier for a set of interfaces (typically belonging to different nodes).

Requirement: The implementation receives the packets sent to that Multicast address.

RFC text: There are three types of addresses: Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance). `{{Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address}}`.

RQ_COR_1609 Address Architecture

RFC 3513 *Clause:* 2 ¶6-7 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6 address.

Requirement: The implementation processes all zeros and all ones values in any [ID or prefix] field of the IPv6 address, unless specifically excluded.

RFC text: In this document, fields in addresses are given a specific name, for example "subnet". When this name is used with the term "ID" for identifier after the name (e.g., "subnet ID"), it refers to the contents of the named field. When it is used with the term "prefix" (e.g., "subnet prefix") it refers to all of the address from the left up to and including this field. `{{In IPv6, all zeros and all ones are legal values for any field, unless specifically excluded}}`. Specifically, prefixes may contain, or end with, zero-valued fields.

RQ_COR_1610 Address Architecture

RFC 3513 *Clause:* 2 ¶6-7 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6 address.

Requirement: The implementation's address prefixes contain or end with zero-valued fields.

RFC text: In this document, fields in addresses are given a specific name, for example "subnet". When this name is used with the term "ID" for identifier after the name (e.g., "subnet ID"), it refers to the contents of the named field. When it is used with the term "prefix" (e.g., "subnet prefix") it refers to all of the address from the left up to and including this field. In IPv6, all zeros and all ones are legal values for any field, unless specifically excluded. `{{Specifically, prefixes may contain, or end with, zero-valued fields}}`.

RQ_COR_1611 Address Architecture

RFC 3513 *Clause:* 2.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6 address.

Requirement: IPv6 addresses of all types are assigned to interfaces, not nodes.

RFC text: `{{IPv6 addresses of all types are assigned to interfaces, not nodes}}`.

RQ_COR_1612 address: Unicast

RFC 3513 *Clause:* 2.1 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation has several interfaces Unicast addresses for those interfaces.

Requirement: The implementation uses any of those addresses as an identifier for the implementation.

RFC text: An IPv6 unicast address refers to a single interface. `{{Since each interface belongs to a single node, any of that node's interfaces' unicast addresses may be used as an identifier for the node}}`.

RQ_COR_1613 address: Link-local

RFC 3513 *Clause:* 2.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation has one or more interfaces.

Requirement: The implementation has at least one link-local unicast address per interface.

RFC text: `{{All interfaces are required to have at least one link-local unicast address (see section 2.8 for additional required addresses)}}`.

RQ_COR_1614 Address Architecture

RFC 3513 *Clause:* 2.1 ¶2 *Type:* MAY *applies to:* Node

Context: The implementation has one or more interfaces.

Requirement: The implementation has multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope for each interface.

RFC text: `{{A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope}}`.

RQ_COR_1615 address: Unicast

RFC 3513 *Clause:* 2.1 ¶2 *Type:* MAY *applies to:* Node

Context: The implementation has one interface. This interface receives and transmits only to neighbors.

Requirement: The implementation uses unicast addresses having only link scope.

RFC text: All interfaces are required to have at least one link-local unicast address (see section 2.8 for additional required addresses). A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope. `{{Unicast addresses with scope greater than link-scope are not needed for interfaces that are not used as the origin or destination of any IPv6 packets to or from non-neighbors. This is sometimes convenient for point-to-point interfaces}}`. There is one exception to this addressing model: A unicast address or a set of unicast addresses may be assigned to multiple physical interfaces if the implementation treats the multiple physical interfaces as one interface when presenting it to the internet layer. This is useful for load-sharing over multiple physical interfaces.

RQ_COR_1616

RFC 3513 *Clause:* 2.1 ¶2 *Type:* MAY *applies to:* Node

Context: The implementation has one interfaces that is not used as destination of any IPv6 packets from non-neighbors.

Requirement: The implementation does not needed in that interface a unicast addresses with scope greater than link-scope.

RFC text: All interfaces are required to have at least one link-local unicast address (see section 2.8 for additional required addresses). A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope. `{{Unicast addresses with scope greater than link-scope are not needed for interfaces that are not used as the origin or destination of any IPv6 packets to or from non-neighbors. This is sometimes convenient for point-to-point interfaces}}`. There is one exception to this addressing model: A unicast address or a set of unicast addresses may be assigned to multiple physical interfaces if the implementation treats the multiple physical interfaces as one interface when presenting it to the internet layer. This is useful for load-sharing over multiple physical interfaces.

RQ_COR_1617 address: Unicast

RFC 3513 *Clause:* 2.1 ¶2-3 *Type:* MAY *applies to:* Node

Context: The implementation treats its multiple physical interfaces as one interface when presenting them to the Internet layer. This is useful for load-sharing over multiple physical interfaces.

Requirement: The implementation assigns a unicast address or a set of unicast addresses to its multiple physical interfaces.

RFC text: All interfaces are required to have at least one link-local unicast address (see section 2.8 for additional required addresses). A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope. Unicast addresses with scope greater than link-scope are not needed for interfaces that are not used as the origin or destination of any IPv6 packets to or from non-neighbors. This is sometimes convenient for point-to-point interfaces. `{{There is one exception to this addressing model: A unicast address or a set of unicast addresses may be assigned to multiple physical interfaces if the implementation treats the multiple physical interfaces as one interface when presenting it to the internet layer. This is useful for load-sharing over multiple physical interfaces}}`.

RQ_COR_1618 Address Architecture

RFC 3513 *Clause:* 2.1 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6.

Requirement: The implementation associates a subnet prefix with one link.

RFC text: `{{Currently IPv6 continues the IPv4 model that a subnet prefix is associated with one link. Multiple subnet prefixes may be assigned to the same link}}`.

RQ_COR_1619 Address Architecture

RFC 3513 *Clause:* 2.1 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6.

Requirement: The implementation associates multiple subnet prefixes with one link.

RFC text: `{{Currently IPv6 continues the IPv4 model that a subnet prefix is associated with one link. Multiple subnet prefixes may be assigned to the same link}}`.

RQ_COR_1620 Address Notation

RFC 3513 *Clause:* 2.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6.

Requirement: [For information only.] The implementation uses three conventional forms for representing IPv6 addresses as text strings.

RFC text: `{{There are three conventional forms for representing IPv6 addresses as text strings }}`: 1 The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. [Examples here] Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.). 2 Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. [Examples here] 3 An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). [Examples here].

RQ_COR_1621 Address Notation

RFC 3513 *Clause:* 2.2 ¶2-5 *Type:* MUST *applies to:* Node

Context: The implementation needs to represent an address as x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address.

Requirement: [For information only.] The implementation uses the preferred form for representing IPv6 addresses as text strings.

RFC text: There are three conventional forms for representing IPv6 addresses as text strings: `{{1 The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. [Examples here] Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.)}}`. 2 Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. [Examples here] 3 An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). [Examples here].

RQ_COR_1622 Address Notation

RFC 3513 *Clause:* 2.2 ¶2-5 *Type:* MUST *applies to:* Node

Context: The implementation uses an address as x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address, and some of them are zeros.

Requirement: [For information only.] The implementation does not need to write the leading zeros in an individual field, but there be at least one numeral in every field (except for the case described in 2.)

RFC text: There are three conventional forms for representing IPv6 addresses as text strings: `{{1 The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. [Examples here] Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.)}}`. 2 Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. [Examples here] 3 An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). [Examples here].

RQ_COR_1623 Address Notation

RFC 3513 *Clause:* 2.2 ¶6-11 *Type:* MAY *applies to:* Node

Context: The implementation uses an address as x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address, and it contains long strings of zero bits.

Requirement: [For information only.] The implementation uses the "::" syntax, indicating one or more [consecutive] groups of 16 bits of zeros.

RFC text: There are three conventional forms for representing IPv6 addresses as text strings: 1 The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. [Examples here] Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.). `{{2 Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. [Examples here]}}` 3 An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). [Examples here].

RQ_COR_1624 Address Notation

RFC 3513 *Clause:* 2.2 ¶6-11 *Type:* MAY *applies to:* Node

Context: The implementation uses an address as x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address, and it contains long strings of zero bits. The implementation needs to compress leading or trailing zeros in an address.

Requirement: [For information only.] The implementation uses the "::" syntax, indicating one or more [consecutive] groups of 16 bits of zeros.

RFC text: There are three conventional forms for representing IPv6 addresses as text strings: 1 The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. [Examples here] Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.). `{{2 Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. [Examples here]}}` 3 An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). [Examples here].

RQ_COR_1625 Address Notation

RFC 3513 *Clause:* 2.2 ¶6-11 *Type:* MUST *applies to:* Node

Context: The implementation uses an address as x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address, and it contains long strings of zero bits.

Requirement: [For information only.] The implementation uses the "::" syntax only once in an address.

RFC text: There are three conventional forms for representing IPv6 addresses as text strings: 1 The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. [Examples here] Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.). `{{2 Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. [Examples here]}}` 3 An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). [Examples here].

RQ_COR_1626 Address Notation

RFC 3513 *Clause:* 2.2 ¶12-15 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6 and IPv4.

Requirement: [For information only.] The implementation uses an address as x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation).

RFC text: There are three conventional forms for representing IPv6 addresses as text strings: 1 The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. [Examples here] Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.). 2 Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. [Examples here] {{3 An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). [Examples here]}}.

RQ_COR_1627 Address Notation

RFC 3513 *Clause:* 2.3 ¶1-5 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6.

Requirement: [For information only.] The implementation uses the following notation in order to represent an IPv6 address prefix: ipv6-address/prefix-length.

RFC text: The text representation of IPv6 address prefixes is similar to the way IPv4 addresses prefixes are written in CIDR notation [CIDR]. {{An IPv6 address prefix is represented by the notation: ipv6-address/prefix-length where ipv6-address is an IPv6 address in any of the notations listed in section 2.2. prefix-length is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix}}.

RQ_COR_1628 Address Architecture

RFC 3513 *Clause:* 2.4 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6.

Requirement: The implementation identifies the type of an IPv6 address by the high-order bits of the address.

RFC text: {{The type of an IPv6 address is identified by the high-order bits of the address, as follows:...}}.

RQ_COR_1629 address: Unspecified Address

RFC 3513 *Clause:* 2.4 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses an Unspecified IPv6 address.

Requirement: The implementation identifies such address by this binary prefix: 00...0 (128 bits).

RFC text: {{The type of an IPv6 address is identified by the high-order bits of the address, as follows:... Unspecified 00...0 (128 bits)}}.

RQ_COR_1630 address: Loopback

RFC 3513 *Clause:* 2.4 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses a Loopback IPv6 address.

Requirement: The implementation identifies such address by this binary prefix: 00...1 (128 bits).

RFC text: `{{The type of an IPv6 address is identified by the high-order bits of the address, as follows:... Loopback 00...1 (128 bits)}}`.

RQ_COR_1631

RFC 3513 *Clause:* 2.4 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses a Multicast IPv6 address.

Requirement: The implementation identifies such address by this binary prefix: 11111111 (8 bits)

RFC text: `{{The type of an IPv6 address is identified by the high-order bits of the address, as follows:... Multicast 11111111}}`.

RQ_COR_1632 address: Unicast

RFC 3513 *Clause:* 2.4 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses a Link-local unicast IPv6 address.

Requirement: The implementation identifies such address by this binary prefix: 1111111010 (10 bits)

RFC text: `{{The type of an IPv6 address is identified by the high-order bits of the address, as follows:... Link-local unicast 1111111010}}`.

RQ_COR_1633 address: Unicast

RFC 3513 *Clause:* 2.4 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation is operating.

Requirement: The implementation treats as a global unicast any address that is not Unspecified, Loopback, Multicast, nor Link-local unicast addresses.

RFC text: `{{The type of an IPv6 address is identified by the high-order bits of the address, as follows: ... Global unicast (everything else)}}`.

RQ_COR_1634 address: Anycast

RFC 3513 *Clause:* 2.4 ¶3, 2.6 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses an Anycast IPv6 address.

Requirement: The implementation takes the Anycast addresses from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses.

RFC text: `{{Anycast addresses are taken from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses}}`.

RQ_COR_1635 address: Unicast

RFC 3513 *Clause:* 2.5 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses a Unicast IPv6 address.

Requirement: The implementation's unicast addresses are aggregable with prefixes of arbitrary bit-length similar to IPv4 addresses under Classless Interdomain Routing.

RFC text: `{{IPv6 unicast addresses are aggregable with prefixes of arbitrary bit-length similar to IPv4 addresses under Classless Interdomain Routing}}`.

RQ_COR_1636 address: Unicast

RFC 3513 *Clause:* 2.5 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses a Unicast IPv6 address.

Requirement: The implementation implements several types of unicast addresses in IPv6, in particular global unicast, site-local unicast [deprecated], and link-local unicast. There are also some special-purpose subtypes of global unicast, such as IPv6 addresses with embedded IPv4 addresses or encoded NSAP addresses.

RFC text: `{{There are several types of unicast addresses in IPv6, in particular global unicast, site-local unicast, and link-local unicast. There are also some special-purpose subtypes of global unicast, such as IPv6 addresses with embedded IPv4 addresses or encoded NSAP addresses. Additional address types or subtypes can be defined in the future}}`.

RQ_COR_1637

RFC 3513 *Clause:* 2.5 ¶3 *Type:* MAY *applies to:* Node

Context: The implementation uses a Unicast IPv6 address.

Requirement: The implementation have considerable or little knowledge of the internal structure of the IPv6 address, depending on the role the node plays (for instance, host versus router).

RFC text: `{{IPv6 nodes may have considerable or little knowledge of the internal structure of the IPv6 address, depending on the role the node plays (for instance, host versus router)}}`.

RQ_COR_1638

RFC 3513 *Clause:* 2.5 ¶3-4 *Type:* MAY *applies to:* Node

Context: The implementation uses a Unicast IPv6 address.

Requirement: The implementation considers that unicast addresses (including its own) have no internal structure.

RFC text: `{{At a minimum, a node may consider that unicast addresses (including its own) have no internal structure:...}}` A slightly sophisticated host (but still rather simple) may additionally be aware of subnet prefix(es) for the link(s) it is attached to, where different addresses may have different values for n:....

RQ_COR_1639

RFC 3513 *Clause:* 2.5 ¶5-6 *Type:* MAY *applies to:* Node

Context: The implementation uses a Unicast IPv6 address. The implementation considers that unicast addresses (including its own) have no internal structure.

Requirement: The implementation additionally is aware of subnet prefix(es) for the link(s) it is attached to, where different addresses have different values for n.

RFC text: At a minimum, a node may consider that unicast addresses (including its own) have no internal structure:... `{{A slightly sophisticated host (but still rather simple) may additionally be aware of subnet prefix(es) for the link(s) it is attached to, where different addresses may have different values for n:...}}`.

RQ_COR_1640

RFC 3513 *Clause:* 2.5 ¶7 *Type:* MAY *applies to:* Router

Context: The implementation is a very simple router. The implementation uses a Unicast IPv6 address.

Requirement: The implementation has no knowledge of the internal structure of IPv6 unicast addresses.

RFC text: `{{Though a very simple router may have no knowledge of the internal structure of IPv6 unicast addresses, routers will more generally have knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. The known boundaries will differ from router to router, depending on what positions the router holds in the routing hierarchy}}`.

RQ_COR_1641

RFC 3513 *Clause:* 2.5 ¶7 *Type:* MAY *applies to:* Router

Context: The implementation is a general router. The implementation uses a Unicast IPv6 address.

Requirement: The implementation has knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. The known boundaries will differ from router to router, depending on what positions the router holds in the routing hierarchy.

RFC text: `{{Though a very simple router may have no knowledge of the internal structure of IPv6 unicast addresses, routers will more generally have knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. The known boundaries will differ from router to router, depending on what positions the router holds in the routing hierarchy}}`.

RQ_COR_1642 Interface Identifiers

RFC 3513 *Clause:* 2.5.1 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses Interface identifiers.

Requirement: The implementation uses the Interface identifiers in IPv6 unicast addresses in order to identify interfaces on a link. The Interface identifiers are unique within a subnet prefix.

RFC text: `{{Interface identifiers in IPv6 unicast addresses are used to identify interfaces on a link. They are required to be unique within a subnet prefix}}`.

RQ_COR_1643 Interface Identifiers

RFC 3513 *Clause:* 2.5.1 ¶1 *Type:* SHOULD *applies to:* Node

Context: The implementation use an Interface identifier.

Requirement: The implementation's interface identifier is not assigned to different nodes on a link.

RFC text: `{{It is recommended that the same interface identifier not be assigned to different nodes on a link. They may also be unique over a broader scope}}`.

RQ_COR_1644 Interface Identifiers

RFC 3513 *Clause:* 2.5.1 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation use an Interface identifier that is not assigned to different nodes on a link, i.e. it is unique in the link.

Requirement: The implementation's interface identifier is also unique over a broader scope.

RFC text: `{{It is recommended that the same interface identifier not be assigned to different nodes on a link. They may also be unique over a broader scope}}`.

RQ_COR_1645 Interface Identifiers

RFC 3513 *Clause:* 2.5.1 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation uses an Interface identifier.

Requirement: The implementation's interface identifier, in some cases, is derived directly from that interface's link-layer address.

RFC text: `{{In some cases an interface's identifier will be derived directly from that interface's link-layer address}}`.

RQ_COR_1646 Interface Identifiers

RFC 3513 *Clause:* 2.5.1 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation uses an Interface identifier.

Requirement: The implementation uses the same interface identifier on multiple interfaces on a single node, as long as they are attached to different subnets.

RFC text: `{{The same interface identifier may be used on multiple interfaces on a single node, as long as they are attached to different subnets}}`.

RQ_COR_1647 Interface Identifiers

RFC 3513 *Clause:* 2.5.1 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses an Interface identifier.

Requirement: The implementation's Interface identifiers uniqueness is independent of the uniqueness of IPv6 addresses.

RFC text: `{{Note that the uniqueness of interface identifiers is independent of the uniqueness of IPv6 addresses}}`.

RQ_COR_1648 Interface Identifiers: Modified EUI64

RFC 3513 *Clause:* 2.5.1 ¶3, *Type:* MUST *applies to:* Node

Context: The implementation uses a unicast address that starts with binary value other than 000.

Requirement: The implementation's Interface identifier is 64 bits long and it is constructed in Modified EUI-64 format.

RFC text: `{{For all unicast addresses, except those that start with binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format}}`.

RQ_COR_1649 Interface Identifiers

RFC 3513 *Clause:* 2.5.1 ¶3, *Type:* MAY *applies to:* Node

Context: The implementation uses a unicast address that starts with binary value 000.

Requirement: The implementation's Interface identifier is not 64 bits long and it is not constructed in Modified EUI-64 format.

RFC text: `{{For all unicast addresses, except those that start with binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format}}`.

RQ_COR_1650 Interface Identifiers: Modified EUI64

RFC 3513 *Clause:* 2.5.1 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation derives a Modified EUI-64 format based Interface identifier from a global token.

Requirement: The implementation's Interface identifier scope is global.

RFC text: `{{Modified EUI-64 format based Interface identifiers may have global scope when derived from a global token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [EUI64]) or may have local scope where a global token is not available (e.g., serial links, tunnel end-points, etc.) or where global tokens are undesirable (e.g., temporary tokens for privacy [PRIV])}}`.

RQ_COR_1651 Interface Identifiers: Modified EUI64

RFC 3513 *Clause:* 2.5.1 ¶4 *Type:* MUST *applies to:* Node

Context: The implementation requires an interface identifier with global scope.

Requirement: The implementation uses an Modified EUI-64 format based Interface identifier derived from a global token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers).

RFC text: `{{Modified EUI-64 format based Interface identifiers may have global scope when derived from a global token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [EUI64])}}` or may have local scope where a global token is not available (e.g., serial links, tunnel end-points, etc.) or where global tokens are undesirable (e.g., temporary tokens for privacy [PRIV]).

RQ_COR_1652 Interface Identifiers: Modified EUI64

RFC 3513 *Clause:* 2.5.1 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation requires an interface identifier with only local scope.

Requirement: The implementation uses an Modified EUI-64 format based Interface identifier where a global token is not available (e.g., serial links, tunnel end-points, etc.) or where global tokens are undesirable (e.g., temporary tokens for privacy).

RFC text: Modified EUI-64 format based Interface identifiers may have global scope when derived from a global token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [EUI64]){{ or may have local scope where a global token is not available (e.g., serial links, tunnel end-points, etc.) or where global tokens are undesirable (e.g., temporary tokens for privacy [PRIV])}}.

RQ_COR_1653 Interface Identifiers (Built-in): Present

RFC 3513 *Clause:* 2.5.1 ¶5-8 *Type:* MUST *applies to:* Node

Context: The implementation generates a Modified EUI-64 format interface identifier from an IEEE EUI-64 identifier by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology). The initial state of "u" bit is 0, and the resulting Modified EUI-64 format interface identifier has global scope.

Requirement: The implementation sets the "u" bit to 1 indicating indicate global scope.

RFC text: {{Modified EUI-64 format interface identifiers are formed by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology) when forming the interface identifier from IEEE EUI-64 identifiers. In the resulting Modified EUI-64 format the "u" bit is set to one (1) to indicate global scope, and it is set to zero (0) to indicate local scope. The first three octets in binary of an IEEE EUI-64 identifier are as follows:... [image of first three octets in binary] ...written in Internet standard bit-order , where "u" is the universal/local bit, "g" is the individual/group bit, and "c" are the bits of the company_id}}. Appendix A: "Creating Modified EUI-64 format Interface Identifiers" provides examples on the creation of Modified EUI-64 format based interface identifiers. The motivation for inverting the "u" bit when forming an interface identifier is to make it easy for system administrators to hand configure non-global identifiers when hardware tokens are not available. This is expected to be case for serial links, tunnel end- points, etc.

RQ_COR_1654 Interface Identifiers (Built-in): Present

RFC 3513 *Clause:* 2.5.1 ¶5-8 *Type:* MUST *applies to:* Node

Context: The implementation generates a Modified EUI-64 format interface identifier from an IEEE EUI-64 identifier by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology). The initial state of "u" bit is 1, and the resulting Modified EUI-64 format interface identifier has local scope.

Requirement: The implementation sets the "u" bit to 0 indicating indicate local scope.

RFC text: {{Modified EUI-64 format interface identifiers are formed by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology) when forming the interface identifier from IEEE EUI-64 identifiers. In the resulting Modified EUI-64 format the "u" bit is set to one (1) to indicate global scope, and it is set to zero (0) to indicate local scope. The first three octets in binary of an IEEE EUI-64 identifier are as follows:... [image of first three octets in binary] ...written in Internet standard bit-order , where "u" is the universal/local bit, "g" is the individual/group bit, and "c" are the bits of the company_id}}. Appendix A: "Creating Modified EUI-64 format Interface Identifiers" provides examples on the creation of Modified EUI-64 format based interface identifiers. The motivation for inverting the "u" bit when forming an interface identifier is to make it easy for system administrators to hand configure non-global identifiers when hardware tokens are not available. This is expected to be case for serial links, tunnel end- points, etc.

RQ_COR_1655 address: Unspecified Address

RFC 3513 *Clause:* 2.5.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The address 0:0:0:0:0:0:0:0 is called the Unspecified address.

Requirement: The implementation never assigns the 0:0:0:0:0:0:0:0 address to any interface/node.

RFC text: `{{The address 0:0:0:0:0:0:0:0 is called the unspecified address. It must never be assigned to any node. It indicates the absence of an address}}`. One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address.

RQ_COR_1656 address: Unspecified Address

RFC 3513 *Clause:* 2.5.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The address 0:0:0:0:0:0:0:0 is called the Unspecified address.

Requirement: The implementation uses the Unspecified address in order to indicate the absence of an address.

RFC text: `{{The address 0:0:0:0:0:0:0:0 is called the unspecified address. It must never be assigned to any node. It indicates the absence of an address}}`. One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address.

RQ_COR_1657 address: Unspecified Address

RFC 3513 *Clause:* 2.5.2 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The address 0:0:0:0:0:0:0:0 is called the Unspecified address. The implementation is in an initial state before it has determined its own address.

Requirement: The implementation uses the Unspecified address in the Source Address field of any IPv6 packets.

RFC text: The address 0:0:0:0:0:0:0:0 is called the unspecified address. It must never be assigned to any node. It indicates the absence of an address. `{{One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address}}`.

RQ_COR_1658 address: Unspecified Address

RFC 3513 *Clause:* 2.5.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The address 0:0:0:0:0:0:0:0 is called the Unspecified address.

Requirement: The implementation does not use the Unspecified address as the destination address of IPv6 packets.

RFC text: `{{The unspecified address must not be used as the destination address of IPv6 packets or in IPv6 Routing Headers}}`. An IPv6 packet with a source address of unspecified must never be forwarded by an IPv6 router.

RQ_COR_1659 address: Unspecified Address

RFC 3513 *Clause:* 2.5.2 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The address 0:0:0:0:0:0:0:0 is called the Unspecified address.

Requirement: The implementation does not use the Unspecified address in IPv6 Routing Headers.

RFC text: `{{The unspecified address must not be used as the destination address of IPv6 packets or in IPv6 Routing Headers}}`.

RQ_COR_1660 address: Unspecified Address

RFC 3513 *Clause:* 2.5.2 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation uses IPv6. The address 0:0:0:0:0:0:0:0 is called the Unspecified address. The implementation receives an IPv6 packet with a source address set to the Unspecified address.

Requirement: The implementation does not forward the IPv6 packet.

RFC text: {{An IPv6 packet with a source address of unspecified must never be forwarded by an IPv6 router}}.

RQ_COR_1661 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address.

Requirement: The implementation uses the Loopback address to send an IPv6 packet to itself.

RFC text: {{The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself. It may never be assigned to any physical interface}}.

RQ_COR_1662 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶1 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address.

Requirement: The implementation treats the Loopback address as having link-local scope.

RFC text: The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself. It may never be assigned to any physical interface. {{It is treated as having link-local scope, and may be thought of as the link-local unicast address of a virtual interface (typically called "the loopback interface") to an imaginary link that goes nowhere}}.

RQ_COR_1663 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address.

Requirement: The implementation treats the Loopback address as the link-local unicast address of a virtual interface (typically called "the loopback interface") to an imaginary link that goes nowhere.

RFC text: The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself. It may never be assigned to any physical interface. {{It is treated as having link-local scope, and may be thought of as the link-local unicast address of a virtual interface (typically called "the loopback interface") to an imaginary link that goes nowhere}}.

RQ_COR_1664 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address.

Requirement: The implementation does not use the Loopback address as the source address in IPv6 packets that are sent outside of the implementation.

RFC text: {{The loopback address must not be used as the source address in IPv6 packets that are sent outside of a single node}}.

RQ_COR_1665 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address.

Requirement: The implementation does not send outside of the implementation an IPv6 packet with a destination address of Loopback.

RFC text: `{{An IPv6 packet with a destination address of loopback must never be sent outside of a single node and must never be forwarded by an IPv6 router}}`.

RQ_COR_1666 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address. The implementation receives an IPv6 packet with a destination address set to the Loopback address.

Requirement: The implementation does not forward the IPv6 packet.

RFC text: `{{An IPv6 packet with a destination address of loopback must never be sent outside of a single node and must never be forwarded by an IPv6 router}}`.

RQ_COR_1667 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address. The implementation receives an IPv6 packet with a destination address set to the Loopback address.

Requirement: The implementation drops the IPv6 packet.

RFC text: `{{A packet received on an interface with destination address of loopback must be dropped}}`.

RQ_COR_1668 address: Unicast

RFC 3513 *Clause:* 2.5.4 ¶1-3 *Type:* MUST *applies to:* Node

Context: The implementation uses IPv6 Global Unicast Addresses.

Requirement: The implementation uses the following general format for IPv6 global unicast addresses: n bits for global routing prefix, m bits for subnet ID, 128-n-m bits for interface ID where the global routing prefix is a (typically hierarchically- structured) value assigned to a site (a cluster of subnets/links), the subnet ID is an identifier of a link within the site, and the interface ID is as defined in section 2.5.1

RFC text: `{{The general format for IPv6 global unicast addresses is as follows: [n bits for global routing prefix, m bits for subnet ID, 128-n-m bits for interface ID] where the global routing prefix is a (typically hierarchically- structured) value assigned to a site (a cluster of subnets/links), the subnet ID is an identifier of a link within the site, and the interface ID is as defined in section 2.5.1}}`.

RQ_COR_1669 Mapping of IPv4 Addresses

RFC 3513 *Clause:* 2.5.5 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses a technique for hosts and routers to dynamically tunnel IPv6 packets over IPv4 routing infrastructure.

Requirement: The implementation is assigned special IPv6 unicast addresses that carry a global IPv4 address in the low-order 32 bits. That unicast address is named "IPv4-compatible IPv6 address".

RFC text: `{{The IPv6 transition mechanisms [TRAN] include a technique for hosts and routers to dynamically tunnel IPv6 packets over IPv4 routing infrastructure. IPv6 nodes that use this technique are assigned special IPv6 unicast addresses that carry a global IPv4 address in the low-order 32 bits. This type of address is termed an "IPv4-compatible IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to 0000, last 32 bits set to IPv4 address] Note: The IPv4 address used in the "IPv4-compatible IPv6 address" must be a globally-unique IPv4 unicast address}}`. A second type of IPv6 address which holds an embedded IPv4 address is also defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. This type of address is termed an "IPv4-mapped IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to FFFF, last 32 bits set to IPv4 address].

RQ_COR_1670 Mapping of IPv4 Addresses

RFC 3513 *Clause:* 2.5.5 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses an "IPv4-compatible IPv6 address" [deprecated].

Requirement: The implementation uses the following general format for "IPv4-compatible IPv6 address": first 80 bits set to 0, next 16 bits set to 0000, last 32 bits set to IPv4 address.

RFC text: `{{The IPv6 transition mechanisms [TRAN] include a technique for hosts and routers to dynamically tunnel IPv6 packets over IPv4 routing infrastructure. IPv6 nodes that use this technique are assigned special IPv6 unicast addresses that carry a global IPv4 address in the low-order 32 bits. This type of address is termed an "IPv4-compatible IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to 0000, last 32 bits set to IPv4 address] Note: The IPv4 address used in the "IPv4-compatible IPv6 address" must be a globally-unique IPv4 unicast address}}`. A second type of IPv6 address which holds an embedded IPv4 address is also defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. This type of address is termed an "IPv4-mapped IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to FFFF, last 32 bits set to IPv4 address].

RQ_COR_1671 Mapping of IPv4 Addresses

RFC 3513 *Clause:* 2.5.5 ¶1-3 *Type:* MUST *applies to:* Node

Context: The implementation uses an "IPv4-compatible IPv6 address" [deprecated].

Requirement: The implementation uses a globally-unique IPv4 unicast address in the low-order 32 bits of the "IPv4-compatible IPv6 address".

RFC text: The IPv6 transition mechanisms [TRAN] include a technique for hosts and routers to dynamically tunnel IPv6 packets over IPv4 routing infrastructure. IPv6 nodes that use this technique are assigned special IPv6 unicast addresses that carry a global IPv4 address in the low-order 32 bits. This type of address is termed an "IPv4-compatible IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to 0000, last 32 bits set to IPv4 address] `{{Note: The IPv4 address used in the "IPv4-compatible IPv6 address" must be a globally-unique IPv4 unicast address}}`. A second type of IPv6 address which holds an embedded IPv4 address is also defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. This type of address is termed an "IPv4-mapped IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to FFFF, last 32 bits set to IPv4 address].

RQ_COR_1672 Mapping of IPv4 Addresses

RFC 3513 *Clause:* 2.5.5 ¶4-5 *Type:* MUST *applies to:* Node

Context: The implementation uses a technique to represent the addresses of IPv4 nodes as IPv6 addresses.

Requirement: The implementation is assigned special IPv6 unicast addresses that carry a global IPv4 address in the low-order 32 bits. That unicast address is named "IPv4-mapped IPv6 address".

RFC text: The IPv6 transition mechanisms [TRAN] include a technique for hosts and routers to dynamically tunnel IPv6 packets over IPv4 routing infrastructure. IPv6 nodes that use this technique are assigned special IPv6 unicast addresses that carry a global IPv4 address in the low-order 32 bits. This type of address is termed an "IPv4-compatible IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to 0000, last 32 bits set to IPv4 address] Note: The IPv4 address used in the "IPv4-compatible IPv6 address" must be a globally-unique IPv4 unicast address. `{{A second type of IPv6 address which holds an embedded IPv4 address is also defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. This type of address is termed an "IPv4-mapped IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to FFFF, last 32 bits set to IPv4 address]}}`.

RQ_COR_1673 Mapping of IPv4 Addresses

RFC 3513 *Clause:* 2.5.5 ¶4-5 *Type:* MUST *applies to:* Node

Context: The implementation uses an "IPv4-mapped IPv6 address".

Requirement: The implementation uses the following general format for "IPv4-mapped IPv6 address": first 80 bits set to 0, next 16 bits set to FFFF, last 32 bits set to IPv4 address.

RFC text: The IPv6 transition mechanisms [TRAN] include a technique for hosts and routers to dynamically tunnel IPv6 packets over IPv4 routing infrastructure. IPv6 nodes that use this technique are assigned special IPv6 unicast addresses that carry a global IPv4 address in the low-order 32 bits. This type of address is termed an "IPv4-compatible IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to 0000, last 32 bits set to IPv4 address] Note: The IPv4 address used in the "IPv4-compatible IPv6 address" must be a globally-unique IPv4 unicast address. `{{A second type of IPv6 address which holds an embedded IPv4 address is also defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. This type of address is termed an "IPv4-mapped IPv6 address" and has the format: [first 80 bits set to 0, next 16 bits set to FFFF, last 32 bits set to IPv4 address]}}`.

RQ_COR_1674 address: Link-local

RFC 3513 *Clause:* 2.5.6 ¶1-3 *Type:* MUST *applies to:* Node

Context: The implementation uses a local-use unicast address.

Requirement: The implementation uses a Link-Local address used for addressing on a single link for purposes such as automatic address configuration, neighbor discovery, or when no routers are present.

RFC text: `{{There are two types of local-use unicast addresses defined. These are Link-Local and Site-Local [deprecated]. The Link-Local is for use on a single link and the Site-Local [deprecated] is for use in a single site}}`. Link-Local addresses have the following format: [firts 10 bits set to 1111111010, next 54 bits set to 0, last 64 bits set to interface ID]. `{{Link-Local addresses are designed to be used for addressing on a single link for purposes such as automatic address configuration, neighbor discovery, or when no routers are present}}`. Routers must not forward any packets with link-local source or destination addresses to other links.

RQ_COR_1675 address: Link-local

RFC 3513 *Clause:* 2.5.6 ¶1-3 *Type:* MUST *applies to:* Node

Context: The implementation uses a Link-Local address.

Requirement: The implementation uses the following general format for Link-Local address: first 10 bits set to 1111111010, next 54 bits set to 0, last 64 bits set to interface ID.

RFC text: There are two types of local-use unicast addresses defined. These are Link-Local and Site-Local [deprecated]. The Link-Local is for use on a single link and the Site-Local [deprecated] is for use in a single site. `{{Link-Local addresses have the following format: [firts 10 bits set to 1111111010, next 54 bits set to 0, last 64 bits set to interface ID]}}`. Link-Local addresses are designed to be used for addressing on a single link for purposes such as automatic address configuration, neighbor discovery, or when no routers are present. Routers must not forward any packets with link-local source or destination addresses to other links.

RQ_COR_1676 address: Link-local

RFC 3513 *Clause:* 2.5.6 ¶4 *Type:* MUST *applies to:* Router

Context: The implementation receives a packet with a link-local address as source address. [The link-local address is not the implementation's].

Requirement: The implementation does not forward the packet to other links.

RFC text: `{{Routers must not forward any packets with link-local source or destination addresses to other links}}`.

RQ_COR_1677 address: Link-local

RFC 3513 *Clause:* 2.5.6 ¶4 *Type:* MUST *applies to:* Router

Context: The implementation receives a packet with a link-local address as destination address. [The link-local address is none of the implementation].

Requirement: The implementation does not forward the packet to other links.

RFC text: `{{Routers must not forward any packets with link-local source or destination addresses to other links}}`.

RQ_COR_1678 address: Anycast

RFC 3513 *Clause:* 2.6 ¶2 *Type:* MUST *applies to:* Node

Context: The implementation is being configured.

Requirement: The implementation allows explicit configuration of each anycast address assigned to it.

RFC text: Anycast addresses are allocated from the unicast address space, using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. `{{When a unicast address is assigned to more than one interface, thus turning it into an anycast address, the nodes to which the address is assigned must be explicitly configured to know that it is an anycast address}}`.

RQ_COR_1679 address: Anycast

RFC 3513 *Clause:* 2.6 ¶2 *Type:* MUST *applies to:* Router

Context: The implementation receives a unicast address to more than one interface (typically along with different nodes), thus turning that Unicast address into an Anycast address.

Requirement: The implementation is explicitly configured to know that the received address is an Anycast address.

RFC text: Anycast addresses are allocated from the unicast address space, using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. `{{When a unicast address is assigned to more than one interface, thus turning it into an anycast address, the nodes to which the address is assigned must be explicitly configured to know that it is an anycast address}}`.

RQ_COR_1680 address: Anycast

RFC 3513 *Clause:* 2.6 ¶3 *Type:* MUST *applies to:* Router

Context: The implementation receives an Anycast address, that has a longest prefix P that identifies the topological region in which all interfaces belonging to that anycast address reside.

Requirement: The implementation maintains, within the region identified by P, the anycast address as a separate entry in the routing system (commonly referred to as a "host route").

RFC text: `{{For any assigned anycast address, there is a longest prefix P of that address that identifies the topological region in which all interfaces belonging to that anycast address reside. Within the region identified by P, the anycast address must be maintained as a separate entry in the routing system (commonly referred to as a "host route"); outside the region identified by P, the anycast address may be aggregated into the routing entry for prefix P}}`.

RQ_COR_1681 address: Anycast

RFC 3513 *Clause:* 2.6 ¶3 *Type:* MAY *applies to:* Router

Context: The implementation receives an Anycast address, that has a longest prefix P that identifies the topological region in which all interfaces belonging to that anycast address reside.

Requirement: The implementation aggregates, outside the region identified by P, the anycast address into the routing entry for prefix P.

RFC text: `{{For any assigned anycast address, there is a longest prefix P of that address that identifies the topological region in which all interfaces belonging to that anycast address reside. Within the region identified by P, the anycast address must be maintained as a separate entry in the routing system (commonly referred to as a "host route"); outside the region identified by P, the anycast address may be aggregated into the routing entry for prefix P}}`.

RQ_COR_1682 address: Anycast

RFC 3513 *Clause:* 2.6 ¶4 *Type:* MUST *applies to:* Router

Context: The implementation receives an Anycast address, that has a longest prefix P set to the null prefix (i.e., the members of the set have no topological locality).

Requirement: The implementation maintains the anycast address as a separate routing entry throughout the entire internet.

RFC text: {{Note that in the worst case, the prefix P of an anycast set may be the null prefix, i.e., the members of the set may have no topological locality. In that case, the anycast address must be maintained as a separate routing entry throughout the entire internet, which presents a severe scaling limit on how many such "global" anycast sets may be supported. Therefore, it is expected that support for global anycast sets may be unavailable or very restricted}}.

RQ_COR_1683 address: Anycast

RFC 3513 *Clause:* 2.6 ¶7-8 *Type:* MUST *applies to:* Router

Context: The implementation is generating an IPv6 packet.

Requirement: The implementation does not use an anycast address as the source address of an IPv6 packet.

RFC text: {{There is little experience with widespread, arbitrary use of internet anycast addresses, and some known complications and hazards when using them in their full generality [ANYCST]. Until more experience has been gained and solutions are specified, the following restrictions are imposed on IPv6 anycast addresses: - An anycast address must not be used as the source address of an IPv6 packet. - An anycast address must not be assigned to an IPv6 host, that is, it may be assigned to an IPv6 router only}}.

RQ_COR_1684 address: Anycast

RFC 3513 *Clause:* 2.6 ¶7-8 *Type:* MUST *applies to:* Host

Context: The implementation uses IPv6.

Requirement: The implementation does not have an Anycast address.

RFC text: {{There is little experience with widespread, arbitrary use of internet anycast addresses, and some known complications and hazards when using them in their full generality [ANYCST]. Until more experience has been gained and solutions are specified, the following restrictions are imposed on IPv6 anycast addresses: - An anycast address must not be used as the source address of an IPv6 packet. - An anycast address must not be assigned to an IPv6 host, that is, it may be assigned to an IPv6 router only}}.

RQ_COR_1685 address: Anycast

RFC 3513 *Clause:* 2.6 ¶7-8 *Type:* MAY *applies to:* Router

Context: The implementation uses IPv6. (Nowadays, during the early stages of Anycast usage in networks where are some known complications and hazards when using them in their full generality, is suggested that:)

Requirement: The implementation has an Anycast address(es).

RFC text: {{There is little experience with widespread, arbitrary use of internet anycast addresses, and some known complications and hazards when using them in their full generality [ANYCST]. Until more experience has been gained and solutions are specified, the following restrictions are imposed on IPv6 anycast addresses: - An anycast address must not be used as the source address of an IPv6 packet. - An anycast address must not be assigned to an IPv6 host, that is, it may be assigned to an IPv6 router only}}.

RQ_COR_1686

RFC 3513 *Clause:* 2.6.1 ¶1-3 *Type:* MAY *applies to:* Router

Context: The implementation uses Anycast addresses.

Requirement: The implementation uses the predefined Subnet-Router Anycast address.

RFC text: `{{The Subnet-Router anycast address is predefined. Its format is as follows: [first n bits for subnet prefix, latest 128-n bits set to 0]. The "subnet prefix" in an anycast address is the prefix which identifies a specific link. This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero}}`.

RQ_COR_1687 address: Anycast

RFC 3513 *Clause:* 2.6.1 ¶1-3 *Type:* MUST *applies to:* Router

Context: The implementation uses the predefined Subnet-Router Anycast address.

Requirement: The implementation's Subnet-Router Anycast address has the following format: [first n bits for subnet prefix, latest 128-n bits set to 0], where the "subnet prefix" is the prefix which identifies a specific link.

RFC text: `{{The Subnet-Router anycast address is predefined. Its format is as follows: [first n bits for subnet prefix, latests 128-n bits set to 0]. The "subnet prefix" in an anycast address is the prefix which identifies a specific link. This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero}}`.

RQ_COR_1688 address: Anycast

RFC 3513 *Clause:* 2.6.1 ¶4 *Type:* MUST *applies to:* Router

Context: The implementation uses Anycast addresses.

Requirement: The implementation supports the Subnet-Router Anycast addresses for the subnets to which they have interfaces.

RFC text: `{{All routers are required to support the Subnet-Router anycast addresses for the subnets to which they have interfaces}}`.

RQ_COR_1689 address: Multicast

RFC 3513 *Clause:* 2.7 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation uses Multicast addresses. An IPv6 Multicast address is an identifier for a group of interfaces (typically on different nodes).

Requirement: The implementation's interface belongs to any number of multicast groups.

RFC text: `{{An IPv6 multicast address is an identifier for a group of interfaces (typically on different nodes). An interface may belong to any number of multicast groups}}`.

RQ_COR_1690 address: Multicast

RFC 3513 *Clause:* 2.7 ¶1-15 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. The implementation generates a Multicast address.

Requirement: The implementation's Multicast address has the following format: [First 8 bits set to 11111111, next 4 bits for flgs, next 4 bits for scop, latests 112 bits for group ID]. binary 11111111 at the start of the address identifies the address as being a multicast address. flgs is a set of 4 flags: [0 0 0 T]. The high-order 3 flags are reserved, and must be initialized to 0. T = 0 indicates a permanently-assigned ("well-known") multicast address, assigned by the Internet Assigned Number Authority (IANA). T = 1 indicates a non-permanently-assigned ("transient") multicast address. scop is a 4-bit multicast scope value used to limit the scope of the multicast group. The values are: 0 reserved. 1 interface-local scope. 2 link-local scope. 3 reserved. 4 admin-local scope. 5 site-local scope. 6 (unassigned). 7 (unassigned). 8 organization-local scope. 9 (unassigned). A (unassigned). B (unassigned). C (unassigned). D (unassigned). E global scope. F reserved. Interface-local scope spans only a single interface on a node, and is useful only for loopback transmission of multicast. Link-local and site-local multicast scopes span the same topological regions as the corresponding unicast scopes. Admin-local scope is the smallest scope that must be administratively configured, i.e., not automatically derived from physical connectivity or other, non- multicast-related configuration. Organization-local scope is intended to span multiple sites belonging to a single organization. The scopes labeled "(unassigned)" are available for administrators to define additional multicast regions. Finnaly, the group ID identifies the multicast group, either permanent or transient, within the given scope.

RFC text: {{Multicast addresses have the following format: [First 8 bits set to 11111111, next 4 bits for flgs, next 4 bits for scop, latests 112 bits for group ID]. binary 11111111 at the start of the address identifies the address as being a multicast address. flgs is a set of 4 flags: [0 0 0 T]. The high-order 3 flags are reserved, and must be initialized to 0. T = 0 indicates a permanently-assigned ("well-known") multicast address, assigned by the Internet Assigned Number Authority (IANA). T = 1 indicates a non-permanently-assigned ("transient") multicast address. scop is a 4-bit multicast scope value used to limit the scope of the multicast group. The values are: 0 reserved. 1 interface-local scope. 2 link-local scope. 3 reserved. 4 admin-local scope. 5 site-local scope. 6 (unassigned). 7 (unassigned). 8 organization-local scope. 9 (unassigned). A (unassigned). B (unassigned). C (unassigned). D (unassigned). E global scope. F reserved. interface-local scope spans only a single interface on a node, and is useful only for loopback transmission of multicast. link-local and site-local multicast scopes span the same topological regions as the corresponding unicast scopes. admin-local scope is the smallest scope that must be administratively configured, i.e., not automatically derived from physical connectivity or other, non- multicast-related configuration. organization-local scope is intended to span multiple sites belonging to a single organization. scopes labeled "(unassigned)" are available for administrators to define additional multicast regions. group ID identifies the multicast group, either permanent or transient, within the given scope}}.

RQ_COR_1705 address: Multicast

RFC 3513 *Clause:* 2.7 ¶16-20 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. The implementation has a Permanently-assigned Multicast address (i.e. the T flgs is set to 0 indicating a permanently-assigned ("well-known") multicast address, assigned by IANA).

Requirement: The implementation's Permanently-assigned multicast address is independent of the scope value.

RFC text: {{The "meaning" of a permanently-assigned multicast address is independent of the scope value}}. For example, if the "NTP servers group" is assigned a permanent multicast address with a group ID of 101 (hex), then: FF01:0:0:0:0:0:0:101 means all NTP servers on the same interface, (i.e., the same node) as the sender. FF02:0:0:0:0:0:0:101 means all NTP servers on the same link as the sender. FF05:0:0:0:0:0:0:101 means all NTP servers in the same site as the sender. FF0E:0:0:0:0:0:0:101 means all NTP servers in the internet.}}.

RQ_COR_1706 address: Multicast

RFC 3513 *Clause:* 2.7 ¶21 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. The implementation has a Non-Permanently-assigned Multicast address (i.e. the T flag is set to 1, indicating a non-permanently-assigned ("transient") multicast address).

Requirement: The implementation's Non-Permanently-assigned multicast address is not independent of the scope value, i.e. Non-permanently-assigned multicast addresses are meaningful only within a given scope.

RFC text: `{{Non-permanently-assigned multicast addresses are meaningful only within a given scope}}`. For example, a group identified by the non- permanent, site-local multicast address FF15:0:0:0:0:0:0:101 at one site bears no relationship to a group using the same address at a different site, nor to a non-permanent group using the same group ID with different scope, nor to a permanent group with the same group ID.

RQ_COR_1707 address: Multicast Address Behavior

RFC 3513 *Clause:* 2.7 ¶22 *Type:* MUST *applies to:* Node

Context: The implementation generates an IPv6 packet.

Requirement: The implementation does not use a Multicast address as a source addresses in an IPv6 packet.

RFC text: `{{Multicast addresses must not be used as source addresses in IPv6 packets or appear in any Routing header}}`.

RQ_COR_1708 address: Multicast Address Behavior

RFC 3513 *Clause:* 2.7 ¶22 *Type:* MUST *applies to:* Node

Context: The implementation generates an IPv6 packet with Routing header.

Requirement: The implementation does not use a Multicast address in the Routing header.

RFC text: `{{Multicast addresses must not be used as source addresses in IPv6 packets or appear in any Routing header}}`.

RQ_COR_1709 address: Multicast Address Behavior

RFC 3513 *Clause:* 2.7 ¶23 *Type:* MUST *applies to:* Router

Context: The implementation receives an IPv6 multicast packet with certain scope in the destination multicast address.

Requirement: The implementation does not forward the multicast packet beyond the scope indicated by the scop field in the destination multicast address.

RFC text: `{{Routers must not forward any multicast packets beyond of the scope indicated by the scop field in the destination multicast address}}`.

RQ_COR_1710 address: Multicast Address Behavior

RFC 3513 *Clause:* 2.7 ¶24 *Type:* MUST *applies to:* Node

Context: The implementation generates an IPv6 multicast packet.

Requirement: The implementation does not generate a packet to a multicast address whose scop field contains the reserved value 0.

RFC text: `{{Nodes must not originate a packet to a multicast address whose scop field contains the reserved value 0; if such a packet is received, it must be silently dropped}}`.

RQ_COR_1711 address: Multicast Address Behavior

RFC 3513 *Clause:* 2.7 ¶24 *Type:* MUST *applies to:* Node

Context: The implementation receives an IPv6 multicast packet whose scop field contains the reserved value 0.

Requirement: The implementation silently drops the multicast packet.

RFC text: `{{Nodes must not originate a packet to a multicast address whose scop field contains the reserved value 0; if such a packet is received, it must be silently dropped}}`.

RQ_COR_1712 address: Multicast Address Behavior

RFC 3513 *Clause:* 2.7 ¶24 *Type:* SHOULD *applies to:* Node

Context: The implementation generates an IPv6 multicast packet.

Requirement: The implementation does not generate a packet to a multicast address whose scop field contains the reserved value F.

RFC text: `{{Nodes should not originate a packet to a multicast address whose scop field contains the reserved value F; if such a packet is sent or received, it must be treated the same as packets destined to a global (scop E) multicast address}}`.

RQ_COR_1713 address: Multicast Address Behavior

RFC 3513 *Clause:* 2.7 ¶24 *Type:* MUST *applies to:* Node

Context: The implementation receives an IPv6 multicast packet whose scop field contains the reserved value F.

Requirement: The implementation treats the packet as same as a packet destined to a global (scop E) multicast address.

RFC text: `{{Nodes should not originate a packet to a multicast address whose scop field contains the reserved value F; if such a packet is sent or received, it must be treated the same as packets destined to a global (scop E) multicast address}}`.

RQ_COR_1714 address: Multicast

RFC 3513 *Clause:* 2.7.1 ¶1-2 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. The group ID's [FF0X:0:0:0:0:0:0:0, FF0X:0:0:0:0:0:0:1, FF0X:0:0:0:0:0:0:2, FF0X:0:0:0:0:1:FFXX:XXXX] defined in RFC 3513, 2.7.1 are defined for explicit scope values.

Requirement: The implementation does not use these group IDs for any other scope values with the T flag equal to 0.

RFC text: The following well-known multicast addresses [multicast addresses of 2.7.1] are pre-defined. `{{The group ID's defined in this section are defined for explicit scope values. Use of these group IDs for any other scope values, with the T flag equal to 0, is not allowed}}`.

RQ_COR_1715 address: Multicast

RFC 3513 *Clause:* 2.7.1 ¶3-4 *Type:* SHALL *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 protocol pre-defines the following Reserved Multicast Addresses: FF00:0:0:0:0:0:0:0, FF01:0:0:0:0:0:0:0, FF02:0:0:0:0:0:0:0, FF03:0:0:0:0:0:0:0, FF04:0:0:0:0:0:0:0, FF05:0:0:0:0:0:0:0, FF06:0:0:0:0:0:0:0, FF07:0:0:0:0:0:0:0, FF08:0:0:0:0:0:0:0, FF09:0:0:0:0:0:0:0, FF0A:0:0:0:0:0:0:0, FF0B:0:0:0:0:0:0:0, FF0C:0:0:0:0:0:0:0, FF0D:0:0:0:0:0:0:0, FF0E:0:0:0:0:0:0:0, FF0F:0:0:0:0:0:0:0.

Requirement: The implementation does not assign these multicast addresses to any multicast group.

RFC text:
```
{{Reserved Multicast Addresses:  FF00:0:0:0:0:0:0:0,
FF01:0:0:0:0:0:0:0, FF02:0:0:0:0:0:0:0, FF03:0:0:0:0:0:0:0,
FF04:0:0:0:0:0:0:0, FF05:0:0:0:0:0:0:0, FF06:0:0:0:0:0:0:0,
FF07:0:0:0:0:0:0:0, FF08:0:0:0:0:0:0:0, FF09:0:0:0:0:0:0:0,
FF0A:0:0:0:0:0:0:0, FF0B:0:0:0:0:0:0:0, FF0C:0:0:0:0:0:0:0,
FF0D:0:0:0:0:0:0:0, FF0E:0:0:0:0:0:0:0, FF0F:0:0:0:0:0:0:0.
The above multicast addresses are reserved and shall never be
assigned to any multicast group}}.
```

RQ_COR_1716 address: All-Nodes Multicast

RFC 3513 *Clause:* 2.7.1 ¶5-6 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following All Nodes Addresses: FF01:0:0:0:0:0:0:1, FF02:0:0:0:0:0:0:1.

Requirement: The implementation uses the FF01:0:0:0:0:0:0:1 address to identify the group of all IPv6 nodes within scope 1 (interface-local).

RFC text:
```
{{All Nodes Addresses: FF01:0:0:0:0:0:0:1, FF02:0:0:0:0:0:0:1.
The above multicast addresses identify the group of all IPv6 nodes,
within scope 1 (interface-local) or 2 (link-local)}}.
```

RQ_COR_1717 address: All-Nodes Multicast

RFC 3513 *Clause:* 2.7.1 ¶5-6 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following All Nodes Addresses: FF01:0:0:0:0:0:0:1, FF02:0:0:0:0:0:0:1.

Requirement: The implementation uses the FF02:0:0:0:0:0:0:1 address to identify the group of all IPv6 nodes within scope 2 (link-local).

RFC text:
```
{{All Nodes Addresses:  FF01:0:0:0:0:0:0:1, FF02:0:0:0:0:0:0:1. The
above multicast addresses identify the group of all IPv6 nodes,
within scope 1 (interface-local) or 2 (link-local)}}.
```

RQ_COR_1718 address: All-Router Multicast

RFC 3513 *Clause:* 2.7.1 ¶7-8 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following All Routers Addresses: FF01:0:0:0:0:0:0:2, FF02:0:0:0:0:0:0:2, FF00:0:0:0:0:0:0:0.

Requirement: The implementation uses the FF01:0:0:0:0:0:0:2 address to identify the group of all IPv6 routers within scope 1 (interface-local).

RFC text:
```
{{All Routers Addresses: FF01:0:0:0:0:0:0:2, FF02:0:0:0:0:0:0:2,
FF05:0:0:0:0:0:0:0. The above multicast addresses identify the group
of all IPv6 routers, within scope 1 (interface-local), 2
(link-local), or 5 (site-local)}}.
```

RQ_COR_1719 address: All-Router Multicast

RFC 3513 *Clause:* 2.7.1 ¶7-8 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following All Routers Addresses: FF01:0:0:0:0:0:0:2, FF02:0:0:0:0:0:0:2, FF00:0:0:0:0:0:0:0.

Requirement: The implementation uses the FF02:0:0:0:0:0:0:2 address to identify the group of all IPv6 routers within scope 2 (link-local).

RFC text: {{All Routers Addresses: FF01:0:0:0:0:0:0:2, FF02:0:0:0:0:0:0:2, FF05:0:0:0:0:0:0:0. The above multicast addresses identify the group of all IPv6 routers, within scope 1 (interface-local), 2 (link-local), or 5 (site-local)}}.

RQ_COR_1720

RFC 3513 *Clause:* 2.7.1 ¶7-8 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following All Routers Addresses: FF01:0:0:0:0:0:0:2, FF02:0:0:0:0:0:0:2, FF00:0:0:0:0:0:0:0.

Requirement: The implementation uses the FF01:0:0:0:0:0:0:2 address to identify the group of all IPv6 routers within scope 5 (site-local).

RFC text: {{All Routers Addresses: FF01:0:0:0:0:0:0:2, FF02:0:0:0:0:0:0:2, FF05:0:0:0:0:0:0:0.
The above multicast addresses identify the group of all IPv6 routers, within scope 1 (interface-local), 2 (link-local), or 5 (site-local)}}.

RQ_COR_1721 address: Solicited-Node Multicast

RFC 3513 *Clause:* 2.7.1 ¶9-14 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX. The implementation needs to generate the corresponding Solicited-Node Address of a certain Unicast address.

Requirement: The implementation forms that Solicited-node multicast address by taking the low-order 24 bits of the unicast address and appending those bits to the prefix FF02:0:0:0:0:1:FF00::/104 resulting in a multicast address in the range FF02:0:0:0:0:1:FF00:0000 to FF02:0:0:0:0:1:FFFF:FFFF.

RFC text: {{Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX Solicited-node multicast address are computed as a function of a node's unicast and anycast addresses. A solicited-node multicast address is formed by taking the low-order 24 bits of the Unicast address and appending those bits to the prefix FF02:0:0:0:0:1:FF00::/104 resulting in a multicast address in the range FF02:0:0:0:0:1:FF00:0000 to FF02:0:0:0:0:1:FFFF:FFFF}}. For example, the solicited node multicast address corresponding to the IPv6 address 4037::01:800:200E:8C6C is FF02::1:FF0E:8C6C.

RQ_COR_1722 address: Solicited-Node Multicast

RFC 3513 *Clause:* 2.7.1 ¶9-14 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX. The implementation needs to generate the corresponding Solicited-Node Address of a certain Anycast address.

Requirement: The implementation forms that Solicited-node multicast address by taking the low-order 24 bits of the anycast address and appending those bits to the prefix FF02:0:0:0:0:1:FF00::/104 resulting in a multicast address in the range FF02:0:0:0:0:1:FF00:0000 to FF02:0:0:0:0:1:FFFF:FFFF.

RFC text: `{{Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX Solicited-node multicast address are computed as a function of a node's unicast and anycast addresses. A solicited-node multicast address is formed by taking the low-order 24 bits of an Anycast address and appending those bits to the prefix FF02:0:0:0:0:1:FF00::/104 resulting in a multicast address in the range FF02:0:0:0:0:1:FF00:0000 to FF02:0:0:0:0:1:FFFF:FFFF}}`. For example, the solicited node multicast address corresponding to the IPv6 address 4037::01:800:200E:8C6C is FF02::1:FF0E:8C6C.

RQ_COR_1723 address: Solicited-Node Multicast

RFC 3513 *Clause:* 2.7.1 ¶14 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX. The implementation needs to generate the corresponding Solicited-Node Addresses of a several Unicast addresses that differ only in the high-order bits.

Requirement: The implementation maps these Unicast addresses to the same Solicited-Node address thereby, reducing the number of multicast addresses a node must join.

RFC text: `{{IPv6 addresses that differ only in the high-order bits, e.g., due to multiple high-order prefixes associated with different aggregations, will map to the same solicited-node address thereby, reducing the number of multicast addresses a node must join}}`.

RQ_COR_1724 address: Solicited-Node Multicast

RFC 3513 *Clause:* 2.7.1 ¶14 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX. The implementation needs to generate the corresponding Solicited-Node Addresses of a several Anycast addresses that differ only in the high-order bits.

Requirement: The implementation maps these Anycast addresses to the same Solicited-Node address thereby, reducing the number of multicast addresses a node must join.

RFC text: `{{IPv6 addresses that differ only in the high-order bits, e.g., due to multiple high-order prefixes associated with different aggregations, will map to the same solicited-node address thereby, reducing the number of multicast addresses a node must join}}`.

RQ_COR_1725 address: Solicited-Node Multicast

RFC 3513 *Clause:* 2.7.1 ¶15 *Type:* MUST *applies to:* Node

Context: The implementation uses Multicast addresses. IPv6 pre-defines the following Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX.

Requirement: The implementation is required to generate and join (on the appropriate interface) the associated Solicited-Node multicast addresses for every unicast and anycast address it is assigned.

RFC text: `{{A node is required to compute and join (on the appropriate interface) the associated Solicited-Node multicast addresses for every unicast and anycast address it is assigned}}`.

RQ_COR_1726 Address Architecture

RFC 3513 *Clause:* 2.8 ¶1-2 *Type:* MUST *applies to:* Host

Context: The implementation uses IPv6.

Requirement: The implementation is required to recognize the following addresses as identifying itself: (1) Its required Link-Local Address for each interface. (2) Any additional Unicast and Anycast Addresses that have been configured for the node's interfaces (manually or automatically). (3) The loopback address. (4) The All-Nodes Multicast Addresses defined in section 2.7.1. (5) The Solicited-Node Multicast Address for each of its unicast and anycast addresses. (6) Multicast Addresses of all other groups to which the node belongs.

RFC text:
```
{{A host is required to recognize the following addresses as
identifying itself: Its required Link-Local Address for each
interface.
Any additional Unicast and Anycast Addresses that have been
configured for the node's interfaces (manually or automatically). The
loopback address.
The All-Nodes Multicast Addresses defined in section 2.7.1.
The Solicited-Node Multicast Address for each of its unicast and
anycast addresses.  Multicast Addresses of all other groups to which
the node belongs}}.
```

RQ_COR_1727 Address Architecture

RFC 3513 *Clause:* 2.8 ¶1-2 *Type:* MUST *applies to:* Router

Context: The implementation uses IPv6.

Requirement: The implementation is required to recognize all addresses that a host is required to recognize, plus the following addresses as identifying itself: (1) The Subnet-Router Anycast Addresses for all interfaces for which it is configured to act as a router. (2) All other Anycast Addresses with which the router has been configured. (3) The All-Routers Multicast Addresses defined in section 2.7.1.

RFC text:
```
{{A router is required to recognize all addresses that a host is
required to recognize, plus the following addresses as identifying
itself:  The Subnet-Router Anycast Addresses for all interfaces for
which it is configured to act as a router. All other Anycast
Addresses with which the router has been configured. The All-Routers
Multicast Addresses defined in section 2.7.1}}.
```

RQ_COR_1728 Interface Identifiers (Built-in): Present

RFC 3513 *Clause:* A ¶2-7 *Type:* MUST *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier from an IEEE EUI-64 identifier.

Requirement: The implementation inverts the "u" (universal/local) bit.

RFC text:
```
{{Links or Nodes with IEEE EUI-64 Identifiers.  The only change
needed to transform an IEEE EUI-64 identifier to an interface
identifier is to invert the "u" (universal/local) bit. For example, a
globally unique IEEE EUI-64 identifier of the form:
[cccccc0gcccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|mmmmmmmmmmmmmmmm]
where "c" are the bits of the assigned company_id, "0" is the value
of the universal/local bit to indicate global scope, "g" is
individual/group bit, and "m" are the bits of the manufacturer-
selected extension identifier. The IPv6 interface identifier would be
of the form:
[cccccc1gcccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm|mmmmmmmmmmmmmmmm]
The only change is inverting the value of the universal/local bit}}.
```

RQ_COR_1729 Interface Identifiers (Built-in): Present

RFC 3513 *Clause:* A ¶8-13 *Type:* MUST *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier from an IEEE 48bit MAC identifier.

Requirement: The implementation inserts two octets, with hexadecimal values of 0xFF and 0xFE, in the middle of the 48 bit MAC (between the company_id and vendor supplied id). [Note that the implementation also inverts the "u" (universal/local) bit from 0 to 1]

RFC text: {{Links or Nodes with IEEE 802 48 bit MAC's. [EUI64] defines a method to create a IEEE EUI-64 identifier from an IEEE 48bit MAC identifier. This is to insert two octets, with hexadecimal values of 0xFF and 0xFE, in the middle of the 48 bit MAC (between the company_id and vendor supplied id). For example, the 48 bit IEEE MAC with global scope:
[cccccc0gcccccccc|ccccccccmmmmmmmm|mmmmmmmmmmmmmmmm]
where "c" are the bits of the assigned company_id, "0" is the value of the universal/local bit to indicate global scope, "g" is individual/group bit, and "m" are the bits of the manufacturer-selected extension identifier. The interface identifier would be of the form:
[cccccc1gcccccccc|cccccccc11111111|11111110mmmmmmmm|mmmmmmmmmmmmmmmm]
}}. When IEEE 802 48bit MAC addresses are available (on an interface or a node), an implementation may use them to create interface identifiers due to their availability and uniqueness properties.

RQ_COR_1730 Interface Identifiers (Built-in): Absent

RFC 3513 *Clause:* A ¶14-17 *Type:* MUST *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier from an Link-layer Interface Identifier other than IEEE EIU-64 or IEEE 802 48-bit MACs.

Requirement: The implementation takes the link identifier and zero fills it to the left. [Note that this may result in the universal/local bit set to "0" indicating local scope].

RFC text: {{Links with Other Kinds of Identifiers. There are a number of types of links that have link-layer interface identifiers other than IEEE EIU-64 or IEEE 802 48-bit MACs. Examples include LocalTalk and Arcnet. The method to create an Modified EUI- 64 format identifier is to take the link identifier (e.g., the LocalTalk 8 bit node identifier) and zero fill it to the left. For example, a LocalTalk 8 bit node identifier of hexadecimal value 0x4F results in the following interface identifier:
[0000000000000000|0000000000000000|0000000000000000|0000000001001111]
. Note that this results in the universal/local bit set to "0" to indicate local scope}}.

RQ_COR_1731 Interface Identifiers (Built-in): Absent

RFC 3513 *Clause:* A ¶18-20 *Type:* MUST *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier for a link that does not have any type of built-in identifier. The implementation has a global interface identifier available from another interface.

Requirement: The implementation uses that global interface identifier from another interface in order to create the Modified EUI-64 format Interface Identifier.

RFC text: Links without Identifiers. There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers must be chosen that are unique within a subnet-prefix. {{When no built-in identifier is available on a link the preferred approach is to use a global interface identifier from another interface or one which is assigned to the node itself. When using this approach no other interface connecting the same node to the same subnet-prefix may use the same identifier}}.

RQ_COR_1732 Interface Identifiers (Built-in): Absent

RFC 3513 *Clause:* A ¶18-20 *Type:* MUST *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier for a link that does not have any type of built-in identifier. The implementation has a global interface identifier available that is assigned to the node itself.

Requirement: The implementation uses that global interface identifier assigned to the node itself in order to create the Modified EUI-64 format Interface Identifier.

RFC text: Links without Identifiers. There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers must be chosen that are unique within a subnet-prefix. {{When no built-in identifier is available on a link the preferred approach is to use a global interface identifier from another interface or one which is assigned to the node itself. When using this approach no other interface connecting the same node to the same subnet-prefix may use the same identifier}}.

RQ_COR_1733 Interface Identifiers (Built-in): Absent

RFC 3513 *Clause:* A ¶18-20 *Type:* MUST *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier for a link that does not have any type of built-in identifier. The implementation has a global interface identifier available from another interface or one which is assigned to the node itself. The implementation uses those global interface identifier in order to create the Modified EUI-64 format Interface Identifier.

Requirement: The implementation does not permit that any other of its interfaces connecting to the same subnet-prefix uses the same identifier.

RFC text: Links without Identifiers. There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers must be chosen that are unique within a subnet-prefix. {{When no built-in identifier is available on a link the preferred approach is to use a global interface identifier from another interface or one which is assigned to the node itself. When using this approach no other interface connecting the same node to the same subnet-prefix may use the same identifier}}.

RQ_COR_1734 Interface Identifiers (Built-in): Absent

RFC 3513 *Clause:* A ¶21-22 *Type:* MUST *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier for a link that does not have any type of built-in identifier. The implementation has no global interface identifier available for use on the link.

Requirement: The implementation creates a local-scope interface identifier that it is unique within a subnet prefix.

RFC text: Links without Identifiers. There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers must be chosen that are unique within a subnet-prefix. When no built-in identifier is available on a link the preferred approach is to use a global interface identifier from another interface or one which is assigned to the node itself. When using this approach no other interface connecting the same node to the same subnet-prefix may use the same identifier. {{If there is no global interface identifier available for use on the link the implementation needs to create a local-scope interface identifier. The only requirement is that it be unique within a subnet prefix}}. There are many possible approaches to select a subnet-prefix-unique interface identifier. These include: Manual Configuration, Node Serial Number, Other node-specific token. The subnet-prefix-unique interface identifier should be generated in a manner that it does not change after a reboot of a node or if interfaces are added or deleted from the node. The selection of the appropriate algorithm is link and implementation dependent. The details on forming interface identifiers are defined in the appropriate "IPv6 over <link>" specification. It is strongly recommended that a collision detection algorithm be implemented as part of any automatic algorithm.

RQ_COR_1735 Interface Identifiers (Built-in): Absent

RFC 3513 *Clause:* A ¶21-22 *Type:* MAY *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier for a link that does not have any type of built-in identifier. The implementation has no global interface identifier available for use on the link. The implementation then creates a local-scope interface identifier that is unique within a subnet prefix.

Requirement: The implementation uses one of the following possible approaches to select a subnet-prefix-unique interface identifier: (1) Manual Configuration, (2) Node Serial Number, (3) Other node-specific token

RFC text: Links without Identifiers. There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers must be chosen that are unique within a subnet-prefix. When no built-in identifier is available on a link the preferred approach is to use a global interface identifier from another interface or one which is assigned to the node itself. When using this approach no other interface connecting the same node to the same subnet-prefix may use the same identifier. `{{If there is no global interface identifier available for use on the link the implementation needs to create a local-scope interface identifier. The only requirement is that it be unique within a subnet prefix. There are many possible approaches to select a subnet-prefix-unique interface identifier. These include: Manual Configuration, Node Serial Number, Other node-specific token}}`. The subnet-prefix-unique interface identifier should be generated in a manner that it does not change after a reboot of a node or if interfaces are added or deleted from the node. The selection of the appropriate algorithm is link and implementation dependent. The details on forming interface identifiers are defined in the appropriate "IPv6 over <link>" specification. It is strongly recommended that a collision detection algorithm be implemented as part of any automatic algorithm.

RQ_COR_1736 Interface Identifiers (Built-in): Absent

RFC 3513 *Clause:* A ¶21-23 *Type:* SHOULD *applies to:* Node

Context: The implementation needs to create a Modified EUI-64 format Interface Identifier for a link that does not have any type of built-in identifier. The implementation has no global interface identifier available for use on the link. The implementation then creates a local-scope interface identifier that is unique within a subnet prefix. The implementation uses one of the possible approaches to select a subnet-prefix-unique interface identifier.

Requirement: The implementation generates the subnet-prefix-unique interface identifier in a manner that it does not change after a reboot of a node or if interfaces are added or deleted from the node.

RFC text: Links without Identifiers. There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers must be chosen that are unique within a subnet-prefix. When no built-in identifier is available on a link the preferred approach is to use a global interface identifier from another interface or one which is assigned to the node itself. When using this approach no other interface connecting the same node to the same subnet-prefix may use the same identifier. `{{If there is no global interface identifier available for use on the link the implementation needs to create a local-scope interface identifier. The only requirement is that it be unique within a subnet prefix. There are many possible approaches to select a subnet-prefix-unique interface identifier. These include: Manual Configuration, Node Serial Number, Other node-specific token. The subnet-prefix-unique interface identifier should be generated in a manner that it does not change after a reboot of a node or if interfaces are added or deleted from the node}}`. The selection of the appropriate algorithm is link and implementation dependent. The details on forming interface identifiers are defined in the appropriate "IPv6 over <link>" specification. It is strongly recommended that a collision detection algorithm be implemented as part of any automatic algorithm.

RQ_COR_9050 Interface Identifiers: Modified EUI64

RFC 3513 *Clause:* 2.5.1 ¶4 *Type:* MAY *applies to:* Node

Context: The implementation derives a Modified EUI-64 format based Interface identifier from serial links, tunnel end-points, etc.

Requirement: The implementation's Interface identifier scope is local.

RFC text: `{{Modified EUI-64 format based Interface identifiers may have global scope when derived from a global token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [EUI64]) or may have local scope where a global token is not available (e.g., serial links, tunnel end-points, etc.) or where global tokens are undesirable (e.g., temporary tokens for privacy [PRIV])}}`.

RQ_COR_9051 address: Loopback

RFC 3513 *Clause:* 2.5.3 ¶1 *Type:* MAY *applies to:* Node

Context: The implementation uses IPv6. The unicast address 0:0:0:0:0:0:0:1 is called the Loopback address.

Requirement: The implementation never assigns the Loopback address to any physical interface.

RFC text: `{{The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself. It may never be assigned to any physical interface}}`.

Annex A (informative): Bibliography

ETSI TS 102 351: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); IPv6 Testing: Methodology and Framework".

History

Document history		
V1.1.1	April 2006	Publication